Tony Shaw

I'm going to be handing off the hosting responsibility to my colleague and friend Dave McComb, in just a minute. I've known Dave for many years, but if you're not familiar with his work, he wrote the book quite literally on semantics in business systems. His current area of interest which is an evolution of that is data-centric architecture. We touched on this topic briefly yesterday, making the point that applications change but the data is forever. And so, this particular case study that Dave's going to introduce is a really interesting example of how to manage this. So, Dave if you can provide the right context and introduce Mark and Hamish for us, I'll turn things over to you.

Dave McComb

I'm the president and co-founder of Semantic Arts, which is a consulting company that helps firms transition to what we call a data-centric approach. I'm going to say a few words about what that is, and why it's necessary, but then Mark and Hamish have a very interesting story I think you'll find quite fascinating and then we'll just have a Q&A.

When they started this adventure, they had all their toothpaste in the tube. Everything was well organized and curated, and all in one place. And then success brought them up to a larger context where the toothpaste was all over the place, more of a brownfield as some people call it. We're going to have them then describe that, and then we'll have the audience Q&A at the end of that. To start with, so we do a lot of work with large enterprises and after a while, I finally decided to sit down with a graphic artist, and have him, in this case it was him, do a composite of what we were finding in all these large enterprises, rather than read-write the same pictures over and over again, which we had been doing. And this is what he came up with. And I think many of you will recognize yourselves in this picture. You can see the big data warehouse up there at the top and the message bus at the bottom and information over there at the left. And, you got the idea that that is what on average, most large enterprises' data architecture looks like.

Now my conclusion to that, is that enterprise software as is currently practiced and currently exists around the world, is a mess. Layered on top of that mess is something called systems integration, which is even worse. That's most of that diagram on the previous slide, which is bizarre. I'm going to talk a little bit about why everyone just thinks that's normal and okay. It's like that old boiled frog story that we'd been in this for so long now and you just got to fix one more ETL script and everything will be fine. So, that's what we're going to talk about.

This state of the current world, we call application centricity. It's where you focused in on the application as the way you're going to solve your problems. So, if you think that the application is the predominant thing, and data is the second class citizen, you're application centric. And you're not alone. I mean, virtually everyone is, everyone has pictures like this. There is application code. Somebody writes it, and we've gone through generations and generations of this, but there is still tons of code. That code accesses a shared database, and that code or some other layer of code presents a user interface. And people present that as, this is where our architecture is, this is how things are built, this is the norm. There's an illusion that there's separation of concerns here in layering and stuff, but there really isn't. Every change anywhere changes everything. You change some of the database, you have to change the codes, you have to change the UI, all that stuff.

Now, you look at this and you see the little can at the bottom and you think, that's a relational database. I should be able to just directly add access to the data and skip the application code and the UI and the APIs and all that crap. Just let me go straight. In fact, I want to do an insert into that database. Well, anybody who's ever worked with any enterprise application knows there's nobody in the world that's going to allow you to do that, because technically, you might be able to do that, but you're going to bypass all the integrity checks and validation and constraints and authorization, all the stuff that exists in the application code. And you think, wow, surely I could at least read data directly from there, but no, they won't allow you to do that either. And they're right, because this model is so complex that the odds that you're reading it correctly are slim. If you take this stuff from column A, you got to know that column Z, if it has an X, that's really a debit, not a credit and all that strange stuff that exists in databases. So, the real truth is while there's a database sitting there, really the only way through, is the application code APIs, maybe you get permission to have an ETL. But really, this is what's going on.

Now, the issue is, how does this persist? What the slide I didn't have there is people then do that 100s or 1000s of times, and you have 100s and 1000s of application systems, each of which has its own data model, which is arbitrarily different, not inherently integrated with anything. So, how did we get there? We get there, we did a bunch of studies with hidden cameras and microphones and boardrooms and stuff to try and figure this out. Here's how you get there. Somebody has a business problem. They put together a committee. We have to solve this business problem. About half the time they figured out their business problem isn't really an information systems problem so they put another committee together and ask what are we going to do about this? And at this point, the trap is set, but it's not sprung. They have this choice they believe in. The important choice coming out of that committee group is, are we going to build a solution or are we going to buy it? Do we buy an application package or we build one in-house? Now there's a three-way choice. You can also rent software as a service. And it feels like you're making an important decision when you do that.

But actually, it doesn't make any difference at all, because all three roads end up in the same place. They all end up with yet another brand new, arbitrarily different, complex application data model, that can only be accessed through the application code that you either built, bought, or rented. So, you end up exactly the same place with one new silo in your architecture of silos. So, that's how these things get created. This is the problem to solve, and we don't have to live this way.

There is another approach we call it data-centric. As we look at firms or parts of firms, you say you're down this path, to the extent that whatever application functionality you do have is loosely bound. So, the application functionality you have in those systems that is described as very tightly bound. Half of all lines of code in most systems are referring to either the schema or the messages, the API or something that is application specific. And so, when anything changes, there's a very tight binding, a lot of impact there. If you have 8,000 applications, you have 8,000 data models, when it gets to where you've got a single data model that is simple, and simple enough that analysts and people writing AI routines or all the other people want to consume this, can understand it. While it can be single, it needs to be extensible because there will be sub domains that want to add on and stuff without breaking or disrupting what's going on.

A federated approach says that you don't have to have all the data in one place as we currently think, with the data warehouse or a data lake or big data. You can have the option of leaving data where it is, as long as you have a single model that describes it. So, in that definition, about five years ago, and I

encourage if you've got a minute right now, you can write this down and go straight there. About five years ago, we put together something we call the Data-Centric Manifesto. This is some of the key principles and if you click on it, there's a longer description of what we're talking about. I think most of the people on the call would probably agree, data is a key asset, should be self-describing in an open format. Getting locked into proprietary formats is about the worst thing you could probably do.

Take the access and security out of these application silos and put it somewhere where it can be shared. And an application can come visit the data, they do whatever they want. But they have to put their so-called insights or transactions or whatever an application does, back into the data layer so that everyone else can share and understand. That's the basic premise of the manifesto. We've so far had 856 people and almost all of them have put very thoughtful and some cases lengthy responses there when they signed the manifesto and I encourage anybody on the call to go out and do that now. Mark Avallone did, quite a while ago. I'm going to tell you the story about some of this in a minute. But that's one way you can join the movement or find some kindred spirits.

Finally, we say, if you're going to become data-centric, you really need to be working on two things simultaneously. One is, be able to model your entire enterprise in a way that you can combine and federate and reduce your systems integration debt if you will. But the other is, you also have to have an architecture. When you decide to move away from the application, there was a lot of things meshed in with all the rest of the stuff in the application. They were actually doing some useful things like security, and authorization, and authentication, and identity management, and resolution, and validation, and constraint, all those things, now have to move to the architecture. Have a way of expressing and managing that, architecturally not application-wise.

Then finally, we're getting close to our guests here. We also had an assessment out there. But once we put it out there, if you want to jot that down or click on it or go there or something. This is a quiz. When people hear about data-centric, the immediate response is, "Oh yeah, we're data-centric". Sometimes it's their data-driven which is nice and interesting, but data-driven is about the relationship of data to decision-making. I think it's a great idea. A data-driven organization says, rather than relying on people's judgements and opinions for decisions, let's rely on data. It's a great idea. But that doesn't mean you can have a lot of silos in data that you're making decisions on.

So we put this quiz out there, and most people take pretty low score on it. And Mark and Hamish sent this email, "Hey we took your quiz, and we got an 80, 80%". I was, "Come on, I don't believe it". So, I call him up, and we have a long talk, I'm going, "Wow, I think you are data-centric". And I flew up to meet them in Virginia, and we spend a day and went through their architecture and answered a bunch of questions. I think they're probably one of the most bona fide completely data-centric firms out there, which is why this is going to be such a lot of fun on this panel discussion. So that was that and I think that brings us to our panel. Mark and Hamish, you want to introduce yourself and then I'll start asking a couple of questions?

Hamish Brookeman

Certainly. My name's Hamish Brookeman, head of data architecture for S&P Global Market Intelligence. A briefly describe what our firm is and also thank you Dave, for the kind words. Some of the evolution of it. So, S&P Global for those who are not familiar with it is a financial services market information

provider, and index business. It has various divisions including ratings, who many people have heard of, the index division and other divisions. The specific division that Mark and I work for is market intelligence, which in one sense can be thought of as an information provider. We think of ourselves as that technology enabled data company. We sell data. The widgets we produce is data.

As such, there's a premium on data, since it's our business managing, selling data. It's our prime asset. I will admit, when we took the quiz, Mark and I both worked for a smaller company at the time. It was SNL financial based in Charlottesville, Virginia. Same business line, and it was about 3000 employees. We were global at the time. We were subsequently acquired by S&P Global. When we were the smaller shop, it was easier, and certainly less complex. We really were a very disciplined and highly rigorous data-centric practice.

And we were absorbed into a rigorous organization that looked more like the diagram that Dave showed as a diagram, the visual representation. That acquisition was in late 2015, and so, in the subsequent four and a half years, we've been applying what we knew on that smaller scale, attempting to bring that slowly over to the larger organization of the division we were into, S&P Global. And it hasn't been easy, it has been a journey. So, with that I'll let Mark introduce himself.

Mark Avallone

This Mark Avellone, head of architecture, S&P Global Marketing Intelligence. It's a really close partnership and I think as we talk through this panel and answer some of the Q and A, you'll see the interplay between data architecture and the data architecture practice that's extremely well integrated into the modus operandi of the company. And then how we operationalize that and develop the tooling and SDK and APIs and the lower level systems right around that ordinarily abstract practice that make it something that really operationalize and has successfully evolved for almost 20 years now.

Dave McComb

Speaking of 20 years, that's where I want to start because I loved your story, the way you guys told that to me when I came out and visited you. In fact, weirdly you guys had an exact date when you became data-centric. And in fact, you've had your 20th anniversary now, already. It was may, wasn't 2000? Tell your creation story.

Hamish Brookeman

Certainly. And yes it was May 5th, 2001. So, we're 19 years old now we'll be 20 next year. As with many things, our data-centric practice was born out of, I would say expediency or laziness as many great solutions are. At the time we were re-platforming, we worked for this company, SNL Financial. It was financial services information firm. We were re-platforming from a FoxPro database to Microsoft SQL server. And so, in order to re-platform from FoxPro to SQL, essentially it's moving everything, but we didn't want to blindly move everything, we were going through what we thought of it, took the opportunity to rationalize. And so, we created an inventory, essentially, a data asset inventory. And initially we did that in Excel as would many people do, in Microsoft Excel spreadsheets. List of tables and then try to establish, what the description of this table because it'd be a table called, plants status.

What is this? And the company was small enough. And we actually at the time had about a thousand tables, which seemed a lot. So, I and some others created this inventory of data assets and it included, as I said, the physical and some business layer descriptions. What is this table? Who owns it? What maintains it? And out of the efficiency, it was like, well, I now have in this Excel spreadsheet, all the tables and the columns and they're required missing nullable. If I use that metadata to generate scripts, the standard alter table create table, that makes my job easier. So, we birthed an application, which is a scripting engine that took reading from metadata. It generated the appropriate physical constructs in the SQL server. And that was great, because for the first time, at least in our experience, the documentation got turned on its head. Documentation often occurs last in a process.

If you've produced the business value, the product is out there, now you want to write the documentation and it's often times omitted and arguments can be made what's worse, no documentation or bad documentation. But in this case, what we've done essentially is flipped that on its head and the metadata that describe what data should exist was no longer simply documentation. It was actually a specification for implementation. And as such, it had to come first. So that critical piece, we described what we had first, and then that generated, that caused to come into the existence the physical aspect of it.

And then we took those Excel spreadsheets which were describing the data and in a similar to ouroboros the snake eating its tail... we described those metadata constructs. So there was a table called tables, that had in it a record named table describing itself. Because we were on essentially a blank piece of paper starting with a SQL re-platforming, we made it such that if something wasn't in the documentation, it ceased to exist. So, it was self-cleansing and self-healing for bilateral. We had scripts that went through and anything that had been created on the physical that didn't exist in the metadata disappeared. And as I said, a critical piece of this, is that... Well, so, that was 2001, the day our metadata started driving the physical implementation. Because we'd re-platformed, we also realized we didn't want to do this again. And so, prior to that date, our applications had been interacting directly with the databases. And so, at the same time, we started developing what we call the middle tier, which is this standards of programming model that map to the physical model.

It insulated the applications from the direct data, taking care of the basic CRUD operations and allowing code portability. Then we have infrastructure as code. And here gets into the chip, I guess I would say between... Architecture is an overloaded term, but between data architecture and software architecture. And the only way to access this data was through this abstraction layer of this middle tier. So, the partnership of the software architects working with that metadata driven, and in fact, similarly, that same, this early day ORM or object-relational mapper, the schema that the code was available to became dynamically updated as we added tables or columns that dynamically appeared in that middle tier available to be queried.

So, that was early 2001. We realized we didn't need to have humans run those scripts. We shifted, we built a tool for us called roll them out, which is an automated schema management utility, essentially push button schema, I think you know. Schema as source control that allowed us versioning and capable and deployment through the various environments. The arguments made over what are the appropriate number of environments to have. The right answer is, as few as you can need. Only as many as you need.

But obviously that varies. And at the time we had three, we're up to seven now. But it allowed us versioning control and schema driven. Then over that period of time, and I guess I'll pass to Mark here to talk about, once we had this, I'll say perfectly curated data architecture, what the software architects could do with that.

Mark Avallone

Thank you. I joined SNL at that 2006 juncture and I was pretty into software architecture, and I'd read about all the patterns and had largely prior to that experience, everything that they've and the folks on the chat are describing with the big squiggly lines where, all you actually really end up getting to do is apply this high fidelity and to your architecture and to your little island of whatever you're sanctioned to right, to Conway's law, the systems look like the organizational structure. So, whatever team you're on and what you're allowed to effect, that's your little sandbox.

So, I had read though about this concept of maybe an enterprise data model and this object relational map or concept was becoming pretty hot. There'd be these logical data objects that you could work with and then those would broker to the SQL, whatever the database would be and give your code some insulation and portability and adaptability. And it sounded really great. And then I come to SNL Financial and come to find that they've been using an ORM for years before people even had coined the term.

And it was of their own creation and there are certainly some aspects that seemed a bit unwieldy. Like you couldn't buy a book and learn how to use it. There was an in-house SDK, so there was a little learning curve for coming in and you couldn't mess with the SQL. You had to programmatically do everything through this SDK. But the company leadership was really bullish about it because they understood the value having operated with this for a few years now they understood what it was buying them as far as the ability to adapt and the ability to manage their code with extreme fidelity across content, ingestion and acquisition systems as well in a product delivery systems.

And I understood it too. So, we looked at it and I think there were just a few things that dawned on us that you don't necessarily even need that rigidness. We figured out ways to keep evolving the tools and APIs and SDKs around the data management system, logical physical data construct. And over time, one of the first introductions that really opened the efficiency was, we were a heavily SQL server shop in a relational database design was the way things worked.

Everyone was arguing, it's the data management system and our logical objects so are they geared towards data ingestion or data retrieval and the answer is really neither. They were really just focusing on eliminating data anomalies, a Boyce- Codd normalization of the data universe, right? So like one giant gold copy of the data wasn't really necessarily good for ingestion or good for delivery. And everyone just worked around it because it was really maintained quality. And that was a cornerstone of our business was to have really good quality data. And so we could, in this middle tier, we could ensure quality checks were executed on every transaction to the database. We could ensure real time derivations were reflected. We could ensure, that that constraints were adhere to that you couldn't necessarily enforce through pure SQL, triggers and what not, or if you could, you probably wouldn't want to.

So that was also incorporated into that middle tier to ensure that the quality of the data was really high fidelity. And that was great on the transaction side, but on their achievable side, it didn't really didn't seem to make a lot of sense. So I was really in product development when I first started and I was really well-versed in T SQL. So he, the fact that I couldn't write reading operations, like I understood the value of, making sure everything transactional went through the Semaphore, that was the middle tier. But I didn't understand necessarily why we couldn't read data. But we still had used the middle tier to read. And so then we, on a little skunkworks project, I did I scripted this database full of views that took the metadata information Hamish was describing and generated just the database full of views that looked like the logical model and then maps to the physical model.

And you use that same metadata to manage that mapping and the way we've managed things with the Boyce-Codd normalization, the fact that was good enough, right? That worked. And then that opened the door to really start, really grow the operation. People coming in knew how to code and T SQL product development could write portable T SQL scripts that read data. Business analysts, product managers, project managers, everyone could really start working with the data fluently, but we still manage the data with the same degree of fidelity and portability and model first development. Now we really just took off from there as far as efficiency gains, where you built up SQL server reporting service type inspired system.

It was again an in-house creation because, I looked at SQL server reporting services and you could have made do with some of what we did with object views. But we really wanted something that was completely flexible and truly dynamic as we were bridging our third re-platforming of our products. Before acquisition, nine years we'd gone through three generations. Complete re-platforming of a product that had well over 1500 reports in it, into completely different architectures, completely different technology stacks, all web delivered but the most recent one is, responsive design single page application, pretty fancy. Well the last one was ASP net web form, and you can a mix of, MVC.

And then the one before that was, ASP net 1.0 . But all three of those generations overlapped for a few years off of the same data model. And we were able to successfully apply if you're familiar with Martin Fowler's strangler fig pattern based on the, the strangler fig tree or vine, which we'll just wrap around a tree, fill it, kill the house, the tree dies inside, but the vine tree show lives on is built in progressed, and moved shell the shell, but maintain the exact, the consistent data model and evolve that data model. And Hamish mentioned that this infrastructure is code. It's an interesting art, our data model has its own CI/CD life cycle. Like it moves independent of the apps.

We, constantly update and release and amend that the data model. And we know, very quickly we know all of the application code that has dependencies on that data model. We know if there is a breaking change potentially to that data model, exactly where to look to assess and to vote and ensure that the code operating on it's not doing something bad, that's going to require some coordination between code and schema, which is very seldom the case. But we, we know that like, we're not surprised by that fact. And so it's been really powerful to have the code and the platforms that we write around it have this amazing contract to adhere to in this, in this guarantee of portability installation. we're running right now on premises and, and AWS and, and Alibaba cloud. So it's, it's pretty amazing.

Dave McComb

We're going to take questions from the audience here in a minute, but I have a whole bunch of my own. But the one, I think there's one or two that I really owe it to everyone to get answered your one that had to do with the title of the talk, which is getting the toothpaste in the tube. A lot of people here are probably jealous that you had this greenfield opportunity that you started from ground zero, and managed to keep it all together by the time I visited you four or five years ago. But then you became part of something bigger that where the toothpaste was out of the tube. And do you want to say anything about the brownfield situation? What did you do once you got acquired, and the architecture got literally got promoted up to the headquarters?

Hamish Brookeman

I guess over that time, we'd started with Microsoft SQL, but very quick and we can adopt to any database platform system we wanted. So, that had extended to include post-grads both RDS and AWS Aurora, Oracle, Cassandra, Elasticsearch, Neptune, only property graph at this point, Snowflake. And we'd layered into that metadata description not just the physical, but also certain semantic or logical business contracts like where should this go? Is it currency convertible and had code that used that? So, there was a lot of benefits too... An example of localize and globalization that part of the data contract, I want to know Bank of America's assets but I want to see it in yen, I want the labels to be in Japanese and that the middle tier or the new reporting, the abstraction layers through which the data was accessed had that centralized feature and functionality.

So, then SNL was acquired. And so what we've done since then, and there's two possible paths for us. And in certain cases we've done one and sometimes we've done the other. The lightweight integration is we describe in the data dictionary or data catalog, whatever it is, think of it as the API or endpoints. The contract that there is a thing right here, let's arbitrarily call it power plants. It has these attributes and our applications can access that. But underneath that, is a black box, either an API or some code. While it is described, it's not fully managed in one sense. We don't know what's under the hood. And so, the other parts of S&P Global, where we're accessing data from, as I said, some brownfields, it's a thin layer of abstraction, but it allows us some enormously powerful capabilities.

We're permanently again, insulated against changes, assuming that contract is already maintained. Everything that comes, that gets displayed on what we call the S&P Global platform, which is loosely this architecture with various delivery systems. In other cases, and it is a minority because it's slower work, we're doing the full integration. Which is, we do the full description of where it's stored, what the physical columns look like, what the physical storage system is. And at that point, it becomes data as code... We know where the data sits. It's fully managed. So, it's allowed us for some flexibility on versus wait five years and then you'll be able to see ratings data inside the market intelligence platform.

And that critical decision framework is sometimes based on, well, how do we decide? Oftentimes it's based on, well, is the thing that the data is coming from now, is it in the future state? For instance, was it just re-platformed? If yes, we're likely to do the fuller integration. If no, let's say it's in an on-premise data center in New Jersey, and we know we're going to have to move at any anyway at some point in the future, we do the lighter description, because going through the effort of making something that has a lifespan is less valuable. Mark, if you want to elaborate or add.

Mark Avallone

Yeah, I think that's pretty much it. I don't want to take up too much time saying the same thing.

Dave McComb

In fact, I got one more question, I'm going to turn it over to Tony for the questions from the audience. A lot of your architecture is model driven, which we love, obviously. Could you say just a word or two about what that is for you guys and maybe some anecdotes or some metrics about the kind of productivity you get from that?

Mark Avallone

In a model driven, we talked a bit about this kind of homegrown object, relational mapper concept and Hamish talked a bunch about documentation becomes a specification, that's exactly right. We have a group that's dedicated that Hamish leads that manages building the models. The models upon building actually become real and workable in the SDKs and APIs so that engineers can build around those, and the engineers aren't actually concerned with the physical topology and the physical resolution of that. Their code is portable, we understand what the state of those models are in the various environments so, if we felt, and we did, that for various reasons that we needed seven environments, instead of three, we can manage the state of all seven of those.

We understand what's where, and in what shape it is, what code will run where, and what code won't run where, and I think it's pretty amazing I would say that, a change management capability that we have, for instance, this idea that, we can make and we constantly just deploy schema changes in the underpinnings data structures, independent of the code, and then we can assess if there is a known breaking change. So, we understand that the change we're making potentially could break code. Or changing a data type, for instance.

We can assess that within hours, whereas often that's, in my previous employers, that was something that required a project, just to assess the ramifications of changes like that. And often you just don't end up making them because it's too risky. So, then you introduce something completely new. You constantly paint yourself in corners and it looks more and more like that squiggly line picture. When the only recourse is to build something new because you don't want to jeopardize, breaking everything, yeah.

Tony Shaw

There's a question about, what's the middle layer built on?

Mark Avallone (44:29):

So, that's an interesting question.There's two generations, but I'll just go to the real important aspect of it. It doesn't matter so much what it's built on. We follow the model that you see in the open source, almost everywhere. You have to build it in something. And that's going to open up the door to other things built in that same tech stack and to have the highest degree of control and capability on it. So, pick .NET, pick Java, pick something, build it in that, and you'll have an SDK that other things built in that

tech stack can work with directly. But then, for everybody else, you also build a restful interface for it. And you can do that in any... Whatever you pick, you can layer on Python, you layer on a restful interface, and then anybody else in the organization, the enterprise can work with it at least with most of the capability.

And that's the model we followed for the last few years with it. And that really keeps those doors to integration, systems integration, third-party integration, the ability to define APIs is the physical resolution of a logical model. It keeps all of that possible.

Tony Shaw

One of the questions is, what are your plans to get from 80% data-centric to a 100% data-centric?

Mark Avallone (46:07):

Well, I'm not sure which questions we missed on Dave's quiz. It was a few years ago. At the time we... an example like where we... I would say, what we don't have... Everyone's looking for it. So, the data federation in one sense, like we haven't solved, if we have data on in an S3 bucket and data. We can query them but it's not all in one place. So, to the end user, performance varies based on what the underlying storage technology is. I guess the other thing I'd add, we have support... The efficiency is here. Yeah, we have an enterprise data architecture team, that it's 17 people. We are globally located so it's around the globe including... But that 17 people, we now have 10,000 , I'm going to call them data objects, but think of them as nouns with over half a million items field including complex data types and other things.

So, it's just 17 people running it, because DBA role is essentially much more of a DBA architect, all of the usual alter table add column. That's entirely automated, it's almost CIC for data, schema. And so, our DBA team is no longer deploying things. We're doing production scheme and deployments. Six days a week, we take Sundays off for backup and restores and things. We're constantly deploying to production which allows us really flexible, deployment patterns. But we do have, and critically it's that support from the gate to get into the platform, which is some from senior leaderships, chief data officer, or chief technology officer, has to be described in that DMS, has to be described in that data dictionary that gate to entry is. There must be in the data catalog, or at the very least that data must be described and that's the price for entry into the platform.

Tony Shaw

There's an important question here. It says, with companies for the most part using COTS applications, how can we wrap this type of architecture around an environment that consists of extremely disparate solutions with different data architectures? And the comment is, seems like if it's not scoped sufficiently large, then is it doomed to fail?

Mark Avallone

Yeah, I think that also, I think answers how to deal with domain-driven design, we did by apps, so that's the beauty. I think you'll see COTS platforms... You want something that can be in the middle, right? So, the data-centric manifesto says, you don't have to have everything actually in the same place physically,

but you do want everything described in one place. And so, when you're starting out in this brownfield or COTS environment, you want to find something or build something or do something to manage it in a central place. I'm not sure if COTS programs that lets you do that necessarily. I'm sure there are some price sponsors in this event. We didn't have luxury of having our own in-house solution for that, but if you can find something that can manage those disparate models across the enterprise and start at least describing them centrally, then you at least are aware of where... It's a capability maturity model going from chaos to whatever the next level above chaos.

You recognize you have a problem and you at least have an inventory of it. And then you can at least start making plans for what a unified rationalized model of that looks like. But that's really what we were thrown into. We took a unified rational model and we got thrown into an environment where it was no longer that, but we had to pick a winner. We have to pick the thing that we want to centralize around. The 800 pound gorilla, that is quote or quote gold. And then we start setting the course for the ship to slowly mold into that, as best we can. And that's a years long things, and sometimes you need reconciliation between two, like things for quite a while. That's where these integrators and data buses and whatnot come into play. But if you don't have a picture of it, a common core picture of it, you'll never know what's even wrong and what needs to be righted.

Tony Shaw

There's a number of other questions I am sure we could dive much deeper into, but unfortunately our time is up for today. Mark and Hamish, Dave, thank you all, I hope you enjoyed this as much as I did. Congratulations to you all for managing to accomplish and sustain this as you have. Obviously, there were some other resources there for everybody to follow up with the data-centric manifesto et cetera. So, thank you everybody, see you again soon.

