**Workshop with Cognizant - Discussion on FSS SDLC Methodologies & Eng Practices - Placeholder - Meeting 2-20231004\_143234-Meeting Recording**

0:02  
You want a Mahindra to share? Anything to kick off,

0:05  
Um,

0:08  
if you can, if you can just show that first slide that you had with like the 16 different topics or whatever and we can just hone in on the the the first ones gonna try and cover today. Yeah. So Dave, this will probably end up being a couple of sessions. Yeah. Your Karen can share screen so that I can start up with the top. Yeah. When you got that open. Thanks. Yeah. Yeah. Alright. Thanks.

0:32  
Uh. It's probably a couple of sessions, Dave. Umm, so uh, I think I've mentioned to you what we try to do is talk about

0:41  
OHH, we've got loads of systems and FSS. We've got loads of different ways. They're all slightly different. The, you know the the stack is very different for them. Our ability to automate things is very different with them. We've got a set one end of the spectrum which is brand new generally probably doing things

0:59  
best practise in the industry these days. You've got things like T20 four maybe Magara probably at the slight the other end of the spectrum and he got opening which allows a third party product is completely,

1:11  
you know we completely go off the rails with how we do it. So it's totally down to us and we've moved forward a little bit with T20 four. So it felt like a pretty good sort of

1:22  
counterpoint to the A stuff. And so we're gonna go through today, the guys will have, uh, a few questions around some of these things. And whereas we spent a lot of time talking about the Menace for a year, it's going to be pretty quick because, uh, you don't do it, but we don't have it. So some of them will be quicker than others.

1:40  
That's the kind of summary of where we are with these sessions.

1:46  
It's our first summary, guys. Is there anything else you want to throw in before we get stuck into somebody's so, so Mahindra, are we gonna be focusing on these blue boxes here or the SDLC methodologies? I don't know how you wanna plan this session.

2:01  
Yeah, yeah. So we covered around um, UH-8, UH topics like principles and patterns around

2:08  
and we will be targeting the remaining topics.

2:11  
OK. So I I think last time uh, Sri and uh, you Paul, uh has given some insights over the application, right.

2:20  
Yep. So we talked, we were talking about ACE last time. This is a different

2:24  
system now. This is this is T24 um, but some of those, some of those conversations we've had around these are are the same. So we've talked about dev methodologies. Hmm. We don't need to go over that again.

2:39  
It's the same principle for for 24. So Dave, sorry to speak for you Dave, but um, so Dave runs A-Team and that team supports and develops the 24 application.

2:55  
Our development methodologies are from project perspective are now pushing towards agile hybrid ways of working which we touched on and we'll get into those sessions a little bit more next week where my my colleague is back from leave

3:09  
UM and in the past everything's been very much traditionally waterfall. We also talked last week about

3:16  
the difference between projects and small change and that's the same here. Our small change is typically waterfall. There's a there's a need for something someone might suspect. There's a, there's business requirements. It comes to Dave's team. We estimate there's a dev phase, there's a test phase because to UAT, but that's no different to what she and I talked about last week.

3:41  
Um, so which topic do you want to dive into today?

3:47  
So basically Paul like we I, we discussed, we got some points around in the first. So basically we're going to the horizontal sequence right team structure, methodology, tooling and technologies. We even requested some documents and artefacts related to it they haven't received. So we could not zero in on on that. But whatever the inputs we gathered, we generated minutes of meeting and we have you know observed a certain areas and points. So if if you really wanted

4:17  
so if some areas have similar methodology for this new team, then we don't have to touch upon that area. But UH, if other areas if for example if you say right some they're following us still a hybrid or a waterfall model. So if you can mention I can take it down and we can move it on the next areas. So it's a team. Team structure is very similar to what we've been through this. So

4:40  
they've just given an overview of numbers of your team and roles. Um, just for the guys, just so we're clear. Yeah, sure. So we've got in all five people on the team.

4:52  
We've got a specialist on more of the infrastructure side. Who's Kirk? We've got Kavitha who's more

5:05  
business and development orientated, uh another guy Manikandan has similar to Kavitha and then Subhiksha again similar to Kavitha as well in that they all do business side of it, UATBU and development. And then I've also got another guy who's we're training up at the moment. So he's just learning the ropes at the minute. But it's quite, quite a small team but there's a lot of

5:36  
technical and business experience spread across them

5:40  
right. Thanks Dave. And and that that seem that team just like the the Ice Cars is is kind of a DevOps. Yep. So so they do development of the product, but they also do support for the live operational side of things.

5:56  
No, yes, Paul. But uh if is it OK if we can share screen and show that UH team structure and composition so that we can get some idea

6:05  
that there isn't really anything to show other than I'll just have Dave at the top and beneath it you'll have for individuals. So there isn't really anything to show to them with this, Not it's not worth drawing that out.

6:15  
Um,

6:17  
such team structure,

6:19  
dev methodologies I think we've touched on. So I'd say projects that come through the domain are pushing to do agile, sort of agile, hybrid ways of working. But the small change that they've referred to is still traditional waterfall. So that's the same answer that you got when we went through this with with Shree

6:37  
tooling and technologies. Um,

6:43  
Dave, I'll let you.

6:45  
I'll let you come in. So sort of sort of

6:48  
you can take over when I run out of steam because I won't remember what's changed since Michael. But so 24 is a, it's a third party product company called Temenos

6:59  
Um, but we do local development on that product. So there's the, the core product that comes from the vendor and on top of that, when it comes to us, there's a degree of customization and local development that our developers do in house.

7:16  
Uh,

7:18  
From a from a technology perspective, uh, it's a Linux stack. I think we're running on, uh, Red Hat

7:25  
7 something 797

7:30  
it is. It doesn't have separate databases, so it's not like an Oracle or SQL or anything. It's a it's a proprietary database that's embedded within the application called Jboss. That's still true. Day but J Base sorry, sorry. J courses,

7:47  
um

7:48  
so the skills needed to develop in that are very niche and very specific to that product. So as Dave said, the team, the experience of the team are combination of people who've been at the bank for a long time and have built up a lot of experience on the product and the business.

8:04  
And people actually who are ex Temenos people who now work for us as Bank of England staff who obviously come with

8:14  
a lot of application technical knowledge. Um, and I'll learn in the business side of things.

8:20  
Tooling wise, I don't think there's an awful lot of tooling involved, but I'll, I will let Dave just run through what sort of tooling we use on the product these days. That's alright though.

8:31  
Yeah, sure. So

8:33  
by tooling you mean what exactly? What sort of things

8:40  
would you consider covered by tooling? Because at the moment we would

8:45  
basically programme from the what we we we refer to as classic which is like a a Patty emulator. All of the developments done in that that's then packaged up into something called toolbox which we use as our basically our our release mechanism

9:08  
and that's it. There's there's nothing else that we would use while we're doing developments within 24 UM. From the front end side, there are various

9:23  
applications and we can tailor those as in

9:27  
inquiries, um screens, reports, uh, there's what we would configure

9:37  
files there as well, so we can get the system to do different things. By the way, we configure these files, again, they're packaged up using toolbox. That's basically the only two ways of developing anything and the only way of releasing anything. So we would package it up in our dev environment, release it to test environment, then release it to UAT and then release it to live. That's it. It's very, very simple in terms of tooling

10:07  
technologies, as Gifts said. We sit on Linux, so you know there are there are some Linux scripts that we do that we use in routines.

10:18  
The browser front end obviously. Uh, we've got Jboss

10:23  
as part of our stack, we've got web servers as part of our stack, and obviously we've got the browser front end and that's basically it.

10:33  
So I'm not overly complicated,

10:37  
David. I've got a quick question on that, which I'm pretty sure it hasn't moved on too much. But one of the things that we come on to and here are things like, um, branch and inversion, code quality and code review. So we're talking about sort of

10:50  
you know,

10:52  
source control, all that type of stuff, but how do we, how do we deal with that in 24? So at the moment, because because of the way the system set up, the

11:04  
vendor solution for source control doesn't fit onto our platform. We have a

11:14  
one for better word, a library of programmes which we will

11:19  
store and then when we come to make changes we will transfer that piece of code out into J base, make the change,

11:28  
once it's changed, transfer it back in. So we have a central repository for that. Obviously that's used more for small change rather than projects work. Projects work is slightly different where the

11:43  
the changes are made and they're not put into that repository. Then once the project goes live,

11:49  
post programmes will be refreshed so that when we have a live change to make we will use whatever's come from that project.

11:58  
And then once we hopefully upgrade to a newer version we'll be able to use the vendor supplied source control. But at the moment we've got an in house development one.

12:11  
It's time.

12:14  
OK. Goes back to you, for him for any any questions or avenues you wanna go down next.

12:21  
Before we start on the questions, Paul, I think we discussed the last couple of sessions with Sri's team, right?

12:29  
Yeah. So why are we now changing the team? Because it would be good to get the remaining practises, for example, like security and knowledge sharing from the same team, right. So it'll be a conflict of interest rate. What do you feel? Yes, that's that's why I asked if we want to move on to 24 for this session or race and the response was to move to 24

12:49  
if if, if you wanna stick with ice,

12:51  
we can, we can do that, but we'll need to get 3 back. He's not available at the moment. So this is about moving on to the next product. If you don't want to move on to another product, if you want to just stick to embrace, fine, I just need to know.

13:05  
But the response was to move on to 24, which is what we've done. Yeah. Because I I could see from the way David explained, uh, even the tooling and technology stack, everything is a bit different And they they were constructing on the back end. He is more of a front end right here. Yep. Yeah. Well that that's what I thought. OK, but but, but, but there is, yeah, there is a front end for some proposed. Correct. Yeah, yeah, yeah. So but like I mentioned explained, right. Paul, if we can share the screen like show the branching diagrams

13:36  
that will be a little bit helpful in in because we were came prepared in particular way because we went through the minutes of meeting and we generated some inputs based on that. Now since we are dealing with a new team, we need to have a bit of understanding before we start on with that, right. We can ask some generic questions, but it would be more helpful if if Dave, Dave School or Dave Jones can share anything if, if possible.

14:03  
Though I doubt there's anything to share on screen that shows you like the branch and stuff because we don't. Dave doesn't have that on his team for 24, OK, that's that's the standard way. The way they do it is ultimately having different dev instances stood up

14:19  
with different iterations of what they're working on at the time, and when they come to complete their work, there's a there's a pool and of it back into to to the master. But I don't think you'll see that anyway, because it's none of it's automated. It's all just it'll just manually done. I think that's right, isn't it though? Yeah, correct. Yeah, OK. OK.

14:37  
OK, fine. Uh then uh, so maybe I'll ask the questions from UH whatever we discussed here since it see basically we are doing this assessment to have some insights on what is being UH being followed in BOE and what can be done better, right Paul. So in such a way I'll try to follow up with the my questions on what we've been discussed with Srishti. So he mentioned like he follows a master develop and feature branching strategy. So the so Dave's 24 team also following the same way

15:07  
for rantings.

15:11  
I don't understand what you mean by branching. We don't we don't really have branching for T20 four is what we're trying to say I think. So this product maintain full

15:21  
sorry yeah because the source code or maintenance source code

15:25  
ohh sorry I can't hear you sorry Paul said something just as you would say in that first part about source code. Yeah, source code where the source code, it is made in the gate or any any other source control mechanism.

15:38  
No, it's so right. So when we the way it works is we have I think now about six or seven development environments. Now when we do a piece of work outside of a project, what we do is we take a cut of live and we restore that to a dev environment. OK. So that's our most up-to-date environment date wise and code wise. Then as I mentioned before, we've got a code repository,

16:09  
so we will book out so that somebody, so that people working on other dev environments knows that that programme is being worked on. OK, That then gets developed in that dev environment and packaged up. OK. Once everything's been developed, tested and finally released alive, then that programme unlocked. So it's the latest piece of code that we've got. Yeah. So that's the only source control that we have. Have It's quite a manual

16:39  
process for us to do it, but because of, as I said earlier, the way that our system is set up, it's very difficult to integrate a source control tool for us. So we've created our own one in order to have that repository of most up-to-date code.

17:00  
As I say with projects, it's different because they will be working on a separate set of systems and they will be developing as they go. They won't be relying on live refreshes into the dev environment like we do for small change in bhau.

17:16  
Can I ask a quick question? Hmm.

17:18  
That um, that process of.

17:23  
Sorry if I got the terminology wrong, but locking

17:27  
piece of code that's being worked on. Did you say? Is that all written in house? Yeah, that tooling to let you do that? Yeah. OK. And do you have any documentation that would explain to somebody new to the team

17:40  
how that works or is it all you just you just teach him the right man? That's not a let me because I've actually got our page open. So we've got

17:54  
a diagram basically of how the source control works, but is that in confluence or something like that? Yeah, yeah, it's under 224 source control. When we when we finish, when we finish the day, if you can share the links with me, yeah, I'll talk to Mike and we'll get some of that. Some of the stuff on their shared with these guys just so they can see it will help the their analysis and stuff. Yeah, yeah, OK, yeah, fine. Yeah, great.

18:19  
Thanks though.

18:22  
Yeah thanks uh David and Paul. So in in such scenario, uh, do you face that as a challenge in order to maintain a proper source control? How the dev team feels without a proper source control mechanism?

18:36  
It's not, it's not.

18:38  
I'm a big issue. Uh, purely because

18:43  
the amount of small change we do, especially now, is

18:49  
a lot reduced compared to how it used to be, so the overhead isn't massive. Um. There's a few manual steps that we have to do, but it's not overly burdensome. So

19:01  
it it works for what we need it for. Um, I think if I if I'll small change portfolio is big

19:09  
then

19:10  
it may cause some issues but for the the amount of change we do not really.

19:17  
So if you had more projects coming in, more small change coming, Dave, it would be more beneficial if you had something like a proper,

19:25  
yeah, if in source control, if we had a lot of small change coming through, not so much project work, umm, because that obviously doesn't go live straight away, but for small change it's a quicker turn around time. So if we had a lot then yes, we would start to sort of feel a little bit using the process we've got at the moment. But for the volumes we have is inconsequence inconsequential

19:51  
and look looking ahead when we move towards tough Jay and that migration to

19:57  
so for the benefit of people in the cool.

20:01  
We run. Our version of 224 is, although it's a fairly recent version. How we

20:09  
install that and now we've run it and kind of a bit of a stack behind it, It's it's quite old fashioned from a terminal perspective. So there's a, there's a more modern way of doing it which we're being encouraged to move to which we'll look into in the future when we do that, Dave, you would then look to bring in Terminus's own source code product. Yeah. So that would that would sit properly. OK on our application then. Yeah, fine. OK. Thanks.

20:38  
So then how do you maintain so as of now until today, how do you maintain the history and archiving the data? So for example, last month you might have when some release in the staging and live right? So now you might be working on some other new changes, even though that changes are small. You how do you get that Release metrics or if someone needed the source code, if someone like Paul Mike coming for a technical internal audit, how do you present that particular source code without versioning and maintaining the history? David.

21:08  
So you see 24 J Base has never done that. Um. So when you change anything and it's recompiled,

21:17  
you lose what was there before. Um, the only way you could see what was there previously

21:25  
would be again to take a copy of live from a previous date. So before you make the changes, and then you could see what the old code said compared to what the new code says. That's the only way you could do it.

21:39  
You. Yeah, you just you don't get a version within J Base.

21:46  
It's it's been around for too long. There's been attempted in the past like trying to get under the source control mechanism. Is there any architect who can oversee these things? Did you know? So it it doesn't, that's that's the thing it it didn't come with source control and it doesn't really work with any off the shelf source control.

22:10  
So that's why we've had to develop our own. We've tried. Yeah, we've we've tried in the field.

22:18  
I'm sure you would definitely would try. That's why I just asked. Uh any architect, uh give us any suggestion, that's fine. Because in in such scenario, right? It will be you know, a headache for you if something goes into conflict. Because as a developer today I might develop something, I might write some 10 lines of code, then next day I might feel like OK, I don't need this 10 lines of code means I might need only the file lines of code. That history has to be maintained in uh two week Sprint. Otherwise they will lose track and they might lose motivation.

22:48  
And so if that is so, what what we do when we write a programme is we have

22:55  
a change history at the top of it. So as someone, so let's say for example this week I'm working on a programme and I need to add in a new subroutine. At the top of that

23:09  
I will actually write what it's related to, what the change is, who've done it and the date. Yes. So it is up to us to maintain that.

23:19  
So it's not completely foolproof, but that's the only way we can see what's happened historically. Unless, as I say, we take a copy of live from a previous state so that we can then see

23:34  
what the code change. You saw what the code used to look like compared to what we've got today. That would be the only way

23:42  
we can see the last time it was compiled. So we've got a command within J Base that allows us to see the last change, but it won't show us a history of any changes before that. It will just say this programme was recompiled on 1st of January 2020, but that would be it. OK, So yeah, sure. David, when you said you and Paul are tried for a source control mechanism, right? Which tool you have tried? Are you used to GitHub or I should have OPS or Bitbucket? Anything

24:12  
so we tried to get a few years ago. I seem to remember and we we also tried the vendor zone product, but as Dave's already touched on,

24:21  
the

24:23  
doesn't work with our

24:26  
our implementation. Basically we have to move to a more modern implementation of the product for it to work and so they're the two that I know of. I don't know. You've done other stuff though.

24:35  
Uh, no. I mean the one we've the one we've done recently, I think it's Azure we're using

24:43  
UM, which we've had to sort of shoehorn into

24:47  
get to work. But as you say the Git when we we tried that that just we just couldn't get the connections going. It just wasn't wasn't any good for us.

24:57  
Um. So, yeah, I mean, we've. I don't think there's

25:04  
any off the shelf ones that

25:06  
was that were good enough. I think there was a bit before my time. I think when Romsley was running the team, they were looking at various things and yet we've got the the vendor one, but that just didn't work properly for us.

25:26  
So does anyone have any, any other team have any dependency on this source code, whatever, you know, it's just it's an isolated team, is it? Yeah, yeah.

25:37  
So you independently develop independently you are, you have your own servers like dev, QA and this thing. Yeah. So we're we're completely stand alone as far as code goes. We've got no dependencies on anybody else and nobody's got dependencies on us.

25:54  
So you follow the UH GEL format, you have any Sprint methodologies, biweekly, Sprint or you don't follow that. It depends on the project. If the project uses sprints, the team will use sprints for small change. I think there's GIF mention we just we work in a waterfall. The request comes in the requirements sent to us, we write the spec. We do the dev test and the deployments.

26:21  
Yeah when when you explain the team structure right you explained us a small team of five member team. Yeah so does uh any you have any technical bias or trying to have the requirements from the business owners? No. So all of the team, to a certain level,

26:36  
will understand what the business does and how it works and how that fits into T24. There's no specific person for doing BA work. They're all highly skilled. Some have more knowledge in certain areas than others. But yeah, they they do everything from looking at the requirements right in the spec code, in IT, testing it. They all do the same thing

27:01  
and the this is a similar situation to what we talked about with privacy actually. So what Dave's talking about there is small change. So what we don't have in a small change process anywhere are actual BA's getting involved. You basically have the business right, their requirements. Now, they might use somebody in the business that they

27:24  
they call a business expert or they might even call them a

27:28  
UH, Yeah, maybe. Maybe they do, um, use the the term. But they're not. They don't really have BA's draw those requirements out. It's typically written by someone in the business who who's identified a need

27:42  
and then people like on Dave's team or Shari's team. All the other teams we have enough S work out from what these people have said, what they think they want to do.

27:50  
That process is different in a project space and the project space you typically have BA's assigned to them. So we have, we have two very

27:57  
discrete ways of working. The projects are done now you'd expect with BA's and doing all the analysis work with the business. Working with tech

28:05  
in a small chain space, we don't really have that.

28:09  
Yeah yeah I definitely understand that Paul. But um, in such scenario like UH, how about UH testing the same questions asked with screenwriter testing and DB. If you have any DB changes or unit testing or testing team if you don't have a dedicated team. Again, we can consider the same number of people as doing a development and testing both.

28:28  
So it's a it's the same thing. So so Dave's team will split those responsibilities between them so someone's gonna do the development.

28:37  
Uh, again, times product 24 is unusual because

28:43  
there isn't a separate database like an Oracle or anything like that. It's all

28:47  
it's all within within the application itself. So it's Dave's team.

28:52  
Do all of it. There's there isn't a separate team that looks after the database. It's all it's all done through Dave's team.

28:58  
For small change,

29:00  
his team would split responsibilities between doing development, their own unit testing,

29:05  
and then other people on the team would do system testing of that change.

29:10  
Then it gets deployed to a UAT environment

29:14  
and that

29:16  
dedicated UAT test team that we talked about in the past couple of sessions, they would pick up and do the business testing for the business that you AT. And when that gets signed off and approved,

29:28  
it goes to production.

29:30  
In a project that's different. There's a dedicated team for system testing and SIT. So Dave's team would just be responsible for development and unit testing.

29:40  
OK, so the teams responsible and from dev and SIT and UAT will be having access by business owners, correct? Yeah. Are they giving the sign off or production?

29:51  
Yes, Yes. Yeah. OK. So does these business owners have UH testing knowledge? How, what are the tools they're using or they're manually testing just like the developers in the business environment. This is that dedicated test team we talked about last time called UAT services. So there are a pool of people in markets and banking whose full time job is working on UAT for various systems that affect their business area.

30:20  
They have their own tools, their own scripting, their own tests. So um, they, they do all that, What tools they use, It's been a while, but typically most of it is manual if I'm honest. And they have a couple of simulators that they use to simulate Swift traffic, but that's about it. It's predominantly manual

30:42  
and the the zero right there will be called a business owner or the epic owner or the product owner. That person would have created that requirement. Once the developer completes, even after for the sign of write, the same person has to sign off or the person in the same team has to sign off or does that happen in the way?

31:02  
So when we do So what would happen is let's say somebody, um,

31:07  
is doing a job that would take a really simple example and they see there's a spelling mistake next to one of the fields. They've raised that as a change. OK, That then comes through to us. We confirm what the screen's called, we confirm the field, we make the change, we do the test, it goes to UAT.

31:27  
UT do the testing and they say, yes, that is now spelled correctly and then we go back to

31:35  
person in the business.

31:36  
Uh, probably not the person that actually noticed it, but someone higher up to say this is all now been done. UAT signed it off when you want it released to live and they will say OK, they look at their testing documents or whatever documents they need to look at and then they give us a date when they want us to release it.

31:53  
So we don't sign off. The UAT don't sign off to say put it into live, they just sign it off to say it's good enough to be released to live. And then whoever in the business happens to be in charge of that area says yes, we're happy with those results. You can release it to live next week.

32:10  
Sort of sign off for that

32:12  
undress. I guess

32:16  
a senior manager within the business is acting kind of as the product owner I suppose in that in that circumstance really

32:22  
um, but ultimately would have been there, he would have raised the requirements in the 1st place and they're then signing off the output of the final testing UAT to confirm that it meets their expectations.

32:34  
Yes, yes poll. So let us, if I go back a couple of months back and if we can ask UH user story has been signed off by a business owner. If I can get that metrics will that will that show that particular name of that person

32:49  
or it's not being documented like that

32:54  
for the for the small change, I don't even think it's in Nigeria. Is it Dave, is it in transition to Gerry yet is it OK, two, UH, we're still in K2, we're in the process of moving it. It's it's a different system to due the the concept of the same,

33:09  
but what you have is effectively a change request raised in the K2 system and that has a history of what's going on. Who raised it?

33:18  
Um,

33:20  
the estimates,

33:21  
who's done the dev? Uh, who's done the testing and who signed it off. So the the audit trail for that is all within this K2 product. So it's not 0. But there is a product that you can track that history in. Yes, sorry. So roundabout way of saying yes. OK no, no that's fine. Ohh, at least there's something from a tool is fine. It can track that. So this change request, what is the tool or the service you're using? Is the service now or it's called Kate too? OK too. Yeah, it's been around a little while, but

33:52  
that's a projects used JIRA. Small change goes through day 2, so you can see one of the problems we've got. The whole 2 are integrations and different things. They're not. Sorry.

34:03  
They're not. Yeah, yeah, and that is, that is a problem, right? Even a small change might affect a big change in the future. So we need some coordination on that. They need to be. They need to be linked together. So there's developers, everyone can just focus on Jira. That's really what we should be doing, but we are wasting time

34:22  
doubling up on things where we don't need to be doing it. I believe there is a project in flight or was in flight to move all that from K2 to JIRA, but I have no idea. Yeah, I have no idea where we are with that, but there is something in the pipeline,

34:36  
yeah. So what they've done is they've actually started migrating over to JIRA, OK, with the existing K2 stuff, but they're still raising in K2, OK. So it's it's not 100% complete yet.

34:49  
Those migrations have got start speeding up the cattle and they don't know. Yeah. OK, cheers. So Paul even, yeah, she was mentioning the other call, right, like he was using a different version control. They're migrating everything to Azure Git and they they are integrating with the pipelines. That's a sad new process. They're facing their own challenges, but they are picking it up. That's how they say, right. So why can't we follow a similar process for Dave's team? I mean, is there anything in the pipeline, any problem?

35:18  
So if we're talking about, if you talk about source control, that type of thing, again, um, no, not just source control like he integrated the pipelines also, right? So yeah, when you have continuous integration, delivery and even the testing, everything comes under one area, right. So do you have any plans for Dave's team also like that in the future?

35:40  
I don't, I don't think there are at the moment. I, I think there's a separate question,

35:46  
Ohh, maybe not separate question. OK. So I actually thought you were overseeing both the teams. So that's one of the things that this work that we're doing here I think is going to probably lead to

35:59  
is identifying where changes need to be made in order for us to get to where we want to be. And I think one of those things is ultimately trying to set our teams up

36:08  
to be like product teams really.

36:12  
So this type of this type of stuff should come through that I think,

36:16  
but there aren't any plans right now to do it. No, OK that's fine. So whatever the source control currently they have if they wanted to see how much of A code quality or the code coverage. Because normally if you go into an enterprise level organisation whether it's a small project or A5 member or A50 member project, the code coverage will be only 30 to 40 percentage. We can we can never pick it up out to 80 or 100 but we can increase up to 60 or 70. But still, do we have any metrics to define the code quality and the coverage for this particular project?

36:47  
Well, I don't think so. No, I don't think so. Dave, I don't think you can do anything that like that, Can you with what you've got for 224. So, no. So just to, I mean just to

36:56  
say something

36:59  
the

37:01  
the the programming language we use um

37:05  
basically date backs to the 1960s. OK, so it's it's gone through a few iterations and improved over the over the decades, but it's not

37:19  
very compatible with, um, tools and processes that we use for more modern languages or NET or whatever it might be. So a lot of the time these tools, because no one's ever heard of J Base or pick as it used to be, these tools aren't designed to work with it. So every time you try something it just doesn't work because it's not

37:49  
done in the same way. The compilers are all different, the the way it's coded is all different, the what it's coded in is different. So it causes a lot of issues. The same for automated testing. For many years, you couldn't get an automated testing tool to work with T20 Four because

38:10  
of the way it's structured. So until we move on to an Oracle database, we can then compile in Java. We can then use our the source control that's provided by the vendor, We can have pipelines, we can have all of the audit trails, history of the code and all of that. But up until now, that's just not been possible. You know, that's why we tend to write a lot of these things from scratch so that we can tailor it

38:41  
to work with T20 four, because off the shelf stuff just doesn't doesn't even know

38:48  
what what Jay base is, never heard of it. So that's the bit where it causes us problems, which is why we've got so many manual processes to do because it's just not, it's just not suited. So this database is a DBA tool?

39:05  
No? Joe Base is the programming language, so it's it's um.

39:11  
It's hard to explain, really. Um,

39:14  
it's a programming language. It's a programming language, but it's not like any programming language you've ever seen,

39:22  
to be honest. There was, there was a, there was a point back in the dim and distant past whether it was actually called English because you you basically write as you would speak for a lot of these things

39:35  
and it's it's just a very

39:40  
antiquated programming language.

39:43  
And because of the database that it sits on, so the whole J base

39:49  
package, it's very hard to integrate things into it, like plug anything in or, um,

39:56  
you know, use a standard set of tools because the tools just don't fit.

40:02  
With the White J base is written or the way it's where it sits,

40:08  
Yeah, we couldn't. We couldn't do a code quality on a piece of code because it just wouldn't fall into any tool that you can use to do that code quality and review. You have to read the code to see if it makes sense. You couldn't put it into a code piece of at all. To look at code and for it to highlight where there's inefficiencies or bad structure or whatever it might be. It just wouldn't do that. I'd look at it and go. I've not got a clue what you're talking about here. So it makes it very difficult for us to

40:38  
in utilise anything like that.

40:42  
Yeah, yeah, just to quickly did a bit of R&D. It looks like a DBA tool, right? A database programming tool. Yeah yes, database programming. So, so just like SQL for Microsoft, nothing like SQL from Microsoft,

40:57  
OK. I mean, I mean, it's up until recently it was all written in like capital letters, you know, it's only because the younger people have come in, they want it to look a bit more modern that they do it in camel case. But it makes no difference whether it's uppercase or lowercase.

41:13  
You know, there's all things like that within it. It's, it's antiquated. But The thing is it's it's once you start doing it, it's very stable. It's very easy to learn

41:25  
and it's very easy to write in,

41:27  
but it just looks completely different.

41:30  
Don't do anything else, so they cannot follow any code naming conventions. As it

41:36  
we don't we don't need to. We can call it anything we like, but I mean within within T 24, so from the vendor

41:43  
you've got things like internal calls, um, which obviously we can't see what the code is behind it. But you know, we've got three ways of reading a file. Or we can use the vendors subroutine that they've written, we just call that.

42:00  
So there are there are certain things that we use as best practise for code, but they're not 100% necessary and the stuff that's vendor written wouldn't show up anyway on anything

42:16  
you know. It's it's up to you if you use it or not.

42:19  
It's it's a very hard thing to explain unless you know, start walking you through the code to explain what each thing does. Yeah, yeah. So normally, I'm so sorry. Please go ahead.

42:31  
Ohh no. So it wasn't any interest carry on what you gonna say? No, no. Normally in in this scenario from the way you explained and I also did a bit of R&D, it's looked like a far deep legacy technology, right? Yeah. So in this scenario like you said the young people coming at, did you find any alternative NextGen technology which can able to do a similar job in a better and efficient way. Have you considered any

42:57  
ohh rather than you know trying to put the same old technology into the source code and creating a pipelines again? The pipelines also might not be adequate or suitable fit for that right? Instead the same job can be done by new technology right? Have you considered any migrations like that, like technology migrations

43:15  
that's that's the next thing that's coming up, right. So, so the next, the next big piece of work for the product beyond we've got a programming flight at the moment. The next big thing for this product

43:26  
is the move to what they call Taft Joe, which is basically Java

43:31  
and it also introduces the concept of an Oracle database for it. So it moves to a more modern sort of setup that then allows us to do all these type of things that we want to do.

43:43  
Exactly. That's the that's the because uh, yeah, we, we can't go through this code quality best practises, right. Uh, if this is the legacy, if when you get to that modernised. But again it should be a parallel development without impacting the business continuity. Yeah, because there are some clients might be still using the technologies, right. Yeah, exactly. Yeah. Yeah. OK. So I don't think then we have any topics on the TDP test driven development and principles and patterns on that point. No. OK,

44:12  
OK. So the database management, so I think it it itself is a database. Uh, right. Yeah. OK.

44:18  
So the people who are working on this have their their own knowledge on that or they have the sequel knowledge and they are coming into the thing and their understanding. So they've been recruited. They don't want J's already when they're coming into the system. Yeah, Yeah, yeah. OK. So they they they are like AJ based developer. Yeah. OK, OK, OK. Then quality assurance already we discussed. So I think the team is doing the QA only in the UAT environment the business users are doing and

44:49  
giving it to sign off, correct? OK. So do they follow any release cycles uh Paula or it's like a same dock one as in when necessary

45:00  
David you can as yeah as a man necessary isn't it time project some projects control their deliveries and then Dave and the team would from a small change it will get scheduled in through a central group I think streaming talked about last time called the coup in markets and banking.

45:17  
So ultimately they organise and arrange with Dave and and other teams in in the domain they're organise the timings for releases to to produce production. So they're kind of overseeing that small change process,

45:32  
Yeah, because if you say small changes are here and there, uh, do the business owners give any timelines or deadlines? Because uh, if we can take a lot of time, we can take a small amount of time, right. Yeah. Only the dev team knows how much, how many term much of time needs to you know, so we we agree that. So we agree that. So when the change is raised, all the estimates are done. And then

45:55  
I mean I'm going back slightly, but typically what would happen is that area would coordinate,

46:00  
um multiple change requests to be developed at the same time to release a package. So we'll do one at a time. We try and gather some the most important ones into window that we think is gonna make sense. Maybe it's a two-month life cycle or something. And then

46:18  
as a team, people work on multiple PC's at the same time to deliver it as a as a complete package.

46:25  
So you say like if a person completes within one week uh some other person completes in two weeks, you get as a whole package so that it has some uh meaningful impact dependency and then putting it in the package in the higher service. Exactly. Yeah. OK that that makes a lot of sense. Yeah. OK. So do do under team collaborative wise I don't think I do it there. There is no multi cultural here, right? Because there is not much of A multiple team involvement pretty much pretty much individual and

46:56  
and

46:57  
I think it's it's true Dave you would still manage the release process yourselves.

47:03  
So we do, we do at the moment, um,

47:09  
because since the upgrade, but yeah what we will go back to is OPS, we'll do the release out of out of ours. So we'll be going back to that, but because we've not really done any small change if it hasn't been anything to release for yonks. So and does that release process use

47:31  
the release management guys in tech? Was it all just done between you and OPS?

47:36  
No, It goes, it goes through the proper processes. OK yeah, OK.

47:43  
So when you say you have an OPS team for uh, deploying things, so how about in dev and uh SIT? Does the developer playing an OPS role there or they will seek help or option

47:55  
the line saying that look after

47:58  
all of our applications throughout tech, you know, they do the end of days, they monitor closer businesses. They're a separate team down in the dungeons and because they work 24 hours a day

48:12  
and our releases to life have to be done out of hours, we give them that to do. But they only do automated releases. So they they literally get a step by step instruction as in click that button, click that button, check that field, click that button, tick the box, you're done. Yeah. So there's they've got no expertise in any of the applications. They just follow a script

48:36  
and they use the the toolbox release tool that we have

48:41  
and that's it. They don't know what they're releasing, They're just following the instructions that are given to them, OK. And they're completely, completely separate from us because when you have both app and DB script, sometimes a DB script needs to be deployed first, sometimes the app needs to be deployed second. They don't know they they'll just follow the dev teams. Yeah, they would not have the faintest idea what you just said there. Put it that way. We have to literally stare management, do this. Step 2, breathe in, step three-step 4, breathe out. You know, it has to be that

49:11  
new level of instructions and to get all of the the, the, the sequence right because they won't know. They wouldn't sit there and think, oh, hang on a minute, shouldn't that be done before that? So it's it's on us to make sure that it's all documented and that they've got every single piece of information that they need.

49:30  
Basically it's like they're having the keys to the service. That's it. Yeah, yeah, yeah, Security. Yeah, they, they, they watch it and if it breaks, they phone. Whichever team looks after the thing that's broken, they don't touch anything. They just monitor it and they run. So they do, they do the backup and all. If something goes wrong, they know how to rollback. No, That's all part of the scripts that we've got. So we've got a suite, they run the suite, and within that suite is all of the instructions

50:01  
of all the scripts that need to be done. So they will click start,

50:06  
that suite will go off, it will do the backup, it will shut services, it will start services, it will do whatever it needs to do. And they will just watch each step. And if it doesn't go green, they phone us.

50:18  
Yeah. And then we will say right. OK, that's because of this, we will fix it and say right, can you press continue? That's all they do.

50:25  
OK, yeah, good. So they do a screen share or something. How they how they do this process.

50:31  
So they they'll just phone us and they will send us a screenshot of what the error is and we will log in and we will fix that and then we'll get probation between dev and OPS team.

50:43  
OK, well why is that? Is that the same with the all of the teams?

50:47  
Yeah, Ohh, it's just that's all they do.

50:50  
But, so if there's ever an issue, they just they just tell the the team that looks after that particular application that's gone wrong, that it's gone wrong. And then we they just sit and wait until we tell them those fixed and they can move on,

51:04  
but only if they have the expertise, only they can do it on their own right. Otherwise they should. No, no, no, no, no, no, no, no. They're all that, OK, this, this is segregation of duties as well as so. So it's to avoid the developers and the guys who've got all the knowledge and can do everything, having complete control about that release process. So the OPS guys do what's needed to release and deploy to the environment and that is literally a tickbox exercise. Press this button, press that, check this goes green. If there's any issues,

51:34  
they contact Dave's team who will go into the application and

51:38  
look at any issues or any flags. Um,

51:41  
so typically when they are releasing something, Dave's team are are online as well if needed to support the process.

51:48  
Yeah, yeah, yes Paul, I was just trying to compare what uh with what Sri has said. She mentioned like even they also have 4 environments, but the dev team have full access to the dev and SIT, they can able to do the deployments on their own and they will do the testing by themselves. But only for UAT and staging and live they will go for the upstreams helps like that. He missed the same for us. Yeah exactly. Yeah, for us, yeah, same thing. OK, OK, so for the dev and sit servers, right, that's the dev team have access to deploy on their own

52:18  
if they can get into the server and override the packages, yeah. So dev, yeah, for us is ours. We can do whatever we like with it. We've got 100% access to it for UAT and for live we have limited access and any enhanced access needs to go through a process where we request passwords that give us lower level access like root and things like that. Otherwise we because as a dev team we don't

52:49  
own UAT and Live so we don't have the access to it. We own dev so we can do what we like with it and we've got full access to everything within that. But any any more access on the other two would need to go through procedure to give us the access that's audited or audit trailed.

53:08  
OK. So

53:10  
do they track all these deployments, Uh, for example in dev, since you have opened control, right, you can deploy a number of times, right? Hmm. So do you have there any track record or the track record comes from the UAT or the live service? So in a in a year how many releases have been went through the track record? So we don't track that within dev, and we don't track it within UAT or live either.

53:34  
OK,

53:37  
OK. So so the only way that it be tracks or to interrupt, the only way it be tracked is because we need to raise a change request a CRQ, basically management, yeah. So they would track what we've released them, what's gone in, but as a team we don't.

53:55  
Yes Sir David, so we had uh you know other calls also like we I think Paul might be part of that. We have done some DevOps release management focus area just like this. This is software development and engineering practises, right. We had had other call with other team members for the DevOps and release management like how the change management, config management works, how the servers are maintained, how the OPS teams working at the they have any proper collaboration. So maybe on that area if you would have

54:25  
that we could get some additional inputs. Are you happen to be part of that? All right,

54:31  
Ohh, no, no. So it it happened a couple of weeks back, yeah. DevOps, release management. So at that time we were raising the same questions to the team, so they were giving their own inputs. So how about the security practises? How do you? Can you give some metrics to prove that the application is highly secure and environments are properly secured?

54:54  
Um,

54:56  
so we get

54:58  
every year audited

55:00  
and we have to prove that the the access that we have as a dev team to live is secure. The live environment is secure that there's up-to-date patches within the systems that we use.

55:18  
Um,

55:19  
that's.

55:21  
There's also the they do a pen test as well to see if they can actually break into the system. That happens frequently. Not that people break in, but that they check that you can't break in.

55:35  
That's all done and looked after outside of my team.

55:42  
So they they will come to us come to ask us to supply them with information and we will supply that information and then they can they make a decision on whether they think it's secure or whether we've got a security risk.

55:57  
OK. So the that internal team whenever they come, so they'll find if they find anything. So the the teams aren't internal to the bank. We're audited by KPMG every year, which actually is going to be starting again in a little while for 2024.

56:15  
Do have an internal audit team as well, but the the pen tests are all done by external companies.

56:23  
It's like a yearly ones exercise or frequent exercise. So auditing is is annual

56:30  
and yeah, OK. So in the interest of time, I'll just quickly get through the technical debt management. Do you think you have a technical debts in your team or do you think it can be managed. So if you're not familiar with the term, I can explain, yeah, I don't know what that means that that's fine. So normally in the normal world, in banking world you have debt means the debt will keep on accumulate unless you clear off the debt. Similarly in a technical debt means you might have some technical inputs you wanted to

57:00  
you know incorporate into your system but because of your UH frequent activities of UH regular UH process, you might be putting that in your backlog. It can stay as a depth forever. So that kind of technical debt management are you doing or do you feel like the there is no need for technical debt here. So just touching back on the Geo rock two discussion earlier, we went through a big process of going through the backlog of small change

57:29  
to see what was still relevant and what wasn't relevant anymore

57:33  
and we got rid of those. So the only ones that are left are the ones that once we finished the project, we will then start working on UM.

57:41  
But as I say, because there's such a small turnover of change outside of projects, we don't really have a lot in the way of debt. So there isn't really anything that we would need to concern ourselves with

57:54  
on that part. I think a few years back when we had an older version of T20 Four, there was a lot of change because the system wasn't what the business required at the time. So yes, there would have been

58:08  
and requirements of manage all of that backlog, but now since we've got a new all singing, all dancing system so to speak, that has gone drastically down.

58:20  
Got it

58:22  
to it. So if I say I'm I'm ready to incorporate dev OPS, best practises and principles and patterns. Is the system ready? Is your system you can you say your system is ready? OK, Mahindra, you come up with your own tooling and technologies, develops, practises. My system is ready. Can you say that or still that is a technical debt for you. I I don't think it is to be honest with you. OK,

58:50  
OK. So how how how about the knowledge sharing and documentation? That's the last area we're touching upon here. So for example, you mentioned that you are recruiting people based on their J based knowledge, right? Yeah. So they are considered themselves J based developer. So when they come into even though it's AJ based developer, they need to understand this banking system and how this particular technology or this project is getting along and what is the end output, those domain knowledge they should be having, right? Do they have any? Do we have any proper documentation,

59:21  
any and all Confluence documentation or video documentation so that the newcomer can able to grasp and understand right So 99% of J Base programmers programme on 24. There's very few other places out there where you would use J Base and not be working on 24. So when we do take someone on, they already know

59:43  
the processes for the system and so there isn't a huge amount for them to learn. Now there are some unique business processes that they need to learn and understand.

1:00:00  
But we do. I mean we've taken on a training at the moment and we do have documentation for T24 for programming um structure of you know the the feeds coming in and going out.

1:00:13  
So we have all of that in place. But it's it's more for for us really people when they start, they don't tend to need it unless as I say we've got a new guy who's never looked at 24 before. So that's what that documentation is in place for.

1:00:30  
OK, You're saying it's it's not like.net or Java. You learn a technology and applying it to a financial sector or a healthcare sector? No. Then sector and the technology remains the same. Are you saying? Yeah, yeah. So basically the only vendor out there that uses this is Temenos and T 24. So to find someone who's AJ based programmer that has never used 24 is very very very rare.

1:00:56  
OK, so to conclude with this discussion, right, I think this is more of a starting from the finish more of a clear cut discussion. We can able to get through all the points. What kind of areas do you think you'll be needing improvement on your own project to perform to a still more modernised and betterment level. You can increase the throughput

1:01:16  
either way increasing the changing the project under CSM or my embracing a modern technology

1:01:24  
so I think that the the upgrade will do next. Once this is finished will allow us to do more code breast, best practise, enable us to have more tooling.

1:01:39  
So I think you know at the moment I don't think there is really anything that we could do to our current setup that would improve anything. But going forward it will be the ability to have automated tools and testing tools, deployment tools,

1:01:59  
best practise code reviews and all that sort of stuff. But at the moment, I can't see anything on there

1:02:06  
that one would work with what we've got. Two would actually improve what we've got

1:02:13  
purely for the reasons said earlier of you know it's no matter what we do there would be a substantial amount of manual effort around all of it.

1:02:23  
Correct. Even if I'm ready and I'm, I'm saying I'm ready coming up with the practises some trade off has to happen right. To embrace for you. Yeah. Yeah it would, it would be, it would be wasted on us as it stands at the moment

1:02:39  
because we wouldn't we just wouldn't be able to implement it fully

1:02:44  
true. True to that David. I think that's it from me. Do you have any questions for me?

1:02:51  
No. All good. I'm just gonna go through time. Afraid we've got a few of us we probably gotta get onto to something else now.

1:02:59  
Umm, great. OK, excellent. So I'll take it from what you said earlier. Do you still want me to set up another final session with three?

1:03:09  
No, I don't think so. For sure. I think no, because uh, OK, uh, I I might ask a question a different way. So what do you think? Uh Sri might give further Harry Jondal remaining 8 topics, right. Like DB management? Because Dave's technology is different, these technologies different, right? Yeah. Let me talk to him. Let me talk to experience if there's anything else to feedback about them and you just need improving. Correct. If you can give an overview of what we discussed. Yep. If we

1:03:38  
really feels like it's fine, that's fine. Otherwise, definitely we can have a 30 minutes quick session that should be there. Yeah. OK. I think more or less we got the inputs, whatever we need to export. OK. Leave that with me and I'll come back to you by the end of the week. All right. Thanks for your time, pal. Thanks for. Thanks, guys. Yeah, no problem. Thanks, Dave. Take it easy. Thank you. Soon. Cheers. Bye. Bye. Cheers. Bye. Have a nice day.

1:04:07  
Hi.