TRASCRIPT: TECHNICAL DEBT - MILE HIGH DREAMIN’

So actually this session is called Technical Debt Is Killing Your Org. but if I'd come up with that topic, they probably would have selected me, because it wasn't very upbeat. But it is actually killing your org, and I've got receipts.

Hopefully this will be useful for you to use this when you go back to the workplace to Build the justification for why you need the time, the space, to go and reduce the technical debt in your org. Because that's what this is all about. I know all of you would like to reduce that technical debt if you could. But there are lots of reasons why you don't. And most of that we'll talk about very quickly in terms of some of those approaches. If that makes sense. Okay? Quick show of hands. Who here are admins working for organizations? Fantastic. So the rest of you I assume are consultants? Or ISVs? Okay. So you've all got the same problem.

It's either somebody else's problem you're trying to fix, or it's your problem you're trying to fix. So

So Forrester came up with this idea of Salesforce at scale dilemma. I don't want you to read all the words. They are the words that are important. Innovation slows, flexibility evaporates, every change is killing your org. Now this was back in 2017. I suggest it's not got any better. And it's this idea that actually the more you use Salesforce, the more the users need, the more you configure it, the more you add, the more difficult it gets to change, the more risky it gets to change.

And eventually you build it up and up and up. And, I haven't got the data for this, but empirically it feels like after seven years people are about to rent and throw away their org. Okay? That doesn't have to happen, but people kind of get to that point of actually there's so much there, let's just kill it and start again.

Which is a massive waste of investment. And often it's because you don't know what's in there. You're about to merge two orgs. You've no idea what's in the two of them, therefore you're going to throw one of those, throw them away and start again. But we had, one customer, they wanted to merge two orgs.

We synchronized it, we connected it. One of the orgs had, the Opportunity Objects account, and they had 256 record types. Okay? For those of you who are admins, I'm not. But there were over 100 picklist fields, many of them had hundreds of picklist values. So, some of you know, you have to allocate every single picklist value to a record site. There are over 5 million mappings. Okay? How many of those record sites had data? One.

I have no idea why they did that, I wasn't there at the beginning, I was only, we were only there to see sort of the results. They wanted to merge it with another org, which is very simple. We went like, you're trying to merge War and Peace and Dilbert, don't do that. You need to simplify them first.

But they had no idea. And I think that's part of the problem, is actually we have no idea how complex the org's got. You only see the bit you've got, and therefore it's very difficult to build a case to improve it or a case to change it. I said I brought receipts. One of the things we do is we synchronize orgs and analyze the metadata.

We synchronize about 50, 000 orgs a month. We analyze about 1. 3 billion metadata items a month. So this is a subset of that, and we pulled it out. We didn't pull out the bad ones, we actually pulled out a subset to get the data.

And 51. percent of all custom objects never get used. That's not counting managed packages. The ones that you've got feedback, you've had meetings to argue about what you're going to call it, you've had Slack messages about where we're going to put it, you've built it, you've tested it fully, you haven't documented it. But all of that effort is wasted.

And if we look at, say, just fields. 41 percent of custom fields on custom objects never get populated. Now I realise some fields never get populated because that's valid. We strip those out. So these again are the ones that you decided you wanted to go and add. 43 percent of standard objects. It actually is a lot worse than that, because there are so many standard objects. There are hundreds of standard objects.

If you think about the core ones, it's about 80 percent when you start getting to some of the, those core objects. And, yeah, these are, these are horrible numbers, and I'm not doing this just because it's a stand up comedy routine. I think there's actually something deeper underneath this, which is the implication or the cost associated with it, because that's what we need to use to build the case for reducing it. So, you look at event, 80 percent of the field is never used. Order quotes and so on. So the, the numbers are really high.

The other bit of data that came out of was actually how many fields you have on page layouts. This is average. Okay? So some of the, everyone in the room is better than this, and it's everybody else. But 150 fields on the opportunity page layout. If your page layout looks like a CVS receipt, your end users are going to do one thing, which is they're going to hit save and go, What's red? What do I have to select to get out of this page? Yes?

The reason there's a big 7 there, the UX experts, I'm not one of them, but the UX experts said that 7 fields is the optimum number so you don't get cognitive overload.

We've now got features like dynamic forms. There are ways we can try and simplify it for our end users. But again, the cost associated with these sorts of page layouts is, yeah, confuse the users. How much time do they waste? But it's also, how much poor data is being pumped in there?

There's a link, so at the very back of this deck is a link to a whole bunch of resources. So the things that I'm talking about, you can go and find those links. So things like the research series, where all that data's come from.

The UX, the technical deck, those levels. So you've got some evidence that you can take so you can build your business case for how you're going to fix this. Is this making sense? It's not meant to be me shouting at you, if people have got questions, please don't leave them to the end, we can pick them up at any point.

That, I think, is what the technical deficit is. You guys start coding, I'll find out what they want. I know it's not as bad as that, but again, every joke is taking things to an extreme. But the point is, a lot of the time, we don't really bottom out exactly what the requirement is. There's an implementation lifecycle.

These may not be your words, but I'm sure you can relate to these activities. At the top there, capture and validate requirements. Create a user story, assess what the change impact is, configure and build, test and deploy, deliver it into production. Yes? All of you go around that change cycle. I think the problem is that, that's the tech tech machine.

You go from a rough idea of requirements straight into configure and build. And that's not because you're bad people. It's because you're not given the time to ask the questions. someone came back from Dreamforce, watched a video and said, this can be done in two minutes, why can't we just do it in two minutes?

And you don't have the evidence to go, I've got to do decent analysis. I've got to make sure that this is really the field you need before I add it to another page layout and to all those permission sets and, and, and, and, and, and all the things which you then have to go down and take. So we need to buy the time to be able to do, capture and validate the clients properly, which is things like map a business process.

If someone says, I really need this, if you get, you either go back to business process you've mapped, or you say, let's map out a business process, and let's see how you want this field to work, one of three things will happen. They'll either go, Oh, we already do what I needed. Fantastic. No changes required.

It was training. Number two is yes. I understand what you need, but you don't need a field. You need a record type, or you need a something else, and you need a something else. We've understood, and by, if you've required that, you also need a report. Or, actually, they have asked for what they really need, but there are other things that are associated with it.

So, building a process map, and also then an entity relationship diagram, so you can start to connect this. Those are tools that you can use to say no by saying yes. Yes, let me help you. Let's sit down and draw a process and understand how you want Salesforce to work based on your business process and the things that have changed.

And again, these are disciplines, if anyone's done the business analysis certification, who has done that? Okay. Everybody else, I would suggest you do it. Even if you don't take the cert, go and look at some of the training associated with it, because the skills you need to do this stuff are all the BA skills.

It's saying yes by saying no. It's understanding how to facilitate. It's okay. Tell me how to map this process. Tell me what happens next. Show me where on this process diagram Salesforce hurts you. And how do we change the after stories. And only once you've validated the requirements, then you should be creating user stories.

Then you've got something which is a piece of work, and then you can understand the impact of the changes. Then you can understand, okay, this user story is going to be really easy. We'll group this together with other easy user stories, we get this delivered really quickly. This is horrible and complex, we need to take it through a different pipeline with Integration test, system test, and all the way through to deployment.

This will take longer, and this is why. Because I've got some evidence, I've got some examples about, this is really complicated. Stage, field, and opportunity. Easy or hard? Hard. Correct. In most organizations, you've got picklists, which kick off flows, which kick off Apex classes. Stuff happens. Yeah? Oh, it's only a field, it can't be that hard.

And that's what we're fighting against, I think, all the time in terms of orgs, which is going, It can't be that hard. I've watched a video. You could just change that. And we've got to, we need the evidence to fight against that.

So that was the, that was the down bit. Let's, let's do the positive bit. That's about the sort of things I think we need to use, the tools we can use to start to get a handle on technical, technical get, and get ahead of it. Okay? So I said three steps. Actually, it's three steps and three demands. The first step is you need to try and quantify it.

You need to understand how bad it is in the areas that are important. If you've got one of those custom objects that's never been used, why fix it? It's never been used, let's go and fix it. Let's worry about the stuff that needs to be fixed and understand how complicated it is. So, question. Who here has got a metadata dictionary?

Oh, that's not very good, is it? Okay. So, by a metadata dictionary, I mean a way of managing and understanding all your metadata. Okay. Let me take your different analogy. Who here's got an opportunity object? Well, that's strange. Why do you have an opportunity object? Because you want to understand how you manage your sales pipeline and you understand, if the sales aren't being closed, you've got a way of tracking them.

That's all a metadata dictionary is, a part around your change cycle. If you think about it like that, let's ask the question again. Who would like a metadata dictionary then? Okay, got it.

Okay, so let me just,   let me show you what one of those things looks like. There are, there are a number of companies who create them.

Ourselves, Ernst Stock Cloud, Metazoa, Sonar, Pania. There are a few others, but if you look for Metadata Dictionary, there is a Metadata API, which allows you to pull all this Salesforce and organize it as a structure, which is easy to manage, easy to then start to do things like dependencies, which is where you can start to understand the technical bits.

It's a place where you can document things, where you can start to list things. Excel spreadsheets are not metadata dictionaries. Okay? Well, I guess they are, but the problem is your world is changing so fast and there's so much data in there. We're looking at tens if not hundreds of thousands of metadata items that are coming out.

And therefore, at least understanding how to manage that is your first step to understanding how to manage technical debt.

So that's what one could look like.   you've got to. Objects, and then underneath objects, you've got custom metadata, and it's basically a tree structure. So you need some sort of structure here. You've got all these custom objects, custom objects have got fields, fields have got,   thickness values. So all of the metadata, whether it's Apex classes or flows, or all of your process builder workflows, or all your permission sets and your profiles, all of those things are metadata which can be stuck into a dictionary, and then a lot of the stuff on the right hand side is metadata.

You can either analyze and interpret, as in, field analysis, or you can, it will come straight off the API. This will give you a far better view of what the issues are than trying to wade through setup. That's, you, you, you don't stand a chance trying to wade through setup. And then build a spreadsheet, and by the time you finish building it, it's now out of date.

Okay? That's why SSL's built the Metadata API. Why they didn't build a product like us on it, no idea. Happy days, we're building that. But, it's something which is, if you're building an enterprise application, this stuff is critical. Doesn't have to be us, has to be somebody else. But what you can do once you've got that, is you can then build a dependency tree.

So you can say the field stage is connected to 17 flows, and those 17 flows are connected to, and connected to, and connected to. This is the piece of, this is the picture when someone goes, it can't be that hard to change. You say, there are 398 dependencies. We've got two options. We can do the, make a change, or you can give me the time to analyze it properly, and I can say which of those, I don't know, 17 flows actually need to be changed if you want me to go and change that, that picklist value.

This is the heart, I think, of the technical debt piece, which is understanding the cost or the implications of it. Blank faces or lots of nods? Okay. So now we need to understand how we actually cost that. So let's get, let's get to the next thing. If you've got all this metadata, AI can help you. Okay? I think AI's had, it's not had a bad rap, but actually, we're looking in the wrong place to make sure AI really works.

So what does AI need? It needs really good data. Metadata is really good, it's 100 percent accurate. You may not like what it says, but it is 100 percent representation of your org. Okay? AI's not very good at coming out with 100 percent accurate results. But you kind of don't need really good results. This is like an intern.

Just go back and tell me where I should focus. Okay? Think of AI as the intern. You're not expecting the intern to come back and go, I've done all the work, the art is completely perfect, you don't need to worry about it. AI is an intern where you look at it and go, thank you, you've given me a steer, maybe I didn't ask the question properly, let me ask the question a different way and come back with some different results.

Okay, now I've got a good understanding about what we need to do. So we've applied AI to some of this metadata. So, go. Don't worry about the pretty graph, you'll understand what I did. I, I, one of the, one of the prompts, again, it's one of the links in the resource you'll find later. It's a prompt you could use.

By the way, the prompt's about that long, OK? It's not three lines, it's fairly in depth. What we asked it to do is say, go and have a look at every field in an object X, look at all the dependencies on that field, all the things which are related, and then using this little table, based on the type of thing that's dependent, So if a validation rule is connected, give it one point.

If it's a report, give it zero points. If it's a flow, give it three points. And add those up, for every dependency. At the same time, also calculate how long we think it takes to analyze all those things. So we'll say, validation rule takes two minutes, flow takes ten minutes. And just build all those up, and add them all up.

And then, group them by how well populated those fields are. Okay? How long would an intern take to do that? Days. It, it, they actually, they wouldn't, they probably quit before they actually got to the end of the job. But AI does a really good job. Is it, is it right? Pretty much. But the point that's good enough that tells me the value field is a monster.

It's got a score of 175. If I look at it, it probably says it takes three hours to analyze versus down at this end. So if I'm worried about changing stuff, this is the the scary end. Down here, here are the fields that don't seem to be used. These are the ones I might need to delete. By the way, you shouldn't just delete empty fields.

There can be at least five reasons why you shouldn't delete a field because it has no data. You need to understand, is it connected to things? You need to know why that got created. But again, if you're looking for fields to delete, it's at this end. So again, your intern, AI, using all this data has given you a good place to go, this is where I should focus.

This is the stuff that's killing us. I need to do some more technical debt reduction. I need to think about better descriptions. I need to do some other things around here. But look, it's going to take me quite a lot of effort. If you want me to do anything, this is the stage field. Okay? Horrible. But again, it's giving you some ammunition to go back and go, This is, it's going to take me longer than ten minutes to go and fix this.

If you want me to actually do any work around the opportunity objects, around these fields, You need to, I need to buy the time to do this properly. Okay?   Bosch. Big customer of ours. he did a,   the product manager of Bosch, four years into an implementation, they're re implementing Salesforce. They kind of started in the wrong direction and they want to have another go at it.

But it's a great video because he talks about the core principles about how they're going to implement. But relevant to this conversation, he said, we allocate 10 percent of every sprint to tech debt reduction. Okay? I think Salesforce talk about a similar sort of number, okay? But again, it's trying to build the time to go and do this stuff properly.

So that second point is you need to try and assess the cost. So you can then build a case to go, I need to fix this, it's going to take this long, this is going to slow us down in the future if we don't do it. If I go back to the data cloud conversation, if you've got that level of technical debt, and you haven't got a metadata dictionary, and you haven't got a decent implementation process that's rigorous, you're Do not start Data Cloud.

If you've learned nothing else today, that's that one sentence that will be helpful. Despite what the AE is telling you. But the point is, Data Cloud is achievable if you have the right disciplines in place, and you have the right approach in place, and you are, you've got the management support to do it.

We've implemented internally, we've got some great results out of it, but we have those things in place. Don't go into this without actually having the sort of things in place to make sure you can deliver it. The Data Cloud Certification, not the topic here now, but the Data Cloud Certification is literally 10 percent of the project.

10%. 90 percent of the project is everything else. Planning, cross collaboration, getting data out, data modeling, data analysis, data volumetrics, building a business case, designing, architecture diagrams, metadata dictionaries, and once you've done all that, build a user story, and then it's really simple. Take every one of those actions and build it out in Data Cloud.

We've done it. But all of those other things have to happen first, because if you don't get them right, you can't unpick it. Am I scaring enough people? Don't be scared, just be aware of what it actually takes to implement Data Cloud. And then go in with your eyes open, and with the right management support behind you, which is what this is all about.

I talked about the cost of technical debt. Just a little concept here. Think about different levels of technical debt. Credit card levels and interest rates. bank loan, mortgage, or parent loan. Again, you need to allocate, again, what sort of technical debt is it? If, if we're dealing with a stage field which is getting changed all the time and it's mission critical, I suggest that it's credit card level of interest, it's costing us money, every time we make a change, stuff breaks, or it takes us 5x the time it ought to go and do the analysis.

We've got to fix this now, build this into this sprint. Bank loan interest, It's not going to affect us now, but we know we ought to fix it if we bump into it. But we're not going to launch a project now. This stuff, they're not changing, it doesn't work. I'm not going to go back and fix technical debt in process builder workflows.

Maybe you need to, maybe a bit of documentation, but you're not going to go back and refactor them. But, when you look at them, if you've got a project, you might go actually, absolutely, I'm going to fix a process builder workflow because I need to refactor Salesforce forces technical debt on us three times a year.

It's not always our fault. It wasn't because we took shortcuts. Sometimes the business moves on. Sometimes Salesforce implements something which we've already built and we need to go and change that. It's not always us taking shortcuts or us not knowing. And therefore you need to try and assess, okay, how expensive is this to us and how much do we want to, and how important is it to change it?

Not all technical debt is equal.

But we've got to stop the bleeding. We've We've got to get an implementation life cycle where we're not putting more technical debt in. We're not loading more stuff in on top of the old stuff. We've got to try and break this pattern. Otherwise, we'll all be in the room here, two years time, going, You know, it's horrible, isn't it?

Yeah, I know. But, you need to try and get ahead of this. And the way to do it is you need to get a decent implementation life cycle. So, do you remember I talked about that? We need to actually go around that. Properly. So who here has got a documented implementation lifecycle? If I walked into your org tomorrow, and you said, right, you're the new person, the new admin, and you said, this is the process we follow, how many of those?

Ooh, some of you. Fantastic. Good. Okay, so if you're sitting in the room and you've got one of those, what's the problem? You're not able to follow it? Or you're inheriting things from previous orgs? Inheriting? Inheriting over time, okay. So you've got a process where at least you're now starting to try and fix some of that.

Good, okay. So I think this is where it starts, which is, this stuff we can do, I get that. The problem's at the top layer, which is buying enough time to do the, the five whys, to ask the questions, the softer skills. I know when there's someone sitting here who goes, I have to have this field. You've got to be able to go, yeah, yeah, of course.

Let's just understand how we might implement that. Or, I'll pop, you can have the field, but I won't put it on a picture.

Technical debt you can't see doesn't hurt you necessarily. I mean, so back to the building, back to building the cost picture. Don't think of the cost of your cost necessarily. Because you're really cheap. Compared with, sorry Matt, you are really cheap. But relative, relative, your day rate relative to the cost of your day.

No, 10 sales reps or 400 service agents who are struggling through pages and putting in bad data. That's, I think, where we should be looking in terms of where you're going to find that cost benefit. Simple spreadsheet. This page layout takes a rep 10 minutes to go through or 2 minutes to go through. I then have to spend 5 minutes fixing all the data at the back end.

And we've got 100 of those and they're doing that 3 times a week. The cost of that is, and by the way, Don't have any aspirations about doing data cloud and AI, because we're not ready. That is the true cost of technical debt. Not the fact that you're going to take a little bit longer to do stuff. I know, I know it's painful for you, and I know there is a cost associated with that, but there are way bigger costs that you can look for.

technical debt stuff, as in poor data, ends up in dashboards, so your senior execs are making bad decisions. Massive cost associated with that, and they understand that. So we need to frame the cost of technical debt in numbers that they will understand, so you can build a business case to go, see, see, that's why I need another 10 minutes, or another two days to do this properly.

And, and my suggestion is, if this is new to you, pick a really low risk, low political case. Don't go after, like, quote to cash, as the first one. Pick some places which are small, where you can prove the benefits, measure what it was worth before. Measure it afterwards and go, look, this is what we say, now let me apply that again and again to bigger and bigger things.

I think this is, this is a lot more, is more political than anything else. It's man, it, it, trying to navigate you around this. The technical side of it, I think, is not that difficult. I think the difficult bit is having the time to do the technical bit. Yes?   the other thing is, actually, documentation in all this.

You generate quite a lot of technical debt due to documentation. A lack of documentation. Let me explain what I mean by that. You created a field called Industry. I don't understand how you use it. This team over here wants a field called Industry. So they go, I'm not sure how I use that. We'll call it Industry 1.

Okay? Yes. You know where this is going, don't you? Because you didn't write down what that field was used, not for, but why they created it. And I'll pick that up in a moment. And we've got situations that we think Opportunity Object, Opportunity Object 1, Opportunity Object 2. Okay. Think about trying to do a forecast roll up over three custom objects.

Just because we ran out of fields on opportunity. And then we ran out of fields on opportunity object one. The, the thing about descriptions, about why that's a documentation issue, and again, it's part of this, which is somewhere around here, and up there, you need to start documenting the changes you make.

You need to document why the field exists. Why this flow works, not what it does. You've got a field called, I don't know, NDA. And you type in the description field, it's the NDA field. I kind of got that. It's in the label, it's on the page, yeah. Why have we got an NDA field? We've got it because finance signed a contract, we've got it because marketing cannot use this customer without reading the NDA in any of our marketing.

That's the why. I'd suggest lots of your fields, if you've got documentation, have got what, not why. Which isn't very helpful. And if it's a what, it's not very helpful. I kind of understand that, so I'm just going to, I don't, it doesn't help me, I'll go and create another one.   again, back to the list of things in the resources.

We wrote a prompt for you to evaluate all your descriptions. So I went and asked OpenAI, what sentence would be a why versus a what, and it gave me a bunch of criteria. You can run it against all your descriptions, it will tell you whether it's a why or what, or whether it's blank. But, it can also tell you which are the most important fields, based on the page layout.

So why do you focus? I've got a bunch of fields here. Let's focus on the ones that have got picklist values, the ones that have got validation rules, the ones that matter if they get it wrong. So again, it's that intern just going, oh, okay, yeah, I can help you, I can look at all the descriptions, I can tell you which ones are most important, now I know where to focus in terms of improving my health.

That then also helps improve data quality, and it rolls on. So these are, I think these are all relatively simple things you can do, just don't bite off too, too big a thing. And then try and implement some of these principles around this cycle just once, and go, look, measure the beginning, measure it again, wow, this works, look at the success we got, right, give me more time, give me more time for the next one.

So you can then start to estimate how long it really takes to make a change. Because most people say a change takes three months, because you've, you know, You've got no measure of how long a change takes, so you have to say three months, because the longest one takes three months. If you could then go, this is a short change, I can get this in next week, this is going to take three months.

And then bring the receipts, this is why, then you've got a reasonable case for going, okay, give me the time to do this properly. We went through this cycle for one of our changes internally, obviously we use Salesforce internally. We did all the analysis, the sales team said they wanted these things, we went through.

We validated the requirements, we looked at the process map, we looked at the ERD, we wrote user stories, we evaluated and said, this will take three people six weeks. And the sales team said, we don't want it then. Do other stuff. Do stuff that's more important to us. So we didn't build it. And you go, what a waste of analysis time.

No, you're looking the wrong way. That was a really good way of saving a bunch of development time, which wasn't the right priority. The moment you have those conversations, you're at a different level of sophistication. Do you still want to do this? Now I've understood how long it takes. Rather than, you said you wanted it, I put it in, why are you not using it?

Because there were more important things to do. So there's a, I think there's a level of just reframing the way we go around this. You're going to spend a lot more time on that bit than that bit. Drag and drop's easy. And I think what's happening is, Data Cloud is going to, I keep on going on about it, but Data Cloud is going to force this into organisations.

Data Cloud is how we should have implemented. The disciplines that Data Cloud have forced upon us are the ones we should have applied to implementing Salesforce. We've just, it's just let us take shortcuts. And we're now paying the price.   internally, we implemented, if I can get the numbers right, 377 releases this year.

In a calendar year, inside our own org. How many rollbacks? Zero. Not me. They follow the process. And every time they release, they go back and go, could we improve that? Could we improve that? Could we improve that? If it's not written down, you can't improve anything because you've got nothing to build on.

People say to me, what's the first process I should map in? This one. It's yours. You own it. You don't have to ask sales about their process. Do this one. Get this one right. And go around that cycle and constantly improve it, constantly improve it, constantly improve it. And then go, that didn't work. Why?

Let's change that. Okay. And go around red sliders like that. It's not hard. Wait, it kind of, no. It's easy because it's possible to do. I don't suggest it's easy in your organizations with the politics and everything that goes around that. What I'm trying to do is give you some of the evidence of the angles that you can take to be able to make this achievable for you.

And it's going to be hard to start with because there will be a lot of doubt. This isn't going to work and why am I wasting time? I don't know. And just build it. But pick some things where there's less pressure on that. Don't pick CPQ. What does CPD stand for? Causes people to quit.

So on that note, I said there's a bunch of resources there. There's a QR code,   the reports, the Bosch video, some stuff around metadata, some things about metadata dictionaries.   a whole bunch of stuff. And then, oh, all the AI prompts I talked about. All there. About five or so minutes for questions.   tech debt is killing you all.

It doesn't have to.   I think with some of the principles here, you can build a business case to start to make some changes.   I'm applauding you on that journey. Happy to connect on LinkedIn, connect through email. and I'm putting blogs out all the time around this sort of content. But if there's stuff you go, I really need a bit of content like that.

I understand you could have said it and I'm a stranger with a briefcase and they'll believe me and they won't believe you. Happy to write that, so you've got that evidence. So thank you.