EPISODE 996

[INTRODUCTION]

[00:00:00] JM: Robotics has changed modern agriculture. Autonomous systems are powering the tractors, cotton pickers and corn cutters that yield plants at industrial scale. John Deere is a company that has been making farm equipment for 183 years. Over that period, the planting and harvesting process has become increasingly mechanized and John Deere has been at the forefront.

Over the last few decades, software has played an increasingly important role at John Deere. Today, there is software inside the vehicles. The tractors, the cotton picking machines, and these vehicles can operate autonomously. They collect large amounts of data and they're supported by a large system of cloud services. The teams within John Deere who create the software have an elaborate testing workflow that allows them to deploy the software to the vehicles and drive them in the field.

Ryan Bergman is a software engineer at John Deere and he joins the show to talk about the software engineering, management and DevOps practices within the company.

We have partnered with SafeGraph for the SafeGraph Data Hackathon Challenge. We're giving away $4,000 in cash prizes as well as Software Engineering Daily and SafeGraph swag. SafeGraph is a geospatial data company which curates a dataset of more than 6 million points of interest. SafeGraph provides a high-volume of location data and you can build apps and data science projects with that data. If you've been looking for a creative opportunity to explore large datasets with the potential to win $4,000 in cash prizes, this is a great opportunity.

The hackathon is hosted on FindCollabs and you can enter by going to findcollabs.com and signing up. You can create any kind of creative project that uses location data and it's a great opportunity to try out SafeGraph's data as a service platform. To check it out, go to findcollabs.com.

[SPONSOR MESSAGE]

[00:02:05] JM: I love software architecture. Software architecture is the high-level perspective of how to build software systems. Much of Software Engineering Daily is about software architecture, and if you're interested in software architecture, there's no better place to go to discuss and learn about software architecture than the O'Reilly Software Architecture Conference, which is coming to New York February 23rd through 26th of 2020.

If you are interested in software architecture, you can go to oreillysacon.com/sedaily. That link is in the show notes, and you can get 20% off your ticket to the software architecture conference. The O'Reilly Software Architecture Conference is a great place to learn about the high-level perspectives and the implementation details of microservices, cloud computing, serverless and also systems like machine learning and analytics.

If you've been listening to Software Engineering Daily for a while, you know that these systems are hard to build and they take engineering details at both the high-level and at the low-level. Whether you're a seasoned architect or an engineer that is just curious about software architecture and maybe you want to become a software architect, you can check out the O'Reilly Software Architecture Conference at oreillysacon.com/sedaily. Use the discount code SE20 and get 20% off your ticket.

There are lots of reasons to go to the software architecture conference. There's networking opportunities. There are plenty of talks and training opportunities, and you can get 20% off by going to oreillysacon.com/sedaily and entering discount code SE20. I've been going to O'Reilly conferences for years and I don't see myself stopping anytime soon, because they're just a great way to learn and meet people. So check it out, and thanks to O'Reilly for being a longtime sponsor of Software Engineering Daily.

[INTERVIEW]

[00:04:23] JM: Ryan Bergman, welcome to Software Engineering Daily.

[00:04:25] RB: Hi. Great to be here.

[00:04:26] JM: You work at John Deere, and we're going to discuss a lot about software, and hardware, and agriculture. I'd like to just start by getting your description for the software and the hardware that runs modern agriculture. Give me an overview of the technology.

[00:04:44] RB: Sure. There's a large piece of agricultural equipment. In the kind of stuff I work on, we're talking about really big combines, tractors with planters, sprayers. It's kind of amazing. It's really a series of a whole bunch of little computers on a lot of different things. You could think of like the tractor is a bunch of microservices and all of the different pieces of hardware often have their own embedded controllers, like even down to individual spray nozzles on a sprayer and they work together, they communicate together and make up a total system. The sheer number of computer systems and moving on these things is pretty amazing.

[00:05:30] JM: What are some examples of devices that John Deere builds?

[00:05:35] RB: I'll kind of layout where I'm coming from with this. I work for John Deere intelligence solutions group. We are a kind of a division or business unit of John Deere that's a little bit independent from, say, corporate or from any of the particular factories.

Generally, what we call a platform is a particular kind of equipment. So, harvest works, makes combines and they kind of oversee their own thing as far as the combine goes. The tractors are made in Waterloo and they have their own kind of thing. What intelligence solutions group does is we build a lot of the more advanced computer systems that kind of help tie all of these things together. We make the display that goes on the tractor or any of it, the equipment.

There's generally an embedded display. There's also what we call an MTG, which is basically a glorified, ruggedized cellphone. All the large equipment comes prebuilt with a telematics device like that. We oversee the satellite receivers. All of the equipment is GPS-driven, and a lot of times the equipment will have more than one satellite receiver, and then we also have a large network of base stations and farmers kind of have their own base stations. We oversee that too.

All of that is – Generally, those parts of it are like embedded C++ kinds of applications, and those will collect all the data. They're used for configuring the equipment. Then we take all of that data and through that telematics device, we suck all of that data up into the cloud and we

process it and a lot of the stuff that I personally work on is all that data that's up in the cloud and then the farmers can use to get reports and maps about their different kinds of operations. They can share that data with, say, an agronomies they're working in and the agronomies can be able to look at that data. Make prescriptions for spring or fertilizing and they can send that back and it automatically goes back on to the equipment and then the equipment can automatically configure itself for those kind of prescriptions. Say of like a spring operation, the agronomies might say, "Hey, you need – Let's do this amount of chemical in this part of the field and this amount of chemical in this other part of the field." That goes under the sprayer. The sprayer then knows, "I need to spray exactly this amount of chemical in this piece of land, and this amount in this piece of land." You can drive the sprayer in figure 8s across the field if you want. It's only ever going to apply exactly what it needs to apply. That way we can severely reduce the amount of chemical use that's applied, because farmers don't want to overpay. Consumers don't want their food over-sprayed. That's kind of like the whole circle.

Then once you get to harvest, we take that data. We bring that up into the cloud and then we can look at that data that they're planting and they're sprinting and they harvest and we can see how those different inputs impacted the bushels per acre that they got in a particular field and a particular area of a particular field.

[00:08:46] JM: But the data as far as the yields, the crop yields at the end, that's data that somebody has to put in manually at the end, right? Because that's not something that is necessarily going to be gathered by the farming equipment.

[00:08:59] RB: Yeah. Yeah, it is. I mean, at the end of the day, you're going to get paid for what goes into the elevator when you go and you weigh it at the elevator. But what we know about the, say, corn, or soybeans, or wheat, or whatever it is, the combine can calculate exactly pretty really close to what you actually have. We know what they're going to have before they even get to the elevator. We then can calculate our sensors that are in those combines and they're looking at the grain. It's going through the chute and we can use that data and we can calculate the moisture. We even know, like when you get to an elevator, you're going to get paid for not only the grain, but also as a factor of that is how much moisture is in it, because if the elevator has to dry it, then there are charges for that because that takes propane gas or some other

source to dry the grains. We know the moisture content. We know how much grain you've got. We could tell you what you're going to get at the elevator before you get there.

[00:10:05] JM: Amazing. I'm imagining these different kinds of machines that drive around the fields. You've got combines and cotton pickers and all these different gigantic machines that John Deere builds. In each of these things, there is some data that gets gathered. There're some calculations that takes place locally on these big pieces of machinery. Then there's some shuttling of data to the cloud. Is the data gathering and the data sending to the cloud, is that heavily separated or can you just stream data to the cloud? Do you have a consistent, reliable network connection to be able to send data to the cloud?

[00:10:54] RB: Well, the answer to that is it depends. Ideally, we will stream it and that's kind of our default behavior, is to stream. We actually stream kind of in 30-second chunks. You don't necessarily like a constant stream. We're sampling that data. So all of these, the planting data and the harvesting data gets – We're not transmitting, say, the GPS coordinates of every single seed we plant, although we could do that. That would just be way too much data and it's not terribly useful. We do a sampling between 1 and 5 hertz on all of that data. It gets kind of little batched up and then sent in. As long as we have a connection, we can do that.

That's not always the case. I live in De Moines, Iowa. I can drive about 30 miles from De Moines and get you to a place where there's no cell coverage, and that's despite the fact that you go to AT&T or Verizon or whatever and you look at their map and it looks like, "Oh! We cover all of America," and there's only like one or two little pockets of nothing, and that's not quite right. There's lots of rural America that has really poor coverage or you do get there and it's only like the weakest of edge. It's pretty bad.

We generally design all the telematics stuff so that it doesn't require a real solid connection. We'll transmit over to 10 cans with some string if we could. If we can't get a connection, it'll will just kind of bundle it up and wait until the equipment gets to a place where it does have a better connection and then it can send it in.

Still, even there, especially at west and rural areas where it's just not going to happen at all, and if that happens, the farmer can plug a USB stick into the tractor. It will pop all of the data off on to that. He could take it back to his computer at home and plug it in and will upload it from a desktop computer or from their cellphone.

[00:12:55] JM: It's not an overwhelming amount of data such that the connectivity, intermittent connectivity across different customers is going to get bottlenecked basically while you're not connected to the internet and you would have to down sample or have to lose data or have to do rollups or something. It's just not so much data that that happens?

[00:13:19] RB: No. Not really. We do a pretty good job. The format that we send it in is pretty compressed. It's nowhere near like watching 4K on Netflix. We're not at that level. Once we – As far as the transmitting of it goes. Now once we expand it, that data gets pretty big and we've got – I think the last time I checked, like 25 petabytes of total agronomic data, which is not tiny. I still don't know if I could call it big data though. I think we're still only a moderately-sized pond.

[00:13:56] JM: Right, like in the scheme of things.

[00:14:00] RB: Right. Compared to some other people are dealing with.

[00:14:03] JM: Yeah. What can you expect out of the equipment operators in terms of technical expertise? Do they need to have significant understanding of anything related to software engineering?

[00:14:18] RB: Well, no. There's a real shortage of farm labor in rural America, which is part of the reason why John Deere and other people in agriculture have been working more and more towards autonomous vehicles and automating all of these, is that there's just not a lot of people out here do this labor and the expertise of the people who are there, we can't count on them having any kind of computer science degree.

Our mantra is someone should be able to just jump in the cab, turn the key and go and the equipment is going to figure out everything that it needs to do. There's probably going to be someone who knows a lot about not necessarily computers. They don't need to know about

that. But just of how to configure some of these equipment, because it is complicated and particularly if you're configuring it to be able to do the self-driving operations, you got to get all of the – Everything kind of configured the right way with the satellite receivers. A lot of people rely on their dealers to help them do that, and we do a lot to help enable those dealers and other service providers to be able to help the farmers in doing that, whether it's being able to send alerts about oil pressure, to the dealers or other things so that they can let a farmer know that he's going to have a problem with a piece of equipment and they should get it fixed earlier or if it's – Like I was describing before with the agronomies and then being able to do those advance agronomic operations and being able to configure the sprayers for what they need to do.

[00:16:03] JM: You have been with the company for more than 8 years, and the period of the last 8 years I would say is this idea of "a digital transformation" has really built a lot of steam. Now, there're a lot of companies that have been digitally transforming for a longer period than 8 years. But certainly over the last 8 years, the cloud has really caught on as a software development paradigm. So you've seen engineering, I'm sure, change at a pretty rapid pace, because the cloud has been such a transformative set of technologies as well as like open source. There's been so much happening in open source for the last 8 years. How has engineering within John Deere changed over that 8-year period that you've been with the company?

[00:16:51] RB: All right. It goes back further than that from when I joined a little over 8 years ago. ISG started I think around 2000, and really it started off with the satellite.

[00:17:06] JM: That's the intelligence systems group.

[00:17:08] RB: Yeah, the Intelligence Solutions Group. Yeah. It really started around 2000 with the satellite navigation and the automatic steering. The original first version of those things, we literally mechanically turned the steering wheel, right? You could retrofit it to like older pieces of equipment too, which is pretty fun.

When I join, John Deere ISG was kind of undergoing a change at that time where they were kind of moving to more agile and XP methodologies. I worked at another place in town that was kind of known around town for – We did a lot of that kind of stuff. We do pair programming and

we did all TDD, and that company was bought by another company and it kind of collapsed at that point, like all the engineers quit, and I think bout 15 of us wound up at John Deere in one capacity or another, and really a lot of things changed around that time.

If you were to come and see – We just moved into a new building, which is pretty cool. It doesn't look a lot different than a lot of the shops you see in Silicon Valley or elsewhere and we still practice a lot of those practices, right? We're doing pair programming. We're doing test-driven development. We do a lot of testing at a lot of different layers. Even, really, it's so much more important with the kind of stuff that we're doing, because it's a lot harder to get an update if there's a bug that happens to impact displays or something on the equipment. That's a lot more expensive to fix than just something that's in a web app, right?

Also, taken into account that or equipment, it's big. It's dangerous. It could hurt you, and we have to take those kinds of considerations into account and make sure everything's pretty solid before it goes out and gets in the hands of customers.

[SPONSOR MESSAGE]

[00:19:20] JM: Today's episode is sponsored by Datadog, a modern, full-stack monitoring platform for cloud infrastructure, applications, logs and metrics all in one place. From their recent report on serverless adaption and trends, Datadog found half of their customer base using EC2s have now adapted AWS Lambda. They've examined real-world serverless usage by thousands of companies running millions of distinct serverless functions and found half of Lambda implications run for less than 800 milliseconds.

You can easily monitor all your serverless functions in one place and generate serverless metrics straight from Datadog. Check it our yourself by signing up for a free 14-day trial and get a free t-shirt at softwareengineeringdaily.com/datadogtshirt. That's softwareengineeringdaily.com/datadogtshirt for your free t-shirt and 14-day trial from Datadog. Go to softwareengineeringdaily.com/datadogtshirt.

Thank you, Datadog.

[INTERVIEW CONTINUED]

[00:20:32] JM: The workflow is of course much more complicated than a web app. If I'm building a simple web app for hosting a podcast, for example, I can do all of the testing that I need to do on my laptop, basically, or some combination of the laptop and remote cloud resources. But you are testing software that needs to be deployed to a tractor or software that needs to be on the cloud, but supporting a tractor in the field.

I imagine you can do some of the testing that you need to do with simulation. I'm sure there are some kinds of your systems that are isolated enough that you don't really necessarily have to do lots of unit testing all the time in hardware in a real-world capacity. But I do imagine the overall workflow needs to account for the kinds of integration and real-world software-hardware interface challenges that are potentially going to occur once that software is out in the real-world. Tell me more about the testing and development pipeline and how you minimize the amount of errors that are going to make it out into the real-world.

[00:21:56] RB: Sure. Yeah. We have kind of all range of different kinds of applications. We do have applications that are really just purely a web application and really only deal with state and things that are within our direct control and could be tested in a normal way with normal developer TDD kind of environments. A lot of the stuff in my area kind of falls into that. We do mobile app development. It's kind of similar. You can mock out API requests.

As far as something goes on the equipment, that's where it gets a lot more complicated. When I first came to Deere, the first project that I worked on is called remote display access. This allows anyone from a computer anywhere to be able to remote into the display. It's basically doing good old VNC technology that's been around since the 80s. But we're doing it to a tractor over a cellular connection that might be out in the middle of nowhere.

That's really valuable for like specially the dealers and the other people who are helping the farmer configure their equipment and they can – If they have to do a service call and drive a truck out, sometimes it could be in like West Texas. It could be an hour and a half, two-hour drive before some tech from the dealership gets to the farmer. If they can just VNC into the

display and they could say, “Oh, yeah. Just click here. Click here. Change that to a two, and now you’re good.”

That’s like the whole thing, right? It impacts the equipment. It impacts the web, and it’s really difficult to test kind of all the way end-to-end really well. At least it’s not fast, right? But we mitigate that in a couple of different ways. One, we have kind of small scale equipment simulators which we can plug-in to displays. If you walk around our office, you’ll see people have a display and they’ll have a steering wheel just like strapped on to the side of it and you could do a little bit of simulation there, or at least you can interact with the display for most of the things.

When we’re talking about simulating things like sensors coming in or we want to do more advanced stuff with the entire cab, we have entire simulators where we basically have a lab and we have the cabs of the equipment, because the cabs are fairly interchangeable. It’s basically the same cab that goes or the same command center if you will that goes on a combine, that goes on a tractor, that goes on a sprayer.

We have these. Then they’re hooked up to what are essentially look like gigantic Nintendo Entertainment System cartridges that are about like – I don’t know. They’re bigger. They’re like a size of a dresser and they contain within them all of the electronic equipment for that particular piece of equipment, so for combine for example. They would have all the sensors and electronic equipment. They’re not as wide as a dresser, but they’re pretty thin. They look like a cartridge.

Then we have a bus that kind of travels below them. So rather than you taking the cartridge and then plugging it into it, the plug kind of moves around the bottom and you decide which one you want to hook it up too and it hooks up to that. That allows a lot more advance simulations, and we’ll run like on the display builds, we’ll often run it through a whole host of different automated tests with the actual hardware.

Then our ultimate thing is we have an entire test farm. It’s about – I don’t know, 15, 20 miles out from our office and it’s pretty nice, because if you happen to be working on something that impacts the tractor and it’s really a nice day in the spring or the fall or something like that, you could be like, “Oh! I need to test it out at the farm.”

We have all the equipment. We have a cotton picker there. It's the only place you'll see a cotton picker driving around Iowa, and you can go out there and we can run the builds on actual equipment. It's a pretty nice facility.

[00:26:05] JM: That's cool. Quick check on autonomy. The idea of a self-driving tractor or self-driving cotton picker, that's a reality today or to what extent is –

[00:26:21] RB: Yeah. We've been doing it for years. Yeah, it will drive up and down – Originally the first version was that it would stay on an exact line. You can imagine with like rows of corn, it could align itself right up with those rows that it knew it planted. We knew we planted these corn rows, these GPS coordinates. So we have a line there, and the equipment can align itself up with that row and it can drive down.

The first version of it, you had to turn it around, but we've got to the point where essentially the equipment could just go into a field and drive back and forth and do everything itself. We do require that a human being still sit in it. It is dangerous equipment and we need that human override in case something were to happen. We don't want our combine like going off and hurting someone or livestock or anything like that.

But the equipment is essentially driving itself at that point and we're getting more and more advanced with that as we go. The equipment also has the ability to communicate with other pieces of equipment in the field. Say you have a tractor come in with a grain cart to offload a combine, the tractor with the grain cart communicates with the combine. The two pieces of equipment figure out exactly how that tractor is going to come up to the side of the combine. The combine can offload its grain and automatically figure out how to fill up that bin.

This is a really stressful thing if you're two human beings trying to do this. Try to drive a really, really big piece of equipment and have a chute off to the side offloading grain and trying not to hit each other can be really, really stressful. The equipment basically does all that part itself and they don't have to worry about that. Then once the grain cart if filled up, the tractor kind of pulls off to a point where the human can safely take back over the driving of that equipment and it takes it off to wherever it needs to take it, a silo, or a semi-truck, or something like that.

[00:28:34] JM: When you have two machines communicating with each other, are they routing their traffic through the cloud or can they talk over Bluetooth or something?

[00:28:46] RB: They're talking to each other over – I believe it's a shortwave radio.

[00:28:50] JM: Shortwave radio. Okay. That would make more sense. Let's talk a little bit more about the software infrastructure side of things. The applications that are running on the tractors themselves, are those mostly C++ or do you have a wide variety of application types that are actually running inside the tractors?

[00:29:13] RB: It's almost entirely C++. Yeah. Especially as you get – Or just C. If we're talking about like an embedded controller on a transmission or something like that, that most likely probably just C, maybe it's C++. It's certainly natively built. The displays are all C++.

We have looked into augmenting that with not everything necessarily like on the display would need to be C++. It is a real-time operating system. We really need that when you click this button that this physical thing happens. We need that aspect to it. So there's performance implications, and a lot of those – The people we have that are really experience with this are experienced C++ developers. It's kind of a natural language, and C++ honestly isn't as bad as it used to be if you're using like C++ 11, I think is the newer version.

We have looked at starting to do some similar stuff with like Rust, or Go and there's no reason why a lot of the applications on the display couldn't be written in something like Python or something like that. Not everything on there needs real-time critical applications. There's like a radio. The radio could probably be in whatever, right? You don't want too many runtimes and dependencies there. But I think Rust is really the most interesting one of the newer languages. I know we have several different groups looking very closely at Rust.

[00:30:50] JM: Because of the memory safety or what makes it attractive to you?

[00:30:55] RB: Yeah. I mean, what's attractive to me is, yeah, the memory safety. The fact that it's just a little bit more modern and I just find it a fascinating language in general as far as – I

don't do a lot of that embedded programming myself, but if I ever get over into that space more, I would definitely want to look at that. Go would be fine too, but it's not as hip and new as Rust is.

[00:31:22] JM: With the C++ stuff, I wonder if this is going to be like what we have with like COBOL and Fortran where there's now a huge shortage of COBOL and Fortran developers. C++, isn't that like the really good C++ developers? They're all aging out. I don't think anybody – Like millennials are not learning C++.

[00:31:47] RB: No. It's hard. What's interesting though is I think a lot of our developers that are on the display, there's quite a few that are just mechanical engineers, right? They didn't go in to necessarily computer science. They went into to become engineers, agriculture engineers going to like Iowa State or University of Illinois Champaign, we have kind of a lab there, with the intention of doing real physical engineering anymore. You can't do real physical engineering without software. So we tend to like get quite a few of our – If you walk through our embedded group, there is a lot of younger developers. There're people in their 20s and 30s there. It's not a bunch of older folks. We have older folks too. But generally I think, yeah, we have a bit of a problem recruiting and we're always looking for people, but we'll train you too. C++ isn't as bad as people think it is, and it's as bad as COBOL, I don't think. Though I've never done COBOL. I mostly from like the Java C# generation. I feel like we have a hard time getting people to do Java.

[00:33:04] JM: I mean, that's the thing that I learned the most. I took the most college classes in Java and I really got used to it and got acclimated to that level, the level of safety and abstraction and really have no desire to program in Java. I don't have really much desire to program in any language, but that's why became a podcaster. But C++, I do remember being pretty tough when I was working with it, but I'm sure it's just a matter of practice.

As far as the server infrastructure, you're handling all the data. You're handling all the data. You're handling the backend server infrastructure. You're handing all the databases, cueing systems. I mean, all of this stuff is abstracted away from the farmer and the farming team, the agriculture team, because again, like you said, these are not like computer science user. They're technical in a certain sense, because the agriculture, like modern agriculture, is a very

technical business, but it's not technical in the sense that you're going to be like writing – You're not going to go home and want to export your data and write scripts, or maybe some more sophisticated agriculture people are, but the average agriculture person, they just want a nice UI, like just a simple UI. Nothing more complicated than like the Facebook interface or the – I don't know, QuickBooks interface. Something like that. You're really handling all of the backend data infrastructure. You're ingesting all of these data and then you're doing something with it and have all these services around it.

Tell me a little bit about the server infrastructure. Give me a high-level and then we can dive into some different components and the management of it. I guess starting with the ingest point, perhaps.

[00:34:45] RB: Right. Yeah, that's exactly what – We like basically a pipeline and we call it IMET, which ingestion, model, enhance and transform. The ingestion step basically takes – Is the gathering point of all that data. That data can come in streaming from the equipment. It might get uploaded as a single solid file. It might be coming in from competitor equipment. We'll read some of our competitors files as well and you can get all of that in. So the ingestion is just kind of like the raw, we're going to take this and put it into basically a very raw common format for the next step, which is model, which is where we take that raw data and we model it into what we want kind of our final state to be at least as far as kind of a scaffolding goes, right?

Not all that data is necessarily going to populate all the pieces of it, but it will all be in the right shape. Then we go to the enhance section, which is where we might take – Do calculations to creates numbers that don't exist in the raw data. The raw data will not tell you how many miles per hour the equipment was operating at a particular point in time in its – Kind of in its space time continuum, right? But it will tell you what its heading was and we can deduce between the sample rate of the data we're getting what its velocity was. At that point we can say, "Right here, this piece of equipment was operating at 15 mph." We'll calculate all kinds of stuff there, and that's kind of the simplest example.

Then we finally take it and we go to the final step, which is to take that data and make meaningful pieces of information out of it. We basically layer all of our data. It exists, the

agronomic data at least, in a kind of scaffolding of the world and we follow – We use Google Maps, which is pretty obvious if you look at our stuff. Although we can use other GIS stuff.

The way that Google Maps works if you ever look at it is like the entire world is one, and then they break the entire world into four squares. Then that's kind of level II. Then if you take one of those squares and break into four squares, that's level III, right? You go all the way down and you get to – I don't know. Generally, I think the lowest level that they support with the imaging is like 12, but in cities, you can get down to like 15 or something like that. We kind a model our data in the same way and then we can have a satellite view and we can take all your agronomic and yield data and we can layer that on top of it. Then we can layer whether data, or soil data from the USGS or something like that, and you can stack all this data on top of it and then you can make – You can do some scientific analysis and you can be able to look at that and say, "Well, how did – I bought this kind of seed this year from this company. How well did that seed perform versus this other seed that I used the year before when we had about the same whether, or how does this seed work in this kind of soil versus this other kind of soil?"

Obviously, seed companies are really interested in this same kind of data. A lot of times they'll have relationships with the farmers where they get to look at that data as well, and usually I hope the farmer gets something out of that. The key is the farmer always has the ability to say who gets to see their data and what kind of relationships they have with it. But technically from the server side if we're talking about like the stacks, that's all done through kind of streaming queues. Most of it is written in Scala and the data just kind of flows from one queue to the next as it goes through that process.

[00:39:01] JM: You're just using like native Java queues. Do you use any distributed queuing infrastructure like a Kafka or Kinesis or something?

[00:39:11] RB: It's all Kafka, or at least Kafka-eque.

[00:39:15] JM: What does that mean? I mean, I get the pun.

[00:39:17] RB: Well, it's not necessarily always Kafka.

[00:39:20] JM: Okay. Not always Kafka. It's not always Kafka itself, but it's something that is almost exactly like Kafka.

[00:39:28] JM: Like a RabbitMQ? What does that even mean?

[00:39:32] RB: Well, like Amazon has their own native implementation basically of Kafka.

[00:39:39] JM: You mean Kinesis?

[00:39:42] RB: Yeah, Kinesis.

[00:39:42] JM: Got it.

[00:39:44] RB: We use a lot of Flink too. That's in there.

[00:39:46] JM: Do you remember the evaluation criteria of Flink, versus Apache Beam, versus Spark, versus Storm, versus whatever? I mean, there's a bazillion of these streaming frameworks.

[00:39:57] RB: Yeah, and we use a lot of them. We use Spark too in different places. A lot of it is we're very bottom-up driven company.

[00:40:07] JM: It sounds like it.

[00:40:07] RB: The different teams have a lot of – The different teams have a lot of autonomy to be able to figure out what works best for them. We will look at some of this stuff particularly if it's open source though and there's not a big upfront cost to engaging in something. We're more than willing to let somebody let a team experiment and try out.

We've changed over time the different exact technologies of these different areas we're working with. We'll use whatever the team decides works best for them. We encourage a lot of experimentation. We do pretty regular hackathons, which are pretty fun. But usually that's a time where some the teams will try out some kind of more far out ideas for changing stuff, and we've

had a lot of success with that as far as people driving down costs or doing things faster or doing things easier.

[SPONSOR MESSAGE]

[00:41:11] JM:

[INTERVIEW CONTINUED]

[00:41:12] JM: Better.com is a software startup with the goal of reinventing the mortgage industry. Mortgages are a $13 trillion industry that still operates as if the Internet doesn't exist, and better.com is looking for engineers to join the team and build a better mortgage experience.

The engineers at better.com are attacking this industry by bringing a startup approach into an industry filled with legacy incumbents. Better.com automates the very complex process of getting a mortgage by bringing it online and removing the traditional commission structure, which means that consumers can get a mortgage faster, easier and end up paying substantially less.

Better.com has a modern software stack consisting of Node.JS, Python, React, Typescript, Kubernetes and AWS. They iterate quickly and ship code to production 50 to 100 times every day. Better.com is one of the fastest growing startups in New York and has just announced a series C that brings the total funding to $254 million.

If you're interested in joining a growing team, check out better.com/sedaily. Better.com is a fast-growing startup in a gigantic industry. They're looking for full stack engineers, frontend engineers, data scientists. They're looking for great engineers to join their quickly growing team and. For information, you can visit better.com/sedaily.

[INTERVIEW CONTINUED]

[00:42:51] JM: Now, what you're talking about there with the bottoms-up developer freedom, that's such an interesting question how much freedom developers have, because you look at a

company like Google and they have this idea of the blessed languages where everything at Google basically has to be written in either Java, or Python, or C++, or maybe Go in some cases. But it's really given how big Google is, they have a pretty narrow selection of languages, and obviously that's a constraint in some regards because you have some developers to come in. They want to work with Rust. They wanted to work with Scala, and the advantage of having bottoms-up freedom is you give developers the freedom to work with whatever language they want to. The downside is when that developer leaves – Like I worked at a company one time where they had a bunch of Scala infrastructure and then all the developers who love Scala had left the company. So they had like – It's like really annoying problem where like how do you support the legacy Scala? There is this risk to having that bottoms-up developer freedom.

[00:44:01] RB: Yeah. We went into Scala very intentionally. Like we thought – Kind of when we started that project was when Scala was kind of hot. Everyone was talking about why Akka and stuff like that.

[00:44:13] JM: Oh, yeah. Six years ago. Something like that.

[00:44:15] RB: It was about six years ago. Right. That was quite intentional. Before that, we were a very much a Java environment and then we had some desktop applications. They're like written in C#. So we had some C# in the office. I don't really get – I kind of sit – I've operated as an architect, and I haven't really had a big problem with like weird rogue languages. Nobody's come in and been like, "Let's rewrite half of this in Haskell."

[00:44:46] JM: Don't jinx it.

[00:44:47] RB: Right. I love Haskell, but I would never do Haskell for someone's corporation. I would never do that to them. But we essentially have kind of some blessed languages, but I don't know that we like have ever written it down necessarily. But it's generally JVM stuff. When I say JVM, I really mean like and Scala. Although quite frankly, if anybody came in and said, "We'd like to do this and Kotlin." I'd probably give it a thumbs up and go like, "All right." C++ is on the embedded side. That's pretty nonnegotiable, though the displays are only going to run. You want to get a runtime on there, it's a bigger deal, right? On the cloud, it's kind of a lot looser.

[00:45:31] JM: Now, there's the blessed languages question. Then for a company like you, there is an equivalent question in terms of cloud technologies, like can you be – As a developer, can I go on DigitalOcean? Can I go on Google Cloud? Can I go on AWS? Do you have constraints over cloud resources?

[00:45:57] RB: Yeah. In that case, we have a lot, because as a corporation, we have to have service-level agreements with anybody we go into. Adding in a new cloud provider is a much bigger deal than picking a language and we would have to go through all of the normal procurement kind of thing. Anyone who's worked on a big corporation, it's difficult to get the kind of the spigot open for a vendor has. Particularly, a company like John Deere that has such a history in manufacturing, we're very serious about our suppliers and how that's going to flow.

Yeah, developers can't just like decide they're going to runoff and put something on some new cloud stuff. But we generally have agreements with the major cloud providers and we'll use whichever ones we see fit to use for a particular technology.

[00:46:59] JM: Are you mostly AWS or do you use some Google Cloud? Some Microsoft? Can't talk about it?

[00:47:05] RB: Yeah. I don't know that I'm supposed to talk about it. Anyone who does [inaudible 00:47:09] can figure out [inaudible 00:47:11] is hosted.

[00:47:11] JM: Fair enough. All right.

[00:47:14] RB: But there are a lot of visits to Seattle, the Seattle area.

[00:47:19] JM: All right. Yeah. That really narrows it down. You mean like Google Cloud is based in Seattle. Anyway, the question of legacy technology. John Deere has been around for 183 years and I don't know how – Like when you started getting into software, but I'm sure it was long enough ago to have a lot of legacy software infrastructure throughout the company. What's the strategy for maintaining legacy software overtime? Do you have a sustainable legacy strategy in place?

[00:47:54] RB: Yeah. I mean, certainly, there're parts of the company. I mean, ISG is relatively young yet, right? We're only like 20-years-old. There are definitely – Within that time though, we've produced multiple displays and the earliest versions of them are no longer supported. We won't do any kind updates for them at all. That doesn't mean you can't use them, but we only support – I'm not even sure if we're doing patches anymore for the one that was the prominent one when I joined. But we'll support those for a certain amount of time as far as critical bug fixes, something that's related to safety or something like that.

The rest of the company, as a corporation, you go back to like the factories and assembly lines and stuff, that's where you get into some really old stuff. I don't really have a great insight in what their strategy is for it, but I see the churn. The challenge with any kind of thing like that is that particularly as you get the older stuff, not just having people to maintain it, but equipment to run it. John Deere isn't – It's usually inclined to run old IBM mainframes that IBMs not supporting, right? I've seen them retire stuff like that and projects of having to just kind of completely rewrite stuff that was an old COBOL or something like that and then rewrite that into Java or something like that and being able to run it on cloud infrastructure rather than dedicated hardware like an IBM mainframe or something like that. There's definitely just always some kind of project going on at Deere and if we think of any large corporation of having to recycle and recreate some of these older systems.

[00:49:54] JM: You've done some talks about the problems of modern DevOps workflows. What are the most acute problems that you see with modern tooling and DevOps workflows?

[00:50:07] RB: My main criticism of the entire thing is I think it's just entirely too complicated. I did a keynote at DevOps Days De Moines, which is on YouTube, if anyone wants to see it. But where I kind of layout, I give like a flowchart of what it means to develop a web location in 2019 and starting from the top with the icons for all of the different things. You write your requirements in Jira and then you go and you write them in this language and you check them into GitHub and then GitHub kicks off Jenkins, and then Jenkins does this, and then that does that, and then it goes to Terraform. Then Terraform builds out like this entire infrastructure.

Before you know it, there's like literally 25 different icons up there all representing something different, right? Something that I as a developer have to know about. I have to know how these

things like hook together. I might have to know up to like half a dozen different languages. Some of them like Terraform is kind of a one shot thing. There's nobody else quite uses that language.

Then the result of all of that is some kind of microservice mesh of systems that all talk to each other. For any developer to have to jam all of that into our heads all the time and understand where anything might fail, it's just too much. It's overwhelming like how much we have to – The cognitive load, compared to when I first started programming as a kid on a Texas Instruments, TI-99/4A. You turn it on. You're given a basic prompt. You program something. You run it. You have no external dependencies, and you turn off the computer and then the program is gone. That's kind of the simplest, oh, back in the day kind of thing, but it's true. We've just kept layering more and more and more and more things on to all of these things. At some point, I think the pendulum has to kind of swing the other way. I think we've already seen it like with the microservice thing where you don't go to conferences anymore and hear people talk about how great microservice architecture is and how you should like split everything up. Now you're starting to see talks where people are like, "You know what's great? You can make a microservice run a lot faster by putting all of its processes in a single JVM."

[00:52:47] JM: I think GitLab is the epitome of this, right? The GitLab model of consolidated DevOps, that's kind of – Now perhaps we're really nearing the tip of the pendulum swing to the monolith side.

[00:53:05] RB: Right, and GitHub too. I mean, if you look at what they're doing, they're basically eating up everything around them with all of their security systems and security alerts, and now they have actions. Where before, if you're using GitHub, you would also have to have your Jenkins and your build servers and like Black Duck or something for security or audit or something like that, and now like they're basically building it all into GitHub, and I'm not sure GitLab. I'm not as familiar with their offering right now, but doing the same thing.

Also, with Amazon, and Microsoft, and manage services, is also hoping to reduce that. If I can just declare A Dynamo table and I don't have to manage a database server, that's a big win for me as a developer. The whole serverless infrastructure I think it's kind of the way things are going to go more and more and more.

It's kind of interesting, because in my talk, I talk about kind of the history of DevOps and how we – All the way from the beginning. It's almost like we're coming back to mainframes, except that now like Amazon and Microsoft and Google are our mainframes, right? They still have limited resources. It's just I'm not aware of them. It's Amazon's problem to figure out how many concurrent lambdas can I run, or whatever.

[00:54:26] JM: Your limited resource is now pure money. It's basically like they've taking care of the scalability of the compute infrastructure. Now it's just like, "Okay. All right. I've got 80 bucks lying around. Time to buy a like some more lambda functions. It's time to buy some more Red Shift processing or something." The only limiting reagent is how much capital you have.

[00:54:52] RB: Right. There are hard limits.

[00:54:54] JM: But it still feels like a single – It feels like a big single mainframe computer though. You're like, "My processing is broken up and do these lambda functions. My databases, they're all distributed across different machines," but it can feel like one consolidated supercomputer.

[00:55:10] RB: Right. Exactly. Yeah. They do have physical limits. I've actually run into them, which is pretty fun. Apparently, you can't have infinite lambdas. Yeah, there is definitely a maximum number of lambdas.

[00:55:25] JM: What were you trying to spin up infinite lambdas for?

[00:55:28] RB: We've kind of gone back and forth with serverless versus dedicated servers, and then also just like size of accounts and stuff like that. You'd be surprised at how quickly you can run into like account limits. Now usually you can go and ask Amazon or Microsoft or whoever, "Hey, can you increase the account limit for the number of –" They definitely have limits to the number of lambdas you can run, the number of RDS instances you can have in a particular account and that kind of stuff. Most people don't hit them, particularly if you're in a startup or experimental phase. But if you're really running a lot of infrastructure, a lot of compute infrastructure, you can run into them pretty quickly.

[00:56:06] JM: All right. Well, as we begin to wrap up, maybe you could just tell me a little bit – I mean, I'm sure we could have gone a lot deeper into the DevOps crisis. But as far as I can tell, there's not really – I mean, you've identified a problem that I agree with you on. I don't think there's like a one quick fix to solving this DevOps crisis of too much infrastructure and too much complication.

But zooming back out to the high-level, what do you foresee in the future of farming infrastructure? Tell me about what we're going to see in the next 10 years.

[00:56:37] RB: This isn't me speaking as like a representative of John Deere, but just as someone who's kind of in the agriculture equipment world and I'm looking out and I got to farm shows. Over the next 10 to 20 years, the way I see things personally going is it's going to get more autonomous and it's going to get smaller. We've got to the point where the equipment is huge. The real challenge is we need to – We're going to have even more people on this planet and we need to be able to feed them, and that's not necessarily going to mean that we need to just like jack up the output of Iowa and Illinois as far as corn and soybeans go. We need that too, but we really need the rest of the world to be able to meet that output. Countries need to be able to feed their own people. They can't all be dependent upon Iowa to just output massive amounts of corn, right?

Some of that big equipment doesn't necessarily make sense in those smaller countries with a whole variety of different topography and everything. Also from our standpoint, having a giant single piece of equipment for a farmer, that's a single point of failure. Just like we want to break up our monoliths, it might make sense for that equipment to get small. If it's autonomous and if it's robots, they could get quite small.

I know personally of at least three or four different companies within Iowa, little startups included running out people's barns where people are trying to make like dog or horse-size robots for farm equipment to go out and do planting, or harvesting, or spraying. We're working on similar things. We have a company called Blue River out of Sunnyvale where we're doing and machine learning spraying. We're having it be able to go through the field and look right out pulled behind a tractor, but it can look at the plants and say, "Hey, that's a weed. Spray that. That's a corn plant. Don't spray that." It's going to get more. We're hoping we can reduce eventually herbicide

use by like 90%. It's just going to get more intelligent and more autonomous and that's going to mean smaller pieces of equipment that are kind of running on their own.

I think it's a really cool future. I just like look out and would love to see just a bunch of dog-sized robots going around picking weeds. It's very science fiction to me. Essentially, doing what people used to do, which is –

[00:59:18] JM: But totally plausible.

[00:59:19] RB: Right. We used to just have kids walking through the fields picking weeds. That's how you did weeding. Send the kids out to walk the beans. Why can't a robot do that?

[00:59:30] JM: All right. Ryan Bergman, thanks for coming on the show. It's been great talking.

[00:59:32] RB: Sure. Thanks.

[END OF INTERVIEW]

[00:59:43] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out

what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

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

Thank you to Triplebyte.

[END]