EPISODE 658

[INTRODUCTION]

[0:00:00.3] JM: TIBCO was started in the 90s with a popular message bus product that was widely used by finance companies, logistics providers and other systems with high throughput. As TIBCO grew in popularity, the company expanded into other areas through products that developed in-house, as well as through acquisitions. One acquisition was Jaspersoft, a business intelligence data platform.

When TIBCO acquired Jaspersoft in 2014, the architecture was a monolithic java application. Around this time, the customer use cases of Jaspersoft were shifting from a centralized reporting use case to real-time embedded visualizations. The use case of the Jaspersoft software was becoming less centralized and less monolithic, and the software architecture needed to change in order to reflect that.

Jan Schiffman is a VP of engineering at TIBCO and Sherman Wood is a director at TIBCO. They join the show today to discuss the process of migrating a large java monolith to a composable set of services. Breaking up a monolith is not an easy process, nor is it something that every company should do just because they have a monolith. In some cases, a monolith is just fine.

Jan and Sherman describe why the business use case for Jaspersoft was becoming less centralized, and why that change in business use case required a change in the software architecture from a monolith to a more micro service-like architecture. We talk through the modern use cases of embedded analytics, which is what Jaspersoft does. We also talk about the interaction between business analysts and data engineers.

At a higher level, we discussed the lessons they have learned from managing a large complex refactoring. This show is useful to anybody who is deliberating how important it is to break up their monolith into micro services, because Jan and Sherman articulate quite clearly how the business use cases for their software changed and why that led to a requirement to change the monolithic software architecture.

Full disclosure, TIBCO is a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:02:24.9] JM: Jan Schiffman, you are VP engineering at TIBCO. Sherman Wood, you are a director at TIBCO. Guys, welcome to Software Engineering Daily.

[0:02:32.2] JS: Well, thanks Jeff.

[0:02:32.9] SW: Thank you.

[0:02:33.1] JS: It's great to be here.

[0:02:34.1] JM: TIBCO was started in the 90s with its message bus product and this is still a product that's used by lots of companies. The message bus was the initial successful product that allowed TIBCO to expand, get into lots of other areas, and we're going to explore some of those areas. Many of the listeners are interested in the business of software and how the software industry evolves over time. Can you describe the initial TIBCO product and how the company grew in the early days?

[0:03:02.5] JS: Sure. Actually, both Sherman and I had a connection with TIBCO, I think early on. TIBCO started out as Technicron, which was a company that actually created a real-time messaging system that ended up on, I think trading room, turning falls, yeah. Yeah, moving a lot of financial data. I happen to work at Reuters in there real-time financial data systems at a time when they purchased Technicron. They later spun it off, it's about two years later, around 1996, I think.

[0:03:32.8] SW: Yeah. That's when I got involved during that time.

[0:03:37.1] JS: Yeah, and then it became Technicron, was then TIBCO.

[0:03:40.8] SW: Yup. TIBCO. It left the trading room stuff behind to Reuters and went purely into the data messaging world.

[0:03:51.0] JS: Yeah. Really the data center connecting systems there.

[0:03:53.7] SW: For applications and so on. Yeah.

[0:03:55.8] JM: Interesting. So as TIBCO changed and expanded, the company started making acquisitions and one of those acquisitions was your company Jaspersoft. Can you describe the strategic reasoning for that acquisition and what the Jaspersoft business was?

[0:04:12.6] JS: I think there are a couple of reasons. I mean, one was the fact that we were a subscription model and I think that that's something that TIBCO wanted to embrace and I think has become an important part of their go forward strategy.

[0:04:26.4] SW: Yeah, we had an open source space, which was interesting for them. They were very interested in – part of the subscription thing is what would be typical high-velocity. TIBCO's sales model was really big enterprise software sales. They had no light touch, light touch sales model. Jaspersoft had developed a very marketing funnel driven approach. A lot of leads, work them automatically, and then push them through a light touch sales model, leading the customers do, or prospects do their own proof of concept, and then go for a pretty low dollar, low dollar sales. That model is what TIBCO was buying into when they bought into Jaspersoft.

[0:05:12.4] JS: Unlike most TIBCO products at the time, it wasn't a perpetual license, so it was a subscription model, which is recurring revenue.

[0:05:18.7] JS: Yeah. Also we were reporting – essentially reporting and embedded the AI platform, which complemented other things that TIBCO had in the portfolio, like particularly Spotify.

[0:05:32.2] JM: That was a shift that happened in software purchasing that probably many people listening to the show were not in the software industry to be aware of. This shift from the long sales cycle, the big price tag, the negotiation that happens around the price tag of whatever large piece of software is being sold to whatever customer, the shift from that to SaaS, or pass, or infrastructure-as-a-service, I can see why the acquisition of a company for other reasons, but also just for the model of how did that sales process work; the customer acquisition process and the – I'm sure you had some domain expertise when TIBCO acquired Jaspersoft in the customer acquisition model of the future.

[0:06:21.2] SW: Really, Jaspersoft as a company came into open source in 2004, where they acquired the copyright and the project leads from a Jasper reports library, which is still involved with us heavily today.

[0:06:37.2] JS: We'll talk more about that actually.

[0:06:38.4] SW: iReport which was a visual designer for JasperReport. They started doing that in 2004. They got money, they got VC money, or funded money to go into open source, because they were previously a commercial software vendor. This was a new market, a new way to go to market in there. We went into that with what ended up being in what we call it open core model; an open source-base large community and then commercial extensions on top of that and we were selling Docker at one point, but that was a bit silly.

A little bit on the services end, but mostly it was about selling these subscriptions going forward. I think that was Jaspersoft was really leading light in that whole approach in that mid-2000s, right?

[0:07:27.7] SW: That model. Yeah.

[0:07:29.0] JS: We developed that over time. We did more commercial things, as well as keeping the open source community happy and going, and then eventually went down the – were acquired by TIBCO. We really had a very into the sales side of things, a very marketing funnel-driven approach. Large numbers of leads coming in, automating, watching the prospect, or the community, or coming into our website, attending webinars that sort of thing, using automation to rank them. "Oh, this person is ready to buy a type of thing," and going through that.

[0:08:03.7] JS: Yeah. I mean, this is something that people do with a lot of CRM platforms now and it's just really watching the conversion from stage-to-stage and within the funnel. You can put a lot of automation around this, and Jaspersoft was an early mover.

[0:08:17.1] SW: Yeah, we got a award from a company with the sales organization about the thing, because we did things like increase RFG conversation rate about 3% or 4% when we went to the automated approach, rather than the manual one.

[0:08:30.0] JS: We're getting pretty deep into that.

[0:08:31.2] SW: Yeah, yeah. The sales model here.

[0:08:33.9] JM: Which by the way, I think the – I mean, there's something that I would never have expected when I started the podcast, but I've actually gotten pretty interested in intelligent marketing, like how you spend your market. Because it's a hard problem and there's a whole lot of subjectivity to it. There's a whole lot of subjectivity, but there's also data that you can centralize decisions around.

[0:08:55.6] JS: Again, yeah, you can take just classic analytics. Whether a classic, or more big data looking for pattern recognition and you can apply that to marketing automation. That applies now to things like recruiting. Recruiting automation platforms, and at one point, I worked for one here in San Francisco as CTO. We applied exactly the same model, the same technology and really the same analysis to actually recruiting.

[0:09:22.6] JM: The product itself, the Jaspersoft product, that was around reporting and business intelligence, which is a – that's one of these areas of software engineering that has been changing non-stop since I mean, since at least 2006 around, or 2005 around that time when you guys were working on – when you started Jaspersoft and it's really, it continues to change, especially with machine learning becoming more accessible and also BI – people who are doing BI becoming more tech savvy, so that the lowest common denominator is getting more sophisticated. Back then when you started Jaspersoft, what did that term mean? What did that business intelligence term mean?

[0:10:09.0] JS: Very different things than it does today. I mean, back then you were looking at technologies like OLAP, for example, the way that you would slice and dice information and can analyze it is probably seems very primitive by today's standards. I'd say that –

[0:10:23.9] SW: With Jasper software that he started on the open source track, we did the initial releases of the server. We were essentially establishing the same model as the existing players like Cognos and Business Objects and so on. Where you're essentially doing, there's some level of reporting, there's some level of self-service for end-users, there's some level of data mapping and connection to data. People would experience that by going to, like they go to a portal and be able to interact with stuff, where they have reports being sent to them via e-mail, or whatever, right? That was the basic model originally there. Really what Jaspersoft was doing at that point was going, we're going to give you 80% of the functionality that you need to do, those sorts of things, at 20% of the cost or free.

That was the essential approach for open source business intelligence in those days. Now of course it's very, very different. The migration just was literally presenting to various people in my product team and engineering team this morning about how that is all shifted. Now the focus is all about, if you look at the analysis firms like Gartner Forrester and all these whose audiences are the enterprise basically, it's all about self-service for with a rich user experience, for people to create and interact with data and then go up from there into the various levels of statistical and up to the machine learning level there. Really, reporting is like table stakes and has been done and everybody does it. It's all about enabling really the end user to create their own experience.

[0:12:11.8] JS: Yeah. I think it's also important to keep in mind that the definition of what is reporting is also – just like the definition of BI itself has changed, and now we're more focused on actually embedding analytic visualizations in existing customer applications. That in some way has become the new reporting. I mean, people less so are wanting exported PDFs, which are then mailed on a schedule to some inbox, where they really wanted something that's visual and on-demand and embedded in a pre-existing user experience.

[0:12:42.2] JM: Right. One of my earlier, actually two of my earlier engineering jobs, I just recall getting into the office in the morning and there would always be a report that I would be e-mailed that would have something to do with statistics from the day before that were map reduced down to something that I could read, that would be useful. Now it's more of a – well, I mean, it varies from company to company and morning reports are still quite useful in some context, but I think the idea has moved increasingly towards constantly updated visualizations, dashboarding, and then on the data engineering side, it's like streaming data instead of batch data. It's interesting to think of the use case being mirrored by the infrastructure that sits underneath it.

[0:13:33.2] JS: Yeah. The expectations of users now, I mean, they want actionable data in real-time, as opposed to just this overnight batch processing. That's changed in just maybe the past seven, eight years. I mean, I actually came from a Jaspersoft customer, there'd been a white paper for Jaspersoft and we did BI reporting for a major American fast-food enterprise. We started out actually compiling these reports overnight that would be then e-mailed to the senior executives at a lot of the franchisees. That actually changed to drill-down dashboards by the time I actually left and came to work for Jaspersoft itself.

[0:14:19.5] JM: In the earlier days, we didn't even really have the term data engineer, we didn't have data scientist back in 2006, but there was – I guess there was a DBA role that was sort of – there was some technical people who would do certain things and there were business users that would do certain things. That is very different for how things are today. I mean, there are still are very technical people that have to do this stuff, but some of the stuff has been pushed into the tooling itself, like the tooling takes care of the data engineering and then the data engineering roles are in some cases even more specialized, significantly more specialized than the DBAs were back in the day. How has that interaction between the business users and the business use cases and the role of the engineers, how has that relationship between the two people and the relationship between them and the data platform itself, how have those things changed over time?

[0:15:13.2] JS: Back to that, my earlier comments about what's analytics nowadays; the expectation is that the end user has got control over the data access, so the data engineer's job is to, or DBA or whatever, is to make sure that that data is available in the various tools, etc. Some of these tools require behind-the-scenes loading, or something like that. They are hooked into data lakes, or whatever. There's this push and pull between data volumes and what's the thing, versus responsiveness and interactivity for that end user.

[0:15:56.5] JS: Well, if we map that into to JasperReport server, we have domains which are generally it's a way of expressing data, allows you to have aggregations and calculated field and a lot of drive data from that. Then surface that to an end user who can then use that in a self-service fashion to analyze and then to generate embedded visualizations, or reports, or dashboards from that. There you can see that the two different roles; so one really has much deeper involvement with the data itself, and it's actually a process that provides that data to the end-user.

[0:16:35.1] SW: There's often a data security aspect to all this as well. This –

[0:16:41.1] JS: Governance.

[0:16:41.7] SW: Yeah, etc., which is very, very important for when you're working across an enterprise or something like that, people are going to use a particular data. The people who governing that data need to have the controls to do that, sometimes they can do that into the data sources, but they can also do that within Jaspersoft and these other tools as well.

[0:17:02.5] JS: Right. Does that answer your question a bit? I think –

[0:17:04.7] JM: Yeah, well so what you're alluding to is that there's continued need for visualization. Jaspersoft was originally used for report generation and what Business Intelligence was back then and the product has migrated, it's evolved to serve more modern needs within those same organizations, so it's gone from this batch reporting to the continuous data visualization, dashboarding and reporting, and that's also reflected in how people want to consume that data, as opposed to just this one single screen that you're viewing all your data on, you want to break it up into this data visualization composable set of elements. That's how the product has evolved, as Jaspersoft has become this thing that you can use to build different dashboards, you can connect to these different data sources that are within your data platform.

The data platform itself is necessary for building these visualizations, whether we're talking about batch visualizations, or the streaming real-time data visualizations. What are the changes that you have seen around the data platform over the last decade?

[0:18:20.0] JS: Well. I mean, traditionally it was your well-known industry standard databases; MySQL, DB2.

[0:18:30.1] SW: Oracle.

[0:18:31.0] JS: Oracle, SQL server. Typically you'd open a JDBC connection to it and it would be typically a schema-based database, and they're all pretty much – they behave the same way. The way they persisted data was for the most part the same. Their SQL dialects varied very, very little and then things really started to change with big data.

[0:18:55.8] SW: Actually even before that, I was seeing a lot of customers in the 2007-2008 start to go down the track where all their data was in behind a message queue, that's what we called them. It's a JMS, or something like that behind the scenes. You would be interacting with that through a SOP. Everything was XML, right? That's what we're going to do. SOP request, push the SOP response into your report, or your visualization basically.

[0:19:27.3] JS: I really miss [inaudible 0:19:27.8].

[0:19:28.8] SW: Yeah. Oh, man that whole XML and all that stuff was the core bar of these over-engineered standard that –

[0:19:38.8] JS: We're really giving away our age now.

[0:19:40.9] SW: Yeah. Unfortunately, yeah. We were seeing that. Then we were able, because of our open source base to rapidly adopt to a lot of the big data things that were coming out. Originally around Hadoop, hive, hive JDBC sorts of things, and then watch that evolve over time. Now SQLs, the Mongos, and that's the thing of the world leveraging their – I don't know, I can do geolocation sort of things in Mongo or whatever.

Then things move to rest. Everything's rest, so I've got – I have complete environments, complete enterprises now where I'm not connecting to a database at all, I'm just doing rest course.

[0:20:24.5] JS: Right. Or maybe an integration of them as well, which you can do with some federated query. It has I think evolved tremendously, but one thing that's been I think to Jaspersoft's advantage is that because it is open source, we've been able to embrace all these evolving technologies through using other open source products, open source projects.

[0:20:45.8] SW: Be very API-driven both from how people access the various visualizations of data, as well as hooking to the underlying data sources that we need.

[0:20:55.4] JM: There is this notion of the operational database and the analytical database. The operational database is if I'm a user that's interacting directly with an application, like a banking application, then I'm interacting with the operational database, I'm interacting with the transactions that are inside the bank, I'm interacting with the debits and credits of the bank, and then there's this analytical database and oftentimes that's used to perform aggregations. You're aggregating all the transactions, you're doing analysis, that's why it's called the analytical database and you oftentimes have different schemas in these different data stores, because if you're doing aggregations in the analytical database, then that's a very different operation than doing individual OLTP, that the point, you're changing a document or changing a row.

You have this ETL job that typically occurs, or ETL pipeline where data from the operational database is taken into the analytical database in batches, or in some cases, and streams, or teed-off to both of them at the same time, or models like that. I'm guessing that most of your users are reading from the analytical database, although you can correct me if that's wrong. I'm just curious how the ETL pipeline affects how you think about product development of a embedded visualization product.

[0:22:20.3] SW: Okay. What's happened over time – well, so first off, Jaspersoft has as an option in its product suite is Jaspersoft DTO, which is a white label of the Talend data integration platform, so which is great for creating that analytic data store that you allude to there. We still see a lot of people doing that work. Either they've done it themselves, or they use Jaspersoft ETL to do that work.

One of the things I have certainly seen is as with the move to Big Data and the whole schema on read thing, there's less of that movement around the place nowadays. You may be dumping things into a data lake and you're relying on the various power of the schema memory type of APIs and systems to go and provide that analytic view of that information. The whole thing where I'm just – you're just seeing what's happening up on the cloud providers now where just dump everything in a flat file and then run a stateless SQL thing on top of it.

[0:23:23.7] JS: Right. Yeah. I think databases have gotten much more efficient actually, executing pushdown aggregations that way, so you can do a lot of that on the fly. Where before, it was really difficult, and that's why I mentioned OLAP before, because that was a technology allowed you to take your data from really a transactional database and then to look at that in an operational view.

[0:23:44.7] SW: Yeah, and you apply the star scheme or snowflake sort of schema model. That's the thing. Yeah.

[0:23:51.2] JM: I think the thing we've been skirting around is the fact that over time, the demands for what a customer wanted out of a data visualization solution changed. It became taken from this monolithic reporting system to a more flexible modular reporting system, which could be articulated as embedded data visualization, or embedded BI, where people want to have a small little dashboard component that fits into some system that they're building. Maybe it's an internal tool, maybe it's a customer-facing tool. Whatever it is, it's some BI chart, or analytics thing that you want to present to the user, if you are the user interface engineer.

In order to create that embedded BI tool, or dashboard, you need to connect to the data sources. The data sources have also changed. Where this gets us is that you needed to migrate the Jaspersoft monolith to a Jaspersoft set of services, or something that is more composable, more modular. Jaspersoft was originally engineered as this monolithic java application. Can you describe the initial architecture of Jaspersoft and describe why that didn't exactly work for the world going forward at a certain point?

[0:25:22.5] SW: I was actually the architect for the JasperReport server and supported the community for quite a few years, and did lots of different things and wrote a lot of code, and my name is still on that, on chunks of that monolith. Originally, because we were working with JasperReports and we were sticking with the java market basically, a lot of our original – that's, that's who we appealed to, so we were building what was – what we tried to do as a best practice at the java application at the time.

Though at that time, there was a big battle between what you've got to use J2EE at that point, but I'd always thought it was like, J2EE was a game core bar. In fact, some of it really is called cor bar behind the scenes, so I picked up Rod Johnson’s work from all around spring and that thing. We did was a spring-based application without HABs and all that thing, very module and componentized. At that point, we needed – so it's a simple war, Java war. We have like a application archive connected to a database with a meta database behind the scenes there.

That made it very easy to deploy. We could go into lots of different J2EE environments, simple thing like Tomcat and we could also do things like use web standard tools like, “Oh, you wanted to load balance that.” Great. That's fine. Nothing proprietary in there to be deployed anywhere. That's that has stood as in good stead as we've moved around and the technology environment go to the cloud, blah, blah, blah now.

[0:27:06.3] JS: Right. I think something that really changed the model for us was actually being probably what was the first BI platform in AWS in their marketplace. We started really thinking about the cloud and how we actually migrated there, probably about six years ago, five years ago.

[0:27:26.1] SW: Initial release was 2013, early 2013.

[0:27:29.3] JS: Okay, right. It's about five years ago. As AWS and cloud technologies evolved, so have we along with it, which is now led to what we're doing next, which is taking the actual data access and rendering core, which is the original JasperReports library and we're packaging that in a container and ultimately, we'll be offering it as a full micro service. We've been embracing Docker technology, I think we had – we had our own support for Docker through a set of Docker files that they do in Jaspersoft. Docker about two months ago.

[0:28:06.1] SW: As typical, we were hearing about Docker from our customers who because of that standardized Java technology, was we're able to build their own Docker files. If you've got a Docker hub, you will see a vast array of these open source things.

[0:28:21.6] JS: Actually that raises a very good point and sort of a blank to fill in here is that our customer base started shifting from users of a standalone BI server to those who are SaaS providers. They were requiring that we fit in to the modern SaaS architecture. They are some of the first to really embrace Docker.

[0:28:39.8] SW: Yeah. Absolutely.

[0:28:42.0] JM: It was a process of architecting that modularization, because I know there's a lot of people that are listening that I think are going through this in some form or fashion, or they're in the middle of their micro-services re-architecture, or they have decided not to re-architect and they've decided to double down on the monolith, and their – making certain re-architectures to respect the monolith. What was the architecture and planning process like when you – as you were deciding to go this more modular direction?

[0:29:15.7] JS: Well, I mean, I think for us it was easy, in that there was this core, this JasperReports library core rendering engine written in Java, it provides data access and it provides visualization and report rendering. It was very easy to pick out what would be the first natural standalone service. We're also able to take some of the APIs that were part of the server with it, and are visualized JS, JavaScript SDK, that allows you to embed directly into, without an iframe directly into an existing web-based application. That made it very simple, right?

What we could do is we could actually take that codebase. Now there was a lot of consideration of where do we go next, what do we – how do we modularize what we're doing is in such a way that we can start to dissect it? I think the reality is you can't always think about it that way. You look at your existing code base and think, "Well, how can I take it apart, maybe unwrap some of the dependencies and untangle them, so I can have standalone services?" I think what really we're doing at this point is thinking about, “Well, how do we decompose the services? Do we actually want to use the same code?” Who knows? There's so many opportunities now to use technologies and languages like NodeJS, or use Go, and which may be more appropriate for a lot of microservices.

It's not always a matter of taking your existing code base and breaking it down to pieces and then making those available as microservices. It's really thinking from the ground up, “What are the services that my existing monolith offers, and how do I want to render those as independent services?”

Even some of the services that your application may currently have are really part of the past platform that you're going to launch on. Why would you want to replicate a service that may be part of a juror, or AWS, or Google Cloud? I think when we’re architecting four services in the cloud, you really have to take a different view and take a step back from your existing code base often.

[0:31:19.7] JM: Although, if you take that to the extreme, it gets you to deploying everything in AWS Lambda functions, or Google Cloud functions.

[0:31:29.4] SW: That's certainly what I'm seeing in now with our customers and prospects right now, where the attraction of being able to create your own environment, or leverage the cloud providers and serverless environment is the way they want to go, because they want the different economics. I'm not paying X amount per hour, or per minute for an AMI, I'm paying on a transactional basis. I'm also allowing greater flexibility and more, say global, or whatever reach going through that, using that serverless type of thinking and the services around.

[0:32:06.3] JS: It's the ultimate alignment really of revenue and cost.

[0:32:10.0] SW: Yup. Absolutely.

[0:32:11.2] JM: We have done a lot of shows on microservices. When I was talking to you guys before the show, you gave some reflections on that aggressive move to microservices from the software development community. It's easy to get the impression, especially if you listen to this show, but everyone should pick up their monolith into microservices and you're inevitably going to move your monolith to microservices.

I mean, I think some people do it just to do it, or they think about doing it just to think about doing it. When in fact, it's an open question like, are you actually producing a better product by moving to microservices? When you think about the trend between monolith and microservices more generally, what thoughts do you have? Is there a herd mentality towards doing that, or is this a rational decision that the market has made?

[0:33:01.6] JS: I guess the comment I would make is that what I'm starting to hear, and maybe you can confirm this because you probably have a better overall view given the nature of this podcast, but that there's some backlash now against breaking everything down to microservices, just because the level of complexity. Sometimes, I wonder if by the time people would actually break their monolithic application or services down into microservices, actually it will no longer be something that people – it won't be fashionable, it won't make good technical sense.

[0:33:37.2] SW: Yeah, this probably – I'd say that there's a lot of use of microservices in net new. I mean, whether you're migrating a monolith or not is irrelevant. For example, nowadays the web developer who's using angular, react, etc., they're creating microservices. Well, the microservices are created for them to deliver the data, or other functionality between the front and backend. Then behind the scenes, as microservices could be in a Lambda queue, or whatever, being processed and then the results would turn to whatever.

It's probably not so much of a migration type of issue, but it's more like, people are doing that more and more. Now what I've seen is that they end up to Jan's point, in microservice hill, right? How many microservices have you created? I have 20, that sort of thing. You end up with this this berserk, unmaintainable –

[0:34:33.2] JS: Well, your DevOps pipeline just starts become unmanageable as well.

[0:34:36.6] SW: Yeah. Your application sort of tail-bite on as well. I've got to wait for all this stuff and maintain all stuff, oh my God.

[0:34:41.7] JS: That’s right. Yeah.

[0:34:43.4] JM: Right. Yeah, open question and we're probably not going to come to a conclusion on –

[0:34:48.3] SW: We can’t solve it here.

[0:34:49.7] JM: We’re not going to solve it here. This migration that you were making from the monolith to the services, or modular, or whatever you want to call it, but because this maybe microservices more modular components, this was occurring at Jaspersoft after you got acquired by TIBCO and it was happening at the same time that you were making a migration into the cloud.

Well, I'm sure there were some advantages and there were some disadvantages to the simultaneity of those two things. I think there are a lot of people who get into this situation where they decided to time these two kinds of migrations, the migration and the breaking up, or the fracturing into different services. What's been that experience of the simultaneity of the movement to cloud and modularity microservice restructuring?

[0:35:39.3] JS: Actually, it wasn't really simultaneous for us, because the move to the cloud was really five years ago when we first had our AMIs available for our server on the AWS marketplace. It was really the evolution from the server being run on an individual EC2 instance as part of a customer's application architecture, to this push to containerization, which I think has driven JRIO and in alignment with what our customers are doing.

[0:36:13.0] SW: One of the interesting things with for us as a software vendor in the cloud is the move to microservice architecture, serverless, or that thing really mux around with your revenue, right? That the original model, say on some of these – I've got a bunch of AMIs, people are paying X amount per hour, or minute, or whatever it is, however long they're running is fine. Now you can actually bump that up and that's great. When you're particularly going stateless, right? Now with the various cloud providers ability to just spin up a container. There's no EC2 instance or whatever, it's just being spun up and it may drop down again. That starts to get into like, “How do I make sure of my revenue? What am I getting out of this customer?” Moves you to much more of a pay for what you use model, as opposed to the way the virtual machine approach. Now where everything is really, really, really going serverless, that's a real problem in software selling.

[0:37:22.0] JS: Can you tell us, Sherman works in sales.

[0:37:23.7] SW: Yeah, yeah, yeah.

[0:37:27.2] JM: Interesting. The migration; how many people were working on and how did you divide up the work and what was the technology stack that you were migrating to? Can you give me more of a picture for the architecture and the planning process around that migration?

[0:37:42.9] JS: Teodor Danciu who was the original creator of the JasperReports library, he and his team actually took the initiative and started working on JRIO. It was really their understanding, their ownership of the JasperReports library that made it easy for them to really do. They worked with our API engineering team and the front-end engineering team that created the visualize JS SDK, the JavaScript library, and the three of them, three of those groups working together really were responsible for creating JRIO.

Initially, to be honest, the scope of it was a bit more constrained. As we got more into the project, we suddenly realized that we could start to incorporate more functionality into it than we originally planned for, which is why we then extended the APIs out to the visualize JS, the JavaScript library, and we found that actually, we created something that was far more capable than we originally anticipated.

The technology stack remains Java, right? The front-end library is JavaScript, although we will be migrating to typescript. The actual library itself is Java, and that is served up via jetty. right? The idea was to take something originally that was language dependent, which was the JasperReports library, and then turned it into something that was really language independent, environment independent, so now I could run this as a containerized service anywhere and integrate it into any environment.

[0:39:23.1] SW: One of the interesting things that we got into is a lot of the way that our APIs work, they are actually conversational. Like run this report for, me now give me this X port for example, right? We actually do have – we've got a cache behind the scenes within the service that the various containers can all collaborate around. It's not a one of the things of “microservice,” oh, it’s all stateless, stateless. It’s like, is we needed some level states, so we included that cache within the environment.

[0:39:57.2] JS: Yeah. I mean, I think it’s much more difficult back with really stateless servers. That's aspirational, yeah, and the way that we've approached it is by having shared persistence, or a shared cache, where we can persist these things that need to be shared across instances, but do need to be persistent.

[0:40:14.3] JM: The product today, you've got actually a number of different products; you've got the open source Jaspersoft software, you've got the APIs, you've got Jaspersoft for Docker and JasperReports.io. What are these different products? How does Jaspersoft, like how do people use it today?

[0:40:34.5] SW: Well, it starts on the reporting. It is the JasperReports library and related to that Jaspersoft studio, which is the visual designer, where you can create and test out your report, connect to data and that sort of thing. You start with that, and then you've got the JasperReport server in the middle there, where you can deploy your reports and then with the servers where there are a variety of additional services like self-service and metadata layers, the repository and that sort of thing.

With where the API is all basic are sitting, the rest pieces, the – it's what visualize.js collaborates with, and you can embed screens and that stuff as well. We've got the Jaspersoft ETL as an option, we've got Jaspersoft ADS, which is a enterprise class data virtualization environment. These other things like, really Jaspersoft for Docker is literally just a deployment mechanism for the JasperReport server.

[0:41:30.3] JS: For the entire server.

[0:41:31.2] SW: For the entire server, right, at this point. That's equivalent to what we use a cloud formation template on AWS, for example, to deploy Jaspersoft into your account up on Amazon. JRIO is going to be starting as a very simple lightweight thing, right? Really focused on out of – whereas, JasperReport server when it's running in a container, it's the full stack, it is that monolith. We do see a lot of people out there, when they go down the monolith route, or they've got the monolith, they just get – it might be a huge freaking container, but t's still a big – it's still in the container environment, may get all the coordination and that thing around it.

Whereas JRIO is aimed out of the box to be that, you're going to be running completely natively in containers, multiple containers doing different things and we can – people can plug that into whatever. They can YAML it, they can Kubernetes it, they can do whatever they want with it, stick it up an EC2, ECS for example, or that sort of thing.

[0:42:34.0] JS: Yeah. Well, the use cases are a bit different, so with the actual full JasperReport server, we have multi-tenancy, we have scheduling, we have again, tendency-based data governance, and JRIO doesn't incorporate all of those. Or really, those things are delegated to the application builder, which JRIO can do is access data and then embed those visualizations directly in a pre-existing contact, right?

Let's say, for example, unlike the way it was 10 years ago, where maybe you had a dashboard in a portal-based application somewhere and that is where you looked at all of your BI, you looked at all of your reports, now we want to embed visualizations that are actionable, right? Within an application in which you're able to actually take action, right? It's not, I'm looking at my visualization, I'm looking at my dashboard, then I'm context switching to another application to actually take some action. We're embedding it within the same app. I think this has become much more modern model, and this is really what JRIO directly supports.

[0:43:36.2] SW: Yeah. We coined the embedded BI terminology back, when was that now, when we were talking about that? Early 2013 type of thing?

[0:43:44.8] JS: Yup.

[0:43:46.1] SW: Lots of other folks have come along and they're using the same thing to appeal to OEMs and the enterprises, you want to embed in portals and all that sort of stuff.

[0:43:55.5] JS: That's really, I guess, the big difference and between JRS as we call it, which is the full-blown server and JRIO. As we've seen the needs of a customer-base and needs of users really evolved, we've moved to something lighter weight that's really focused on those specific needs.

[0:44:13.6] SW: A basis to start, we'll put other services around that that collaborate with this, to add additional function and like eventually the data governments and all these other things. Yeah.

[0:44:23.3] JM: Interesting. What have been the biggest challenges in developing that new product?

[0:44:28.5] JS: Because this move to a containerized environment is a very technical thing, you really needed to work on the use cases. I would agree, really how much is enough? How much is really going to be useful to our customers? I think that's where I saw the scope expand once development was actually underway. We really developed this in a truly agile fashion. We would look at requirements and reevaluate what was the MVP for this as we were in motion. That's how we ended up with what we have right now and expanded it out and really looked at and considered what were the core needs of our customers, and aligned what we were building with that.

[0:45:11.9] SW: That collaboration is the thing that Teodor and his team do is like, we think about doing this. "Oh, that means we can do that." That's the back and forth. It was a lot of fun.

[0:45:21.7] JS: Yeah. Well, they're a great team to work with and really very, very agile. Really great –

[0:45:27.1] SW: They're all in Bucharest, Romania.

[0:45:30.2] JS: They are. Yup.

[0:45:31.5] JM: The planning process, it took a few at-bats, but then eventually you really identified the surface area of what you needed to change.

[0:45:39.3] SW: It evolved completely through the process. The initial idea was, all we'd have some – we'd have this microservice that exposed some rest endpoints and you could generate just content.

[0:45:49.4] JS: Yeah, that was it.

[0:45:51.5] SW: Now it can do that, but now can also do the deep embedding of the visualizations through the JavaScript API as well.

[0:45:58.7] JS: Right. I think that really turned out to be critical. At the time we thought, “Well, this might be interesting.” Then we realized that without it, its utility was somewhat questionable.

[0:46:10.3] JM: This was a significant migration. What lessons did you learn around software architecture, around estimations, around just managing a large software project from this significant migration, the significant refactoring?

[0:46:28.8] SW: At least one of things I'm sure Jan can chime in here, is that you needed – when you start, you don't know what you're doing, right? You need essentially proof of concept time to work out all the knobs and dials on the thing, and then you come up with a candidate architecture, and then you iterate on that stuff. I think that was the – you don't go down this track like with any technology without spending a bit of time kicking the tires, thinking about it and Teodor and team did that really well, with house-arresting them, that sort of thing.

[0:47:02.7] JS: Yeah, and then they’re somewhat a unique team. I think, one of the lessons we learned from this is as we start to build a set of, I guess more complete services around it, we have to consider how do they work together and how do they work alone. Does it add value on its own? What does it add to the existing set of services that we have? I think we've learned definitely something from JRIO just because the scope expanded and we realized, this was really going to be the MVP for it. We weren't really sure at the beginning. I think as a result of that, some of that experimentation has taught us lessons that will take to the rest of the services that we develop.

[0:47:44.7] JM: Cool. When you think about products, the data visualization layer has changed rapidly in the last 10 years and same with data platforms, and it will undoubtedly change again in the next five or 10 years, what are the predictions that you have around how this interface between users and data is going to change?

[0:48:08.2] JS: Well, I mean, I think we've already seen some of this, and I think the service sees this with prospects and customers all the time, it's going from having more canned visualizations, ones that you can perhaps style to I really just want some data, because I've got my own visualization library, and they want to be able to easily just get back data, maybe perhaps with JSON. I don't want the complete embedded visualization. Or maybe I want just a partial visualization and I want to augment that with my own, whether using D3, or any other library.

[0:48:43.9] SW: That's making things go down say a more technical level, but I think it's also going up the other way to people are expecting a richer information experience. The tools and platforms that give them that are the ones that are basically going to win. How do I – I've got some – I'm not just looking at say, operational data anymore with some lights of KPIs, I'm really looking at more of an analytical type of model here that's looking at my process, my business, whatever, and that's people want to get that, increase that value out to there and save themselves internally within their organization, or for a software vendor, or SaaS provider to say, I need, I have richer and richer analytics going out to my customer base.

We've seen that, we've done the analysis to see the software vendors who really go for that morality end, they're able to charge where it's more valuable, that sort of thing. I think for us as – majority of our customer base OEM Jaspersoft, I think our ability to help them get there, because they might not have that skill in-house, I think is going to be pretty critical going forward.

[0:49:57.9] JM: Okay. Well Jan and Sherman, I want to thank you for coming on the show. It's been really great talking to you about this migration and data platform and data visualizations with a wide-ranging conversation.

[0:50:07.9] SW: Great. Thank you very much.

[0:50:09.6] JS: Well, thanks Jeff. I also just want to say what a pleasure it is to be on your show. I've been a big fan of your podcast for a number of years, so it's a real pleasure.

[0:50:18.3] JM: Cool. Thank you. We really appreciate you listening and maybe you can give me some feedback on the show afterwards. I'm always looking for ideas or improvements to make.

[0:50:25.8] SW: I think this was a good conversation. You brought a lot to it and I had fun, I know that.

[0:50:31.5] JS: Absolutely.

[0:50:33.0] JM: Awesome.

[END]