Bhavna Wadhwa: In the world of banking, even among some of the smartest and most thoughtful people, the code remains something of a mystery at many levels. Does legacy automatically equal obsolescence? When is the right time to update functionality? An update to what, exactly?

The questions go on and on, and the banking lives of countless consumers depend on the right answers.

To join host Dharmesh Mistry on the inaugural episode of season one as he delves into these queries with the core technology luminary Dave Revell, Chairman of the Board at Zafin, exclusively on Banking Blueprints.

Welcome to season one of the Banking Blueprints podcast.

Dharmesh Mistry: It's a brand new series by Zafin and me, Dharmesh Mistry, and I'm super excited to be your host for season one. Throughout this inaugural season, we'll delve into the realm of banking innovation, exploring strategies for breaking free from traditional constraints and innovating beyond the core.

Now as I clearly don't know everything, although I've been around a long time in banking, I'm calling upon the experts to join me this time. And this week, I have Dave Revell, who not only is the Chairman of Zafin, but has a long history with banks.

And I'm going to let him introduce himself because he does a better job of that than I would do.

So welcome, Dave.

Dave Revell: Good morning, Dharmesh, and thank you for having me here today. As you mentioned, I'm currently Chairman of the Board of Zafin. I'm also a director on a number of other financial services companies and FinTechs. Prior to that, I was the Global Chief Information Officer for CIBC for about a decade. I had 15 years before that as the Head of Application Development, Strategy, Architecture, Business Reengineering at Bank of Montreal, and as a kid coming out of school, I spent the first decade with IBM in a variety of sales, consulting and marketing roles.

DM: You have similar kind of background to me where you've been on the bank side as well as on the vendor side. So it's a good breadth to have, right?

DR: Yeah, I've enjoyed it. And I guess the other thing that I've been involved in for probably the last 10 to 15 years, I've been involved with early-stage companies as a startup. I'm an Angel investor, so I sit on boards, I give advice and I write checks for companies as well, which tends to focus the mind a lot when you're looking at new ventures you might get involved in.

DM: Well, I'd wish I'd met you a long time ago because I've been through a few start-ups myself and I promised my wife after selling the last one, I'm not going to do anymore. Don't worry I won't be tapping you up for some cash.

DR: No problem.

DM: But great to have you on the show.

DR: Glad to be here.

DM: This week, what we really want to delve into is, we hear a lot of chat about, you know, banks are being held back by their core.

DR: Right.

DM: You know, what is the problem here? I mean really when I look at some of the things that banks are doing, they're not really innovating products. The core actually works, it's been there. Does it need to be changed at all? From your perspective, you've been inside these banks: what is the problem with their core?

DR: That's a great question actually. And I have lived with this through two different banks over 25 years, and I've seen the evolution of the cores. And as you mentioned, the thing about the old core systems, the legacy systems at many large banks is: they work. They've been there for a long time. They're effective, they've been tuned and they run pretty efficiently over time and they basically work to serve the business needs. I think there's a number of issues but for me it really boils down to two.

And I think the first and major one is a question of agility, and agility would be both time to market and cost to get to market. So, for a typical large bank, the time from having an idea for a new product to actually getting that launched and delivered, you're probably looking at best case maybe 12 months, probably more accurately 12 to 18 months to bring something to market.

And part of that is a factor of the huge complexity that's built up over years with these systems, plus the fact that they are very difficult to work with, which leads into the next item, which is around skills.

So increasingly, it's extremely difficult to get people that are trained that can actually work on the bank's legacy systems. They're not training kids out of school on the type of technology that legacy systems are written in. And yes, you can train people, but it's not really considered an attractive path. If you're a new engineering or computer science grad and you're coming out of school, the type of skills you would need to do work on the legacy system are not typically something that you would want to gain up to speed on and putting on your resume. And in fact, the people that could train you on that at the bank that have been around for multiple decades and understand both the technology but also the complexities of how it's written is rapidly diminishing.

They're pretty much all on the retirement corridor. Many of them have happily retired, lots of them have happily retired and are now consulting back to the organizations because the skills are needed. But it's becoming increasingly difficult to get the skills that are required to keep the systems up to date.

So other reasons as well, but for me, the two big ones as I said are agility, which is time and cost to market and the skills to maintain and to build on those old systems.

DM: I get that, and I've heard that a fair bit, but let's not take it, you know, at that level. Let's delve a little bit deeper.

DR: Sure.

DM: When we think about the time and agility, right, I don't see banks innovating on a very rapid basis. It's not like they're bringing out brand new products very regularly. In fact, if I think about it, the last time in fact, if I think about it, the last time I saw a really innovative product maybe was the offset mortgage. Some might argue 'Buy now, pay later' has become the latest, but you know there's a 20-year gap between the two. So, everything else in between could be really just a rate change or some basic terms and conditions. Is time and agility really that important that they're not able to change their systems in time?

DR: Yeah, well, take the two issues you just spoke about there. Even things like simple rate changes, simple rate changes and minor changes to the products that are not considered dramatically innovative in terms of something new out to market, they still take a lot of time. They take a lot of time to do because the complexity of these systems, you could have hundreds of interfaces that have been built up over the years, all of which need to be individually tested, they need to be coded, they need to be tested. So, the time you'd be quite shocked at the time that it would take to do things like quote unquote simple rate changes and things for products. It would apply to an increasing number of regulatory demands that are coming out that also require tweaks and changes to your product systems.

So my view is that it's not just coming out with a new creative product, it's the bread and butter. The basic products you have to do are still overly long and overly expensive to go and bring to market.

DM: I think a penny just dropped for me. If I'm paraphrasing you or putting words in your mouth, just say so, right?

DR: As long as the words are better, feel free to put as many words as you'd like.

DM: I think the penny is that it's not about how long the change takes, it's how quickly you can test those systems and deploy that update. Is that what we're talking about here? It's that, when you said it's not just retesting the core, but because the core is integrated with other systems that rely on these products and their data, then it becomes a big task irrespective of the size of the change, right.

DR: That's absolutely true. And you think about these old legacy systems, even the words we use 'legacy system' makes you think of kind of a single monolithic system. But in fact, what's happened over years and over decades, these things have continued to grow and there's kind of a spaghetti nest of systems that are connected into it that have been added on. So, when we talk about the core, the core is really a very complex series of systems put together. And again, hundreds of interfaces into other different systems that have built up over time that all need to be tested.

So if I took the example of a modern, newly architected core system, what it would take to do a rate change is often times it wouldn't even need to be involved. It's basically the business could sit down and on a screen being able to go and choose and change the parameters and get that out to market very, very quickly without technology really being involved. An old legacy system, because of the things we just spoke about, have all the testing requirements, have all the work that has to happen.

So it's months and months and months worth of work. And if you think about the environment that we're in right now, you think of what's happened with interest rates in the past year or so. For me that would be something that banks have had to struggle with a lot, a rapid change in interest rates in terms of how do you meet customer demands, how do you change your products. And those banks that have had to rely on changing their legacy systems have been very slow to being able to respond to that both to meet customer needs or also to try to address just basic profitability of how the bank does business.

So, it's a real inhibitor.

DM: So, I get this now is that the time factor is a combination of two things in effect, really. One is the fact that the core is a monolithic application. So even though there might be a small piece of code that really handles, let's say, a rate change, it's part of this overall suite that does everything, you know from onboarding the client or creating the account, operating the account, applying the charges, producing the statement, etcetera.

Now those bits may not have been touched, but because you're deploying all of that code again with this small change in it, it all needs to be retested, right?

DR: Absolutely. And you have millions of customers that are relying upon this.

So if I could put on my old CIO hat for a minute, the thing you hate as a CIO when you're involved in changes like this is, although there's many factors that are involved in launching a new product, the business needs to be involved. There's a variety of other things. But when you look at it, the thing I always hated as a CIO is when we're the long pole in the tent, when the time frame to get something delivered has this big bulge in the middle, which is we're waiting on IT, we're waiting on IT to code, we're waiting on IT to test. And it's never a situation you really want to be in as a CIO.

DM: I love that analogy of the long pole in the tent. So in my naivety, I guess part of me thought, well you know by the time you update the call centre people with the new product, by the time you've created new collateral, by the time you've gone through compliance and regulation and then rolled out, you know updates to the branches, IT would be ready. But actually, in my simplistic view was just thinking about the small change that we're making, not about how expansive that impact has.

And also, you know, everything that goes behind getting out an update in effect.
And I guess this is where like modern systems that are based around micro services fundamentally change the game because you're not deploying A monolithic piece of software anymore.

If you make an update to 1 micro service, you test the micro service and redeploy just that one bit rather than the whole thing.
And the way that it's tested with its automated test scripts mean that anything that was consuming that service is automatically retested at the same time, but it's in an automated fashion, right.

DR: That's absolutely true, and in fact I'll extend that a bit more. So, the whole promise of modern systems where you have isolated, you've got the API and you've got the micro service.
So yes, you just have to change the micro service, but in many modern systems you're not even doing that. It's parameter driven for things like rate changes and things. I'm not changing the code. IT doesn't have to be involved in doing that. I can sit on the screen, and I can decide that I want to change a number of the different factors related to rates etcetera and essentially build a new product and have it deployed without having to do. So, it's basically you know a no code change to do some of the basic things you need to do.

DM: Yeah, I guess, you know, I've taken it for granted that the core is where you define the product and then you on board the customers and then you operate the product. But that doesn't have to be the case, right? You can still have a call, but something else allows you to define the product and you know to apply any product behaviour to it outside of the core, right. Is that what you're saying?

DR: Absolutely. And that can be in a no code environment and there's many advantages to taking when we talk about product, again, product in the core systems has built up over time that there's product for your consumer lending, there's product for deposit systems and they tend to be isolated and embedded within different parts of the core. There's a tremendous amount of value that can be gained by actually putting all that product information together into a central spot, in essence, creating a product master with all that information and then drawing on and changing that.

Once again, we think of the old core systems, and we think about them monolithically and we talk about them being engineered.
They're not.

Maybe they were engineered when they were first built, but then they have been patched.
They have been extended; they have been Frankenstein'd over many different years on top of that.
So when you look at something and say, well this is the way it's been built, it's been built because 20 years ago someone had requested something different with a product or a regulator had asked for that and it never gets cleaned up. So, they are as far away from a tightly engineered system as you could imagine. And because of that, the complexity of trying to go and introduce things is much more difficult than it would be if you had a new modern, well-architected purpose-built system.

DM: So what you're talking about here is that banks over a few decades have created silos or product silos quite often using different cores for different products sets like mortgages or for lending or for term deposits, etcetera, right?

DR: Correct.

DM: That’s okay, because maybe 20-30 years ago, or 50 years ago, certainly the credit card only came about in the 60s, right? Banks didn’t necessarily have the foresignt of all the products that they were going to have. Secondly, as those organizations grew, right---and I am making an excuse because it’s a valid excuse, businesses grow, and we don’t know where the customer’s going to go. Therefore, we have to build things flexibly. But how many times would we get a budget in IT that says, look, I’m building you this current account capability. It does allow you to have an overdraft. But you know, if we build in a bit more flexibility, we could treat it like a loan, you know. And we don’t get the budget for that because the guy that owns the current account says ‘No, just build it for me, and I need it as soon as possible’, right? So, there’s amny reasons why these silos exist, but having the silos is a problem because you have to make different changes for different products in different places, right?

DR: Yeah.

DM: And they all impact, you know, this ecosystem of systems.

DR: Yeah, I’ll tell you a story about that and one of my earlier roles that we talked about. These systems being around for decades? Decades is optimistic. In some cases, 20+ years ago at one of the banks I was involved in, we were doing changes to the credit card system, which I won’t name the bank, but the credit card system was called NCCS. And I never actually realized what the name meant. But as I found out afterwards, the name was the new credit card system, which was already 20 years old at that time, was a deep legacy system, was called the ‘New Credit Card System’.

And when it was built, it was intended to be the first phase of a launch that would then include other products. And as often happens, it, you know, took too long, became too complex to do so, it became just the credit card system. That's pretty much the case of what occurs at many other banks and in an environment where all banks are created equal and they all have the same constraints, maybe that's OK, it's not great for customers, but maybe it's OK.

I think what we're increasingly seeing with banks is you're seeing the neo banks, you're seeing new startups, digital banks that have been created that are attacking that as a problem with banks. So, they're saying we don't have to have every single banking product the same way and create a full-service bank, the same with all the different channels. We can create a digital bank, we can have brand net new systems and we could operate at a fraction of the cost at a much higher pace, and we could actually bring innovation to banking and kind of dance around the old elephants that have been in the field there.

And that's a real risk for banks. And as technology continues to advance, there's more and more cases of either startups that are doing that or really large banks that have decided to innovate and attack themselves. Where they in essence have created their own digital banks. And they've decided it's easier to start new and build a net new digital bank, maybe migrate their customers over to it over time. Because there's this pressing need that they know that they need to modernize their systems and they're struggling to find different ways to go do that.

DM: So, so true because here in the UK I've seen both Monza and Starling and some of the smaller neo banks launch new term deposit savings accounts in response to the interest rate hikes that we've had, right. And there's been a backlash with the incumbent banks. People are saying, look, you guys have most of the customer base, but you're not giving us the benefit of these interest rate hikes. You know, it's OK when you're lending to us at cheaper and cheaper rates, but now that we've got the chance to save and earn interest at better rates, how come you're not raising the rates fast enough? And now I do understand that actually it's not necessarily the banks just being greedy and not wanting to pay, but it's their actual ability not to be able to, because who doesn't want to compete for those deposits anyway? It doesn't make sense to not give that extra rate and keep the customer rather than let them go away to somebody else, right.

DR: That's true. Because of course every bank wants to be more agile, they want to be able to meet customer needs more and that drives the success of the bank. And you mentioned before, Dharmesh, about the more modern architecture about microservices driven.

And that's another thing, when I think about core modernization in the past, core modernization often was seen as a single entity, a thing that you would have to do and for a large bank that could be a literally multibillion dollar exercise taking place over years. And because of the complexity, the rest of the activities of the bank really kind of shut down while you're doing this. So much energy is focused upon doing the new transformation that you're not really innovating with customers, you're changing technology, you're changing business processes.

And I think what's happened now is that the technology has advanced to the stage that what we're seeing much more of is really an incremental approach to modernization, which means I don't have to rip and replace everything all at once. I can take those old core systems, I can skinny them down. I can externalize some parts of it. And externalize means take the mode, put them into a modern architecture, separated but connected back into the old systems. And I can get at the modernization in different pieces and I can actually deliver a better business case from doing so.

DM: Wow, so what you're actually saying again is that there's things like being able to launch new products, people deem that to be a core activity. But actually, you could externalize that, keep your old core, but now start by using a third-party component, start to launch new products faster without going through the full expense of replacing your core. Is that what you're saying?

DR: That is what I'm saying. So, I'll do a paid commercial announcement for Zafin, because it's actually one of the things that Zafin does extremely well. Zafin is in the business, product and pricing platform. What you can do is you can actually take that product and pricing, you can take them out of the different core systems, create a single product master and basically have externalized that into a modern flexible architecture. You could have all the benefits on the product side of the business of being able to go and define your new products, bring the new market quickly and actually get a business case while you're continuing to do the other heavy lifting about, you know, changing the rest of the core systems along with it. And in fact, you also have the ability to, if you have decided on new core system, you can connect to both the old core system and the new core and do them both at the same time as you're doing the migration. So, there are techniques that are available now to make it simpler and also to reduce the risk

like a rip and replace is a very risky operation.

Doing something more incremental where I can take it in stages and also see some business benefit of doing that is very attractive compared to what the alternatives were even 10 years ago.

DM: Look, I don't want to join the bandwagon of promoting your products, but of course but what I have seen in the last decade or so is things that were in the core inherently have now got so specialised that it makes sense now to replace those bits of functionality with something that is just dedicated to that thing. I think of credit scoring and anti-money laundering, fraud detection, security, a lot of these things were packed into core at a time when you know the complexity behind AML processing wasn't that great. But now with new data sources and new AI driven algorithms checking the data, you know these specialisms are far better than what was originally created in the core.

So, what I'm starting to see now is banks, I think you called it ‘skinning up the core’ right here. You know, I’ve heard the term ‘hollowing out the core’. So naturally it's doing less and less and less while these specialized components are taking over and doing a better job of what the core used to do.

DR: Yeah. And Dharmesh, that's a very valid strategy that a number of global banks are doing like rather than trying to approach saying I'm going to completely change the core, they do what you're describing your Halloween out, you take the core and you reduce it back to the basic components are really what it was intended to be and you externalize some of the different functions along with it. You wrap it with APIs, you create microservices on the outside. So you wind up with a much smaller in your term hollowed out core, but still very effective and in some cases quite efficient core with many of the other systems that have been externalized and modernized and give both the employees and the customers that flexibility to market you need and again while reducing the risk of doing a complete rip and replace the system. So that’s a very valid strategy. I would say most banks have probably gone some way down the road on that strategy even if they are concerning at some point doing a complete change over the new core. That's often the first path that you would take is to hollow it out, externalize some things and simplify the systems you already have.

DM: Great. Okay. So, at the beginning you said there were two main problems. We kind of discussed agility in the cost side of it. The other point, it was really about the skill set. Now, I get it. Even when I was at Lloyds, our core was actually written in assembly language, right? And yeah, I dare say there aren't going to be too many people that really want to start to mess around with the assembly code itself or that, too many people that have that specific skill set.

But actually, a lot of the banks are running on COBOL. Again, I know it's deemed to be a legacy language, but A., it works, B., there are still thousands of COBOL developers out there, right? And it’s a three GL. After all, it’s more verbose than something like assembly language. So, I can imagine even if I was a C programmer or a Java programmer, actually doing some COBOL would be relatively easy or easier for me to get onto. So, what is the issue with the skill sets really?

DR: So, I think Dharmesh, the issue with the skill sets really is, and you're correct, there's a difference between some of the old systems that were written in the assembler versus the COBOL ones. But in any case, it's still a lot of skill that's being trained out of school. And there's the coding of the systems, but

there's also the complexity of understanding what happens in those systems in behind it, the architecture.

And those two things are intertwined, the knowledge of the systems and the knowledge of how the systems have been built and architected. So that when I make a change, it doesn't have an unintended consequence. And there have been multiple issues around the world.

I won't name some of the different banks where you've seen very bad major outages that have occurred based on what was considered kind of a simple routine maintenance change.
And in many cases, that's a combination of you've changed or done something to a core system and really not understood what was in the background and behind that.
So I think the longer things go on, the more there is that risk associated with things of having people that are knowledgeable in the code but are also knowledgeable in the systems and architecture.

And you know, you're right, you can train people to do some of the old systems.
At one of my previous organizations, we actually created a university program with one of the leading universities in Canada and we took new kids coming out of school.
We married them up with the guys that have been 2025 thirty years at the bank and had the knowledge and we basically put together like an old mainframe class, and we paid people on a basis where this was considered a hot skill, and it was great.

That's a wonderful stopgap measure and it worked for a number of years. But eventually the people want to move on. If they look at their resumes and they look at moving on to another organization, saying, I have become like a Level 3 expert in COBOL, quoting at a Canadian bank in this situation doesn't really get them where they want to go. So certainly, you can do those types of things with people, but you're really pushing against the tide.

You can delay things; you can improve your organization a bit.
But the fact is that the new people coming out, they're working with modern systems, they want to do that.
There's advantages to it and you can push back time a little bit, but you can't stop the flow of time.

DM: Yeah, this is my second penny drop. It’s the cost of these skills sets is one factor, right? But it’s almost like the change being small, therefore it shouldn’t take too long, right? But actually, even if you could get a flurry of these COBOL programmers, the fact that it's a monolithic piece of code with millions of lines of code there, training somebody in COBOL doesn't mean that they understand the system, which means that they make a change. We still have to retest everything and also the risk of them upsetting something is far higher because they really don't know the impact of that change that they're making, right?

DR: I cannot tell you I cannot tell you the amount of times over my career where you would have a ugly outage or something that occurred and you have the war room with people involved in it.
And I stare around the room at the people that have been in the bank with the most knowledge for 30 plus years fixing this. And you think, dear God, what happens if they leave?

DM: There's a lot we take for granted, especially with compilers and stuff. They highlight the errors with modern languages, right? But I remember a time, and this shows my age, I was working on a mainframe batch program, and I got to the end of the week and I should have been done to hit my target and the program just kept going into the error routine.
I'm like, why is this going into error?
Everything, the data, the parameters, everything was correct. I followed this line by line and like I just can't see it. And then I decided that, look, Monday morning this code has to be correct.
So I'm in at the weekend and I spent the whole day Saturday just checking the code.

I rewrote it and it still was going to the error routine.
And then Sunday evening when I was going back home with my tail between my legs, thinking, God, I just can't understand why I'm such a bad programmer.
Looking at this piece of code on the green sheets of paper that you go off the mainframe, right printed off code.
And I noticed that my last statement before the error routine didn't have this full stop.
And it's like, oh, it thinks it needs to naturally flow, not break at that point.

It's that one tiny little error that forced into the error routine. Jeez, I've just wasted a weekend over a full stop, right.

DR: The other analogy I do is, and I'm hoping this is an example that resonates in the UK, is when you're looking to modernize.

If I'm building a house, I'm modernizing an old house around there and I can build on the old systems and things that are around. But if the house was wired electrical with old knob and tube wiring, at some point in time as you modernize and you build your house, you have to bite the bullet and you have to change.

You're creating a fire trap.

You're creating something that is extended beyond the architecture of what it was originally designed to do, and IT systems are the same away with banks. You can work around it to a certain degree; you can extend things.

But at some point you have to bite the bullet and recognize that technology has changed, the needs of the organization have changed, and you have to find a way to address it, either by the technique we talked about of either kind of incrementally hollowing out the core, hiring new skills, buying new products to come in, or complete system replacement.

You have to get at it, because if you don't, people won't want to be buying that old house.

They're going to just buy a new house. They're going to buy a nice, modern new house and it may not have some of the same features and some of the character of the old house, but it's going to work and it's going to be maintainable.

DM: Okay, okay. So I get your two points now, but, there's a small subtlety I think that stops banks from changing their core. And because I have that privilege, I'm going to just spell it out and just say, look, there are CEOs, mainly CTOs and CIOs of banks that are probably about my age, reaching retirement, right? Do I really want to be changing the core at my time in life? You know, just can see my holiday

home on the beach. I'm close to getting my pension: why should I take the risk? What would you say to those people?

DR: I would say to those people, "I have lived your situation." I have been in that situation, and for me, the key word that you said is risk. And if you're in that situation as a CEO or a CIO and you see this change as being overwhelmingly risky change to do, then you're going to be very reticent about starting it. And to give people credit, it's not usually because, hey, I just want to get out of here in a year or two. It's 'these projects could be long and complex and they need some continuity to get them done'. I think what's changing these days and why we're seeing more banks embracing this is that the modern systems, the techniques we talked about, about an incremental approach to modernization and the hollowing out, they're actually reducing risk. So you can say, look, if I'm in the C-suite, either a CIO or a CEO, my responsibility is to the organization. And if I can help move the organization forward, and do it in a manner that isn't taking on a crazy amount of risk, and I can do it in an incremental fashion, why would I not want to do that? Regardless of where I am in my career? So I think that the new changes, the new approaches we can have with technology, an incremental approach, change that equation because they actually make it less risky to go ahead and do the modernization, and it allows you to attack it in pieces, and begin seeing a business return for it as you do those pieces. So for me, I think that would be the answer I'd have to your question.

DM: Fantastic. I think you're spot-on with the risk side of things. Is there any other advice that you'd give to a CTO or CIO that's deliberating or sitting on the edge about thinking what they need to do with their core, whilst they've got the business in there going, well, look, I just want my new product out quickly.?

DR: Yeah. You know, Dharmesh, in my role now, I know a lot about banks as I worked within the organization of a couple of organizations over the last 25 years. In my current role acting as a corporate director and an advisor to many organizations, I actually see dozens of large banks around the world.

And I can tell you, I would be hard pressed to name a single one that isn't thinking about this issue.

DM: Right.

DR: They're either thinking about it and planning for how to do it, or they're at some stage along the path of doing it. But if you're not thinking about it and you have a major organization right now, you know, I think you really have to question why not because the world is moving around you and you need to have a strategy for it. The strategy doesn't have to be the same with every organization, but if you don't have a strategy for how you're going to modernize, how you're going to meet evolving set of customer needs, how you're going to respond to the new competitors and how you're going to change the cost curve of your business, if you don't have a strategy around that, are you really doing your job?

DM: Fantastic. Look, Dave, it's clear that you've got a wealth of experience. Thank you for sharing that with us. Thank you for helping me and the audience to understand what really are the challenges. You summarize them as two main things, as agility and cost. The cost being, you know, not just the change, but the impact of that change inside a monolith that's then ingrained in, you know, a whole ecosystem or other systems that interact with it.

And the second being the skill set issue.

I now also understand that it's not about learning that skill set, but the impact of that change on this monolithic piece of code.

Again, so appreciate that and also see that risk is a massive factor, but there is a different way of doing this other than just trying to do a wholesale replacement of your core.

I really appreciate your input on this, and it's been great talking to you.

DR: Thank you, Dharmesh. And well-summarized; it’s been a pleasure talking to you today.

BW: Thank you for tuning into Banking Blueprints. We invite you to stay tuned for future episodes. On behalf of our season one host Dharmesh Mistry and the team here at Zafin, we're glad you're here. Thanks again. We’ll see you soon.