EPISODE 1540

[EPISODE]

[0:00:00] JH: Hi, everybody. This is Jocelyn Houle. Welcome to Software Engineering Daily. Today, we're going to interview Robert Cooke from 3Forge. Welcome, Robert.

[0:00:08] RC: Hi. Thanks for having me today.

[0:00:11] JH: Can we get started maybe with a little bit of personal background and your journey? How did you come to start 3Forge? What problem were you solving? That type of thing, just to get grounded in the context.

[0:00:21] RC: Sure. Absolutely. Thanks for asking. I started 3Forge in 2010, but I'll start at the beginning, or at least my beginning. Ever since I can remember, I've always been into computers, even before I really knew what computers did. Maybe it was watching Superman, or something back in the early 80s. I just knew there was something magical about these boxes, and I wasn't quite sure what they did, but I really wanted to learn more. I got my first computer when I was about six or seven-years-old, and really none of my friends, or anyone had any idea of anything to do with computers.

By just typing keystrokes and reading some books, I finally started to understand what these magical boxes did. Then started building software at a pretty young age. I think I did what a lot of kids wanted to do, which is I started building video games, platform games, things along those lines. Then as I got older, I built an accounting system in my early teens. Pretty much my whole life, I've been building software.

[0:01:21] JH: Which is like a game. This is like a game for adults; accounting.

[0:01:25] RC: Exactly.

[0:01:25] JH: For the business world.

[0:01:27] RC: Yeah. In fact, I mean, I remember when I was a kid, I would always bug my parents like, “What can I program? What can I program next?” It was very tough to come up with ideas. That's been my life story is looking for a problem to solve, something to really sink my teeth into. I have to admit, when I first graduated college in 2000, I went out to Silicon Valley, and I thought that would be very interesting. It was. There's a lot of interesting things going on.

Then I got what I thought would be a very boring, stodgy job at Bear Stearns. It turned out that actually, fintech is quite interesting. You're challenged with a lot of unique problems, which I would say, in concert, makes I would call the lifelong pursuit to build great software. I call it the four Vs, which is velocity of data, the volume of data, the veracity of data, and the validity of data. Those four things in concert, again, the idea that data has to be valid and all these things, that really got me thinking, as soon as I joined my first tier one bank, how can we solve these problems? How can we build a platform for this?

I moved from company to company in fintech, always thinking about this problem and exploring and learning and trying to work in as many different departments as I could, front office, middle office, back office doing tax rebates. I spent a lot of time on high-frequency trading. All of this trying to learn everything I can. Really, again, going back to when I was bugging my parents, “Hey, give me something to program,” I'm really thinking about what is the problem? What is the problem that the industry is facing and how can I try to objectively define that?

Then I finally in around 2010, 2011 understood what it is that I felt was missing. I stepped outside. In fact, I could even say that I understood towards the end of I'm working in these banks, but it was very hard to, I would say, to pursue that when you're sitting at a financial company, because I felt it was actually very data agnostic. Once I understood what the problem was, I said, this is a data agnostic problem that we're looking at here. This has nothing to do with finance in particular. I stepped out, started 3Forge, and then that's really been the last 11 years, it's been focusing on that, on the problem and coming up with this platform that we call AMI.

[0:03:54] JH: I was just thinking about what you're saying, Robert Smith of Vista Equity. Yeah, I think so said, all software products taste like chicken at some point. I feel like, that's what you're saying too, is that the problem statements converge at some point around fintech.

[0:04:10] RC: Yes. I would say that fintech – I look at fintech as the canary in the coal mine, because other than maybe for military purposes, I think computers have been used in finance for about as long as anything. I think as other industries mature, they're starting to see the same creaks and cracks that the fintech has seen, and the same challenges of fintech has seen. Again, I look at it as fintech has been doing it longer, and so I think looking at the problems that are faced in that industry, give us an indication of what other industries are going to face as they move forward. You're right. At the end of the day, or I guess, I think you said, Robert Smith. Everything does start to taste like chicken. I do like chicken, by the way.

[0:05:00] JH: Perfect.

[0:05:01] RC: It's not a bad thing. But yes. At the end of the day, I think it's important to classify software based on the problem set, as opposed to the very specific thing you're trying to solve.

[0:05:16] JH: Let's talk about that problem statement for you, because I'd love to just talk a little bit of time to talk about what 3Forge does now and what problems it's focused on. To me, when I met you guys and I was looking at your materials, it's, I guess I'm into quoting people today, but it's like, it tastes great, less filling, because it's highly performant. This is what we always hear in business, when you're a product manager. We want something that's enterprise great, highly performant, works the same in sub-second performance worldwide. Then the next thing you hear is, "Well, we want to enable our end users. We want to make it easy." You're constantly ping-ponging between these two things. That was something that stood out to me that you're really trying to conquer two seemingly opposed goals, but probably more goals than just those two. Tell us a little bit about what 3Forge is trying to solve and why.

[0:06:09] RC: Right. Well, I do look at it as, well, I guess going into the culture a little bit and how I think culture matters in terms of the type of software, or the quality of software you build. First off, I've always had this opinion that building software should be about saving the computer in the user's time, not the developers, the ones that are building the software. For the platform that we're building, if it takes us 10 times the amount of time to build a particular component, but that component can be 10% faster and 10% easier for the end users to use, I think that's well worth it.

Just like, why so much money is spent on a movie. Because someone at some point realized, wait a second, even if you spend 10X on the budget for this movie, the viewership is global. The number of people that are going to consume that movie is so huge that it justifies the cost for doing that. I think that's core to how we build software, or at least how I've always built software. Then I try to persuade that across the firm, I would say.

Instead of me thinking of it as – instead of going back to that analogy with the movies and the viewers, I think of it as actually, first off, you've got the end users of our software, but then you've also got the computer itself. The computer can do billions, I mean, literally billions of executions per second. I think one of the tragedies is a lot of those calculations are spent needlessly executing code that is inefficient in most cases.

Billions is a lot. That's a pretty big number. We're talking executions per nanosecond. I just have always felt that if you really focus on each part and you break the problem down and you try to make it as efficient as possible, it's actually incredible how much you can eke out of a computer. Maybe this goes back to the 80s when I was trying to build platform video games, and you literally had a 23K, 23 Kilohertz machine and every single instruction, you had to make sure it's correct. I still take that philosophy with me today.

One thing I have learned being on both sides of the vendor relationship, purchasing vendor software and building vendor software, is that performance really is critical. No matter how performant the system is, and no matter what the capacity of the system is, as an end user, I would find a way to leverage that. As a vendor, I've definitely seen plenty of platforms that that becomes the limiting factor, is that you can't process the amount of data you want, or that ultimately becomes the limiting factor, how you can use it. I look at it as a vendor. Being able to build something that you can do more with less is always better. Going back, yes, I think we definitely strive to be, to taste great while be less filling. Yes.

[0:09:16] JH: Let's talk a little bit about that. Then I want to have a quick discussion. Well, maybe not quick, but then let's talk about architecture. I'm not going to overlook that, but if you think about these two quadrants, and I'm using shorthand here, highly performant, scalable, big shoulders, right? Processing. Then the other thing is enablement, making it easy to create the

applications. Let's start with that first bucket. What's special about 3Forge that people should know technically? What technical decisions have you made to support that goal that's unique to 3Forge?

[0:09:52] RC: Well, I think when we talk about performance, it comes down to – I know there's this concept of premature optimization. I'm actually not a big believer in that concept, because I found once you build something that isn't optimized, it's actually quite difficult and quite scary to go back and optimize those things later. I think you're much better trying to focus on that upfront. That goes tenfold for scaling. It's extremely difficult to build a system that is not designed to scale and then make it scale later on.

I would say that almost in every case when I've seen someone take a product and make it scale that wasn't designed to scale from the beginning, it looks like a whole bunch of band aids on top of each other to try to get that to work. You end up actually having a lot of costs overhead just trying to do the scaling itself. When we built our platform, and I'll talk about the platform, its capabilities a little bit, but when we built the platform, scaling was what we spent probably the first three or four years on, was how do you focus on scaling? Really, how do you fundamentally get a piece of data from point A to point B as quickly as possible and as reliably as possible?

[0:11:10] JH: You have some contrarian philosophies around development processes. I guess, what some things that we hold is now is just common ideas in software development, or platform development. I really do like your point of view though on some things are hard to optimize after the fact. I think we do have a prevalent and sacrilege to say. For the most part, just doing things very quickly and editing and changing on the fly works. It works really well in the application world, for instance. But there are some really large-scale platform problems, where you should at least start out trying to optimize. Start out trying to hit the mark, because it is hard to change it later, or at a platform level, or doing something difficult, like moving data from one place to the next as fast as possible. I like your contrarian point of view on this. I encourage you to bring more of those to the rest of this conversation.

[0:12:06] RC: Okay, I'll do my best.

[0:12:09] JH: I can tell you're hard to draw out on this, but – Let's just think about then on the end user side. Maybe you could share some typical use cases. How do people use – people, regular people, consume 3Forge?

[0:12:25] RC: Right. Well, our users, what I was focused on is going back to when I was an employee at these large tier one companies and smaller firms was I felt like, I was wasting a huge chunk of my employer's money solving the same problem over and over again. It's as if I would be working on a, let's say, a back-office application. For some reason, the back-office ecosystem, they have a certain flavor of how they like to do things, and then you're working on HFT and they have a different flavor of how the tools they like to use in this and that.

Again, 90% of the problem is overlapping between them. What I was trying to focus on was how do we let our users build applications where the 90% of overlap can be handled by the platform and then they can instead focus on the intellectual property; the actual business intelligence that they need. Really, people use our platform to build applications, and these applications can be anything from a simple reporting tool to a real time monitoring application, to some sort of database aggregation solution, all sorts of different things.

Really this is, this is the struggle, I would say, with our product. By the way, we were used at many of the world's largest firms in very, very sophisticated use cases. We're also used at small organizations as well. The whole focus has been, how do we articulate this product? I would say, we came up, in the way I've looked at it now is we are a layer. If you think about the history of computing, it's come down to several layers. I think a few things about a layer is once, and I'll give some examples of layers first.

I would say, the first layer is if you go way back into computing, you've had hardware machines and anytime you wanted to program it, you're literally changing hardware. Then someone says, “Well, wait a second. We can use software, instead of just using memory to store the data itself, we could actually store the program and software itself, then you end up with compilers.” Each one of these becomes layers. Then at some point, you get operating systems and then you have databases. Then you get a graphical Windows-based operating systems, and so on and so forth.

Each one of these layers almost becomes ubiquitous. Once it's there and once it works, you forget about it. It's critical to actually moving, I would say, the computer civilization forward, if you will, right? Because imagine trying right now to build a product without using an operating system. It could be done. It actually could be done. You could also build a data warehouse without using a database. It would be insane to do that though. That's how I feel about how most software is being built today. Because again, it's as if every time we start a new greenfield project, it's sit down and solve the same problems over and over again.

[0:15:52] JH: Yeah, let's double click on that, because I like this idea of the overlap, or solving the same problems. Can we break down what some of those problems are, because I'm not picturing, I don't think entirely. One thing that jumps to mind is pipelines and integrations, but there's probably a short list of things that isn't that overlap. Typically, what are those things?

[0:16:10] RC: Right. Well, I think one thing is integration with your underlying systems. Typically, our customers will have many different databases. We have adapters to all of the different databases, so we can just integrate with those directly. Then we focus a lot on how do you actually access multiple systems at the same time? That's an important thing that needs to be done, which is query system A, get the result back, use that result for query system B, get the result back, query system C. We call that a cascading query.

At the end of the day, that's a problem that is solvable, but ends up being solved over and over again. Because again, once you get to large organizations, they're not just sitting on one particular database, or sitting on multiple databases. That's one example. Another one is how do you handle the movement of real-time data and making that available to users? That's actually a very interesting challenge. It's thought of as a very expensive problem, because to develop it is typically takes a long time and it's fraught with air. If done right and done once and done to perfection, then everyone can just use it. That's another thing we focused on is we can connect into pretty much every type of real-time messaging system, and then you can build real-time dashboards on that. That's the second problem that I feel comes up all the time.

By the way, most times what happens is they say, “Well, it's too difficult to build something real time, where the website will actually react as data is changing. Instead, we'll do polling, or we'll hit a refresh button, or we'll just hope the user hits refresh themselves, or something like that.”

By doing that, you're actually creating a lot more work and a lot more conditions. Well, what if they hit refresh and now the tickets have been sold out, blah, blah, blah? If you just do things in real time, it actually saves you a lot of these concurrency issues.

[0:18:10] JH: That's right.

[0:18:11] RC: That's a second problem that we've worked very hard at solving. The third one is how do you manage historical data, like large sets of archival data? Basically, take that data, store it, store it permanently and be able to retrieve that. Do we have a historical petabyte database, which basically addresses that?

Then I think the biggest thing is that now, let's say, you've made a decision, I'm going to use this for my database, I'm going to use this for my real time, I'm going to use this for my web server, so on and so forth, these components weren't all designed to work together perfectly. Again, the computer and the developers end up spending a lot of time integrating these components. I want to say, the computer spends a lot of time, you almost inevitably have lots of adapter and glue code between these things that slow things down, mapping, you get only the least common case value out of this multitude of products.

Instead, we've said, we're going to basically have a full stack platform. This gives you your data virtualization layer, this gives you your real-time dashboards. I haven't talked about workflows yet. This gives you your workflow solution. This also gives you your archival historical database, all in one installation. You install this one thing, you have it all, it's designed to scale, and now you can start focusing on the intellectual property.

How do I put this? Our customers have practically unlimited dollars when it comes to how much money they spend on technology. Some of our customers, the joke is some of our customers, their IT budgets are greater than several countries' GDP, you know what I mean? We're talking billions of dollars being spent. So much of that money has been spent solving these problems over and over again. They are now turning to our solution to start to build and replace a lot of these things. Because at the end of the day, even for these companies with extremely deep pockets for building IT and building software, eventually, these budgets creep up.

[0:20:30] JH: Yeah, yeah, yeah. A couple million here, a couple 100 million here and there starts to add up after a while. You've got this overlap world. Things that people have typically have to resolve all the time, and this is things like integrations, connectors, access and permissions. It's the speed of transforms and queries, and it's getting that real time data and that analytical data teed up and ready. It's all of those tasks and probably more. Now, does that also include deployment scripts and making it very easy to push button, put install? Or is that something that you guys do with **[inaudible 0:21:08]**?

[0:21:10] RC: Right. Well, our thing is DevOps. Well, it's actually funny, because the original use case for our software was around DevOps. Again, DevOps, the funny thing is when you start thinking about what it means to deploy software and all this stuff, that's actually just a subset of the same problem. In fact, most DevOps tools – we use our platform for all of our DevOps. We use it for all of our deployments and all of our testing. It's actually great, because unlike most DevOps tools, everything is real time. As soon as someone does a deployment, we see it show up within milliseconds. If a build fails, we know milliseconds later, as opposed to screen refreshes and things like that. DevOps is a subset of it.

Now with that said, DevOps is a very – there's a lot of vendors focusing on that space. Really, the reason people turn to our platform is because they're looking for some bespoke solution. We're not necessarily saying, go and replace your typical deployment tool and build it on us. Although, you could probably build something a lot more customized and a lot more interesting. Again, there's a lot of vendors out there focusing on that.

Really, again, where people are choosing to use our software is because they are – the alternative is to hire a team of developers and build this bespoke solution from the ground up in turn. Instead, they can choose us, build it in a fraction of the time. It's much more stable, much more reliable, the amount of code they're building is a fraction of what it would be otherwise, etc., etc.

[0:22:51] JH: Okay, I think I'm a better understanding of this – for application development platform and data movement platform, right? It's interesting. I'd love to hear a little bit about just a little bit more about your origin story. You talked about your personal background. Just having been around this business a long time, what you're suggesting is, “Hey, I'm going to get in there,

pay a large financial with a lot of sensitive data and IP. Why don't you put me in between everything and I'll show you what the software can do?" That seems like a tough for sale. How did you make your first – How did you target your first customer and start deploying this? It's a big idea.

[0:23:32] RC: Yeah. Well, so first off, my opinion was, I think just like premature optimization, which I don't really believe in, I also don't believe in going after the low hanging. Because if you go after the low hanging fruit, it's actually much easier to go after the low hanging fruit. That's why it's the low hanging fruit. The problem is, if you build a solution that only lets you get to the low hanging fruit, then what do you do once you've gotten the low hanging fruit, right? I felt that I needed to prove our solution could solve for the fruit at the very top of the tree, the hardest to get to. That really is, we're talking the top tier one banks. Those are the firms we went after.

In fact, our first customer was the number one largest investment bank in the world and they were using us to manage all of their orders and executions. Around 350 million per day. That's in just a 390-minute trading session. This is many, many executions, many transactions per second that we're monitoring, managing in real time. Once we knew that our software could handle that in scale and handle that use case, it made it much easier for us to now in confidence, go to the small organization and say, "Look, this platform can easily solve what it is you're looking for."

Now, how did we get into the largest organizations? It is very difficult to build. I would say, it takes a lot of patience and a lot of time and a lot of understanding to build a real-time dashboard that can handle hundreds of millions of transactions and display that. That's what we do. This is something that even the largest firms with very, very deep pockets have not really been able to solve in a generic way, and we have.

[0:25:35] JH: Do you have a moat around that – that's a moat that you can build right around this technical capability. I think a lot of listeners would really be interested, too. I think I know the answer, but let me ask you this. Right now, times are tough for a lot of software companies, especially younger software companies. It always creates a flight to enterprise sales. Everyone's like, "I'm going to shoot to the top of the enterprise." They've got the big budgets. Everybody's clustering around like, "Hey, let's talk to those guys." For you, I mean, you had a lot

of product completed, though, when you closed that first deal, so that you could go for the highest fruit. Is that right?

[0:26:12] RC: Yeah, absolutely. That's what made it, I think – that's what I think made the 3Forge story fairly unique. I think, to sit back and say, and by the way, when I was at these firms, I would talk to them when I was employees, I would talk to them about putting together budgets and putting together team and building this out. I usually could get the budget. Not usually. I'd be able to get the budget and I get the team and we'd start building it out. The problem is that you'd have these business cycles. Every two to three years, managers change around. I think the average employment time in a particular position for a manager in fintech is around 30 months.

It would be very hard to actually build a piece of software that would take half a decade. That was very difficult to do. The software that's changed the world, that's really changed the world, most of that software has taken half a decade or more. I have an appreciation for that. Whether we're talking about the game engines, or we're talking about the operating systems we use today, or the databases, all these things, they take time to build, to really build well and build them reliably and fast. I knew this is what would be needed.

Yes, we had to spend a lot of time and a lot of resources internally building this out before we could really go to market, which I think is unusual. I think most firms, instead, are looking to try to get product out the door in 12 months, and they turn to open source and just try to get something out the door. For me, I really do look at this as this is my – I hate to say YOLO. You got one life to live. I really believe that there's a chance to make a change for the better in this world and produce a new layer that allows people to build applications much faster. That's what we focused on. Took five years, five years. Takes 10 years, 10 years, 20 years, whatever. It's just, this is what needs to be done to move us forward.

[0:28:15] JH: Let's talk a little bit about that idea of the layer that you're building. This is a fun game I like to play, in which we have no diagram, but we just talk through the architecture. Let's just talk through the flow, so that we understand what this proposed layer is, what's happening. If you think about your marchitecture type of diagram, on the left-hand side, we've got a whole bunch of data resources. Things are happening. You've got all your files and your data sources

and your scripts and your messaging. Then walk me through from left to right, how the AMI relays, centers and web servers work. What are those process steps that are happening in between those data sources and then you've got your end users.

[0:29:01] RC: Right. Studying and looking at – what I did is I looked at as many different architectural diagrams, how systems were built. Tried to consume that and map those to logical ideas. I think one of the fascinating things, by the way, is when you talk to most architecture teams, they – I feel a bit condescending saying it, but they feel like they're solving a unique problem that no one solved before. Funny thing is then you go to the competitor across the street and they're solving the same exact thing and they're solving it at almost the same thing. I think that's what happens when you start to get high level and you look at this –

[0:29:40] JH: I'm looking at you microservices. I'm looking at you.

[0:29:45] RC: Right. After a while, I distilled it down to a few logical components that you need and then you need to make these logical components work together. For me, you need the ability to do the transfer of data in real time, which means being able to ingest streams of data. You need to be able to ask a question to a system and get an answer back. That's request response. There's some correlation that needs to take place there. You ask a question, that is a correlation ID. It might be some period of time. You get a response back. You need to correlate that so you can answer it. I'm not going to get too much into the currency part of this, but that's an important thing.

Then you have workflows, where people need to be able to enter data and then have that travel through some decision tree and then ultimately, go into a system. In fact, if you really focus on those three things, the ability to ask a question about your data, the ability to stream data and look at it in real time, and the ability to enter data and manage workflows across users, that's actually just about everything there is when you're talking about enterprise software.

The analogy I often make is just to make this clear is think about email as an example. There's really a few things you can do. You can sit there, you can open up your email and you can search for a particular email. Ask a question to the email server and it comes back and it says, "Here's the emails that have that keyword in it." That's a question, response. You can also sit

there and stare at your screen. When a new email comes up, it shows up. That's the ability to manage streaming data. You can also write an email. That's the ability to submit data. That's just one. I don't want to get caught up. Sometimes I feel like, when I give an analogy, it sticks almost too well. That's just an example of these three components.

In fact, the military has this concept, OOGA. Observe, orient yourself. OODA. I'm sorry, OODA. Observe, orient, decide and act. They spent probably billions of dollars coming up with this. Really, that's also how the military works. You observe something, and then you orient yourself and then you make a decision, you act. That actually falls into those three buckets as well.

In any case, the architecture really, when once we understood these different flows, we could think about the architecture, and the architecture came down to three things. One, you need to read what we call the relay layer, which is ability to interact with the existing systems. That's kind of, I almost call it the immune system. That's actually what connects to the outside world to – the when I say the outside world, I mean, the electronic, the existing systems. Again, that has the ability to submit data into the systems, ask questions of your data and consume real-time data.

[0:32:40] JH: Let's take that example. Yeah, just to work through a live example. Let's take the example of transactions that you just mentioned. A lot of transactions will run on thesis, or fidelity way down in the guts of the system, right? Those transactions are hitting thesis, let's say. This relay is able to talk to thesis and pull that out, or work with it?

[0:33:02] RC: Yes.

[0:33:03] JH: Okay, thanks. I just want to make sure I understand what the piece parts are.

[0:33:06] RC: Right. This is where the architectural question comes in. There's two ways to set this up. It could either connect to it and consume that data in real time, basically get a drop copy of that data. Or, it could instead passively ask when the user's interested, right?

[0:33:21] JH: Right. Okay. Thank you.

[0:33:21] RC: The users, just I want to know about this transaction, ask about it. That's what the relay layer is responsible for. Each one of these layer scales, etc. That's the relay layer. The next layer is the – what we call the center. That has the ability to do caching and all of those things you need, so you're not necessarily going back to the underlying system every time, so you can cache if you need to. Again, these architectural decisions that need to be made, they're use case specific. But this is another important building block. Have to have the concept of caching. That's really what the center is about. It also manages what I call the HDB, which is our petabyte historical database. That's the ability to take this cache at the end of the day, store that and then be able to retrieve that.

Because a lot of times what happens is our caching system will sit somewhere. It's pulling in all this real-time data for many different systems. At the end of the day, you just want to take that data and store it. Now, we take that data, push it into our HDB. Then you can retrieve that. It's basically there for posterity sake.

Then the third thing is what we call the web layer. The web layer is actually composed of three components itself, if we really get into it. Ultimately, the web layer is what's responsible for allowing people to interact with this data. This is actually managing the building of dashboards and the viewing of data and managing workflows and all of that stuff. That's what's actually managing the GUI itself.

[0:34:44] JH: Let me ask you this about caching, and some of the guts of the system. When you think about the largest multinational financials, and if I had this deployed worldwide, even if it's all on-prem, you've still got pods in every continent will be running a pod. I would assume with an implementation of 3Forge, or what happens? I've got Hong Kong and I've got Detroit trying to process transactions, and they want to see it in one view. Is there special goodness inside of your architecture that's supporting that type of, I guess, the problem of time? It's the problem of time.

[0:35:28] RC: The one thing you can't buy. Well –

[0:35:32] JH: But you know what I mean. Just like in a game, in a multi-massively multiplayer game, if you knock me in the head with a mallet in Hong Kong and I'm playing in Detroit, it has

to all keep happening seamlessly in what feels like real-time to both players. I think, the same thing is true in these financial transactions. You want it to be as close as possible to real transactions.

[0:35:55] RC: Yeah. Oh, yeah. I mean, this is a pretty common use case for us is that global interaction. Without getting too deep into the architecture, now there's a few different ways you could solve this. You could have a single caching point where everywhere – and by the way, that's one of the important things about the relays is you can put these relays in different locations. They collect data locally. Then they're responsible for distributing this across the pond. We focus on never being a slow consumer. What I mean by slow consumer is if your cache is hooked up directly to your system and that cache can't keep up, you don't want that to have trickle down effects where now your external systems with perspective to 3Forge are basically backing up, trying to push data.

Basically, the relays make sure that we can always keep up. They're consuming data locally. Once those relays have the data, you can now configure it to distribute that data however you like. Distribute to a single center, and then everyone views that center. A lot of times what you do is you have multiple centers located around the world, so that they can view data for their center locally, will be very fast. Then if they're interested in viewing data from a different region, then basically, they're viewing that other center and then they're going to see that latency, but only when they're looking at that data.

[0:37:12] JH: Okay. All right. That helps. That's interesting. I think it's an interesting problem. The problem of time across these systems.

[0:37:19] RC: It is. That problem is, it's an interesting problem, but it's a solved problem. Again, going back to this layer concept is that to be solving these problems over and over again and having teams sitting down and thinking about and reconfiguring and redesigning and redoing this over and over again, that's just a waste of employer's money. This is a solved problem. That's what we focused on is taking that solved problem, buttoning it up and making it easy to deploy and for people to use.

[0:37:57] JH: As its own layer. That's the layer, right?

[0:37:59] RC: As its own layer. Exactly. Set it and forget it. Focus on the intellectual property. Right. Because if you think about what we're really talking about here, this has nothing to do with intellectual property. If you said, what firm would identify themselves and say, “Yes, this is what we do that makes us special, the ability to concurrently manage data across different regions.” That's not the definition of really, a bank. A bank is to manage finances. That's not the definition of insurance. That's not the definition of health care. That's not what they do. That's not what makes them different.

The idea that that people are spending a lot of time thinking and solving that problem inside these organizations is a bit of an identity crisis there, right? It makes sense for a vendor to solve this and then for other people to just use that solved solution.

[0:38:49] JH: That's interesting. Yeah, there's a pendulum swing that goes every few years between, it's all about subject matter expertise. It’s subject matter supported by software and then it goes the other way. It's really software supported by a subject matter expertise. I think what you're saying is, let's separate those two. There's the domain and then there's this set of problems that have a solution that you want to make repeatable. That's, I think, I'm saying it to make sure I understand it myself. I think that's what you're saying.

[0:39:17] RC: Yes.

[0:39:18] JH: Along with this, is you have a concept of a consortium. Is that right?

[0:39:22] RC: Mm-hmm.

[0:39:23] JH: What is that?

[0:39:26] RC: This concept is that I basically look to the customers and the problem set of the customers drive the architecture and how we build and how we improve the product. In fact, when I was proposing it – and by the way, if you go to just about every large organization, any organization, let's say over 10,000 employees, at some point, they created an architectural team

that was supposed to build a platform similar to what we have, take input from all the different groups, build this platform, and then use this internally across the organization.

I don't know of a single organization that actually succeeded at that, but that is something that almost all large companies endeavored on at one point. Actually, it makes a lot of sense. The issue is, though, that for none of these companies, did it really make sense for just that company to focus on it. Again, I was at several of these companies, and I would be on these committees and exploring these ideas of how do you build this? Because let me step back. You're a manager, and you've got a few hundred people working for you, and you've got 15 different teams.

At some point you realize, well, this team over here is trying to build a PDF report, and this team over here is trying to build a PDF report. This team here is trying to figure out how to scale data globally. That team over there is trying to figure out how to scale data globally. Can we have them work together? In fact, can we just take the best and brightest from all these teams, have them work together, solve this problem once, and then everyone can use that? It's a logical step. It makes sense for management to think that way.

The problem is, again, the ecosystem inside these companies, because they're not technology, they're not software companies, it's hard to solve that. Because at the end of the day, what drives their bottom line is their intellectual property, not these architectural platform. I believe that that concept is correct, but I believe where it was being implemented was incorrect. That's why I said, I need to step outside, start an organization, and then once I've got this platform in place, go back to these organizations, find these managers that get that concept, talk to them, and then bring this platform in, and then pretty much follow the same concept. Once the platform's installed, then we can start to listen to the feedback from all of our users, and take that, look for what's common, what are the common requests, and build that in and improve the product. In a way, it's just the logical evolution of software. It's just this simple concept of write once, use many times.

[0:42:12] JH: Interesting. Let me ask you this. When I first checked out your website, and I saw some of the dashboards, I thought, “Oh, this is one of these high-end finance plays.” I'll tell you why I thought that, because finance loves these extremely dense dashboards. So many people

want less information in a dashboard, but others want more. What are you hearing from customers in terms of the types of dashboarding they're looking for?

[0:42:43] RC: You have to break it down into two categories. I think this is a missed concept a lot. But when I at least took my UX classes back in the 90s, you have to ask yourself, is this user interface designed for ease of access and for casual users, AKA just simple consumers? Or is it part of their livelihood and part of their jobs, and do they make money using this? Based on the answer to that question, you go down two very different paths.

Again, I think this has missed a lot. Spend a lot of time studying UX and GUIs. I think organizations, or when you're building dashboards, that's the first thing you have to ask yourself, and we can do both. Now, we focus on the dashboards where the end users are using that as part of their daily lives, or their daily jobs to make money. In that case, getting more information on the screen is better.

I'll give a simple example. I don't know if you've ever been to an airport. I travel all the time now for work. I don't know if you've ever been to the airport where at some point, you're like, can you choose your seat and then they just flip the screen around. They just turn the screen around and say, “Here, choose your seat.” Now, if you look at your app, it's a very simple, very clean interface. You can only see a few seats on the screen. They've got nice little icons, this and that. Then you see what the employee sees, and they have a special icon for the emergency seats. They can see which seats can have pets in them. They can see all the information. This is by design, right? The same information is being presented in a much simpler, cleaner, less cluttered way for the casual end user.

[0:44:48] JH: It's edited.

[0:44:49] RC: Then it is for the employees. Neither is right or wrong. It's just depending on who's consuming it. Now, the thing is traders, and yes, we had traders. I'll never forget, when we first shipped our product, we actually had a minimum font size of eight. You would go. It’s very customizable in terms of what you could do. Our minimum font size was eight. The traders were like, “I want six.” Now, I don't know how often you've tried using a font six, but that's tiny, however –

[0:45:15] JH: That's not quite right.

[0:45:17] RC: You can get more information on the screen and you can do more work. You can manage a larger plane, to make the analogy. I think that's why maybe the website is to be added, maybe our website's too focused on those use cases. That is a majority of our users are really using it to look at large amounts of data. Yeah.

[0:45:39] JH: It's definitely appropriate. It's something that it's just unique to this particular world, where most of your customers are. I just wanted to ask a little bit about it. You're thinking, I'm interested on the – well, first of all, let me – I want to ask a little bit about the business side of the company. Before I do, let's talk about some of the technical choices that you made. You have a particular point of view, right? I can prompt you with one thing, but you probably have more, of this is a technical point of view that really shapes the product and the product roadmap forever.

There are some choices that you make that then become an internal philosophy, a point of view. One for you is you're not using open-source software. That's one. What are some other choices, or decisions you've made?

[0:46:34] RC: Well, I would say it starts with that. That's a very big decision that I made was to not use open-source software. That really stemmed from the, again, going back to performance and being able to have full control over the customizations we make. Building our own web server, we didn't do just because we felt like it. We knew we needed to, because we wanted to have then –

[0:47:02] JH: Let me ask you this, though, about the – sorry to interrupt you, but the open source, I'm just curious. I'm trying to picture you and you're like, as you're designing your architecture, did you wring your hands about this at all? Or were you like, “No, we're just definitely going to go down a more proprietary path”?

[0:47:15] RC: No. Well, we started off in the beginning using some open-source software. Then once you started going through the performance benchmarking and things along those lines, we

quickly realized the optimizations we needed to make, pretty much precluded us from using most of the open-source software that we wanted. I can, without getting into too many details, it's just we found that open-source software is generally just trying to solve the problem, not solve the problem ideally. We want to solve the problem ideally. Again, we think of this as something where our code gets executed trillions of times a second across the globe. We want to make sure we're not wasting our customers' electric bills.

[0:47:59] JH: Yeah. One way you keep it quick is by keeping all this – a lot of times, there's just an efficient code. People are using open-source libraries, and it's not as distinct as it should be. Are there other ways that you're focused on making sure that the proprietary advantage turns into advantage for your customers?

[0:48:20] RC: Well, the other thing you mean around, what other advantages do we get by not using open source?

[0:48:26] JH: Yeah, yeah. Especially around speed, I think, is one area.

[0:48:30] RC: Right. Well, I think one of the things that slows a computer down is, and I can get very technical very quickly.

[0:48:37] JH: Let's do it. This is the place.

[0:48:41] RC: I think at a high level, one of the greatest wastes of computing resources is the copying of data. The copying of data is how you can build a library that you have no idea how other people are going to use it. That's something you can protect yourself when you write code. If I'm writing code, if I'm building an open-source component, and I have no idea how people are going to use these things, and I want to basically share some internal piece of data, copy it and give it to the consumer. When I say the consumer, I mean the developer that's using my open source. That ends up being very expensive.

If you really start to break it down and look at it in a profiler, that's a huge cost. This is just a bullet point one of many bullet points that I can go through where you take that cost. Then on top of that, like I said, you now suffer from the worst-case scenario, where you have all these

different libraries, and you can only get the minimum functionality based on what's shared across all of them. You know what I mean? If you're trying to do –

[0:49:51] JH: To race to the middle. It's a race to the middle?

[0:49:53] RC: Right. Exactly. If you're trying to support a new type of data structure, but this library doesn't support it, now you're in trouble, or you end up building all these different plugins and things around that. On top of that, I've also found that if you use a particular open-source library, that library is probably also using other open-source libraries, which makes sense. It's part of the open-source community. The problem is, now you run it to again, this gets technical, you run into two issues. You either get library version conflicts as you've got two different libraries that are trying to use the same library, or the other thing is maybe they're using two different versions of open source to solve the same thing.

Let's take something simple, like we just want to do, I don't know, string manipulation. You've got one open-source library that decides to use a particular library for string manipulation. You've got another open-source library that's using a different one for string manipulation. Now, one's going to perform better than the other. But then on top of that, now you've got two sets of libraries floating around that ultimately, are doing the same thing, but they're necessary. It actually ends up being a very inefficient solution.

Everything I've talked about, the real-time database, the messaging solution, the customized web server, the reporting solution, the historical database, everything is under 10 meg. That entire package is under 10 megs. Most apps I download on my phone, something as trivial as like, finding – plugging in my electric car.

[0:51:36] JH: Your quality index, for instance.

[0:51:37] RC: Your quality check, exactly. Yes. That's very germane. Yeah, something along those lines. That can be a 50 meg app. It makes no sense. The reason it is 50 megs is because there's tons and tons of libraries sitting behind there.

[0:51:51] JH: Yeah, interesting. Yeah, let's talk a little bit about the business. You've got customers, anybody would be super excited to have in the software world. You've got those customers. They're reasonably happy. What do you see as – I have some other questions along this vein, but what's next for 3Forge?

[0:52:09] RC: We focused mostly on the tier one banks. What's been interesting is that as employees from these tier one banks have moved to other industries, like hedge funds, buy side, things along those lines, RAAs, they've reached out to us and now they've become customers as well, part of the consortium, if you will. We've been able to expand in other industries. But really, we've been very heads down building out the platform, making sure we have a viable product, or I'd say, a complete product that can be used.

By the way, I don't believe in MVPs either. I don't like the concept of a minimum viable product, because then, that ends up limiting what – You end up going out with a minimum viable product and then it's hard to expand on that later. I think you really almost have – unless the M stands for maximum. Because really, we've focused on –

[0:53:11] JH: Maximum viable product. That's the name of your speaking tour.

[0:53:16] RC: I like that. MVP. Yeah. Yeah. I think, we've focused a lot on having a complete solution that solves what, I would say, the most demanding customers in the world need. Then really, we looked at later this year in 2024 is really going out and starting to explore other industries. Because again, the platform itself is completely data agnostic. We've just been focusing on the most challenging use cases up till now.

[0:53:45] JH: How is it licensed and is there any concern around – how does it work with the cloud data?

[0:53:52] RC: Yeah. Well, the licensing is basically, a SaaS model. Again, we look at it as a consortium. It's a subscription model. The idea is as we add new features, those get included. All of our customers are on the same version of the product. Whether you are managing 360 million transactions a day, or 30,000 transactions a day, or you're using us to replace an Excel

use case, it's all on the same platform. We basically have “one branch of code,” that all of our customers are running on. Whenever we add a feature, all of our customers get it.

[0:54:30] JH: And they’re consuming. Okay.

[0:54:33] RC: They all get that. Yeah. That is why we call it a consortium. Because if after we decide, or look at what all of our customer base is interested in, we add that to the platform, that then gets rolled out to all the customers. We always make sure it's backwards compatible.

[0:54:47] JH: Okay. The SaaS model though is you'd like credits I'm consuming, or what am I paying for in this? Processing?

[0:54:54] RC: Oh, I see. Yeah, that would depend on the type of customer. I mean, it is different if it's a firm with 10 employees, versus a quarter million employees. We'll structure it differently accordingly. Yeah.

[0:55:08] JH: Okay. Okay. That can be deployed inside an environment, or you'll support it?

[0:55:15] RC: Yeah. Yeah. The software can either be hosted by 3Forge, or it can basically be hosted by the customer itself. A lot of our use cases, they wanted to be internal inside their four walls. There's huge security concerns. By the way, that's another, I think has turned out to be a huge blessing for us about not using open source, because going through the vendor onboarding process at these large firms, I'm actually curious how most software vendors are getting through, because they literally make us now, we get this form where you fill in every open-source library that you're using, how you're supporting and monitoring that library, what checks you have in place, da, da, da, da, da, all of these things. It becomes this huge process to bring that through. We get to skip by that, because we have no open source. We have no –

[0:56:05] JH: You’re like, “Nope. Nope. Nope. Nope.” That's good. I like that. This is one thing I wanted to ask you, which is, all right, so let's say there's somebody out there at a top 10 bank who owns all the data platforms, or maybe a line of business, right? They're like, “Hey, I want to do something like this. I want to buy something like this for my company.” What should buyers of this type of – not just your product, but if you've got this problem, what are four or five things you

should be thinking of? What's the framework for understanding how to buy this? What kind of problems should I be thinking about?

[0:56:45] RC: Right. Well, I think flexibility is an important thing to focus on. That's a tough thing to do during a POC. I think, it's about asking other people in other use cases what they're interested in. I think, one of the things that we hear over and over again is one of the things they find very valuable about our platform is we actually decrease the number of vendors they need, not increase. I can say again, being on the other side of the fence when I was getting vendor products, it's very annoying having to deal with more and more vendors that solve one niche thing.

For me, I think when it comes to evaluating a piece of software is to see how many use cases can it solve? How many different ways can it be used? Really, look at what the roadmap has been. Is the product on an incline, or a decline in terms of what's the history and what's the trajectory of where that product is going? Again, these are hard things to do, because you can do a POC, and usually, you can find a very specific problem that will – product that will solve that POC. Then the problem is, ultimately, you're now constrained by what that particular implementation can do, you know what I mean? Looking for flexibility, I think, is a critical piece.

[0:58:10] JH: Yeah, I think that's right in the POCs, people forget to ask that. They only ask the question, does it do the thing it said it was going to do? That's one problem. The other problem is, can it live in your environment and meet all the other non-visible requirements, right? POCs don't shed enough light on that second topic often, unless you're really just focused on really specific about it. How long does it take to deploy something? Let's say, I have five or six sources of data. I've got two different lines of business who want dashboards and want to be able to query historical data, streaming data. For that package we described starting cold start. How long does it take on average to deploy?

[0:58:58] RC: We've had POCs done in a few days, where they've basically gotten the product implemented. Depending on the use case, could be a few days, could be a few weeks. I would say, it is safe to say that the alternative, usually when someone chooses our platform is to instead, go and build a bespoke solution. We look to cut that down by at least 95%. Whatever it

would take to build something, regularly it's about a 20th of that. If it took you a year, maybe take one or two weeks to build it on our solution. That's what we focus on.

I know that sounds like a crazy number to say we can save 95% – we can cut down 95%. What's funny when I actually look at the platform itself for us and where we spend our time, we spend a vast majority of our time on actually making tools so people can build solutions faster. The actual features and functionality for the end product, we're actually, we're more or less done years ago. We had most of that in place of what's being used today.

What we focus on is paying attention to what helps our customers develop solutions faster, pinpoint problems faster. We have use cases where our users might have a 1,000 panels or more, and literally a dashboard with up to a 1,000 panels. Once you start to get to a 1,000 panels, you have 20 or 30 different subsystems that you're hitting, that becomes a complicated thing. How do we build tools around and optimizing that problem? By focusing on that and streamlining that development process, that's how we've been able to continuously reduce and reduce and reduce the amount of time it takes for people to build a solution.

[1:00:50] JH: Okay, interesting. Well, it's been really great talking with you. I understand your strategy is like, you started with the top-hanging fruit. Now, you're expanding to the mediums of the other organizations. I know you've done some speaking in the past. What would you recommend for fans and potential buyers? Should they follow you on – they go speak anywhere, anything you want to plug right now in terms of where to get more in touch with you?

[1:01:19] RC: Yeah. I think following on LinkedIn makes sense. 3Forge LinkedIn is definitely our – that's where we do our, most of our social media notifications, things like that whenever we have new databases, adapters and new features and new releases, stuff like that. Yeah.

[1:01:34] JH: Okay, great. I'll put that in the show notes. Thanks for spending some time talking to us. You've already been very successful with your product. You've got a particular point of view that really seems to be serving the customers very, very well.

[1:01:48] RC: Okay, thank you very much. Thanks for having me.

[1:01:50] JH: Thanks.

[END]