wavemaker

WEBINAR

How low-code is delivering business results for enterprise software makers

HOSTED ON
Wednesday, August 16, 2023

SPEAKERS

Chris Gardner
Guest Speaker, Forrester

John Bratincevic
Guest Speaker, Forrester

Hashim Toussaint
SVP & Division Head,
Digital Solutions, FIS

Sameer Sainani
Senior Director - Product,
WaveMaker

Sameer Sainani (WaveMaker): Welcome everyone. If you're looking to learn more about unleashing the power of low code and really ignite business success for enterprise software makers, you've come to the right place. I am Samir Sanani, Senior Director of Product at WaveMaker, and with me, I have a distinguished panel of speakers. WaveMaker, as you know, is a leading low-code platform that helps professionals rapidly build modern, scalable, and secure software products and applications. Forrester, you have all heard of, and is one of the most influential research and advisory firms in the world.

FIS is a global fintech leader in digital banking solutions and is recognized as the largest processing and payments company in the world. The format for today's webinar is that we will begin with our speakers from Forrester, and then get a real-world perspective from our speakers from FIS. We will then break into a Q&A session, and the audience can use the Q&A feature at the bottom of the screen to ask questions from the panelists. Feel free to add your questions during the webinar.

However, we will only be able to answer them during Q&A. We have several attendees in the audience, and if you're unable to get to your question, we will follow up with you after the webinar to get them answered. This webinar is being recorded, and we will send you a link to the recording for your reference.

Allow me to introduce the speakers. We have **Chris Gardner**, who is a VP and Research Director at Forrester, and leads a team of application development and delivery analysts who are focused on the future of software development. We have **John Bretoncevic**, who is a Principal Analyst at Forrester, and his research focuses on citizen developers, local development platforms, and digital process automation.

From FIS, we have **Hashim Toussaint**, who is the SVP and Division Head of Digital Solutions, and owns the entire banking solutions ecosystem, including enterprise and UX strategy for digital solutions. We are fortunate to have such an august panel, and without further ado, let's dive into the topic at hand. Chris, would you mind starting?

Chris Gardner (Forrester): Yeah, thank you. Thanks, everybody, for joining us today, and thank you for inviting us to this call. So our presentation is around low code in the digital era, and let's start off with what is digital? It's a fairly basic question, but digital means that technology is at the core of the business model, basically, so when we mean that, when we take that a step further, we have to look at software. And software is the expression of that business. It's the digital form that business takes. So policies, procedures, data, know-how, all those things have to be written into software. And then they need to be changed in real-time in order for an organization to be considered truly digital. This is an aspiration for many companies, but it's not an easy aspiration to make.

It's often been said that software is eating the world. Mark Anderson is the first person to really kind of vouch for that. But the challenge is that software demand is greatly outpaced dev capacity. And whether economic conditions improve or fall, it's always hard to find developers. In particular, it's hard to find what we call high-code developers. A high code developer would be somebody who is a C++ or C or .NET developer. It can be very difficult to find these folks. So, you often have to look beyond what's currently being offered in terms of development in order to achieve what's needed to directly address software demand.

So on top of that, there's a brutal reality, and that's most companies aren't really digital. If you look at these stats here, the surest sign is that workarounds, paper-based or manual electronic workarounds like excel or email are still prevalent in most companies. 76% of companies are still doing paper-based manual routing of tasks, and 97% are still doing electronic manual routing. So, using Excel and using email. And, this takes up more than its fair share of processes. So, what we're having a situation of is; undervaluing software. If we're going to use software, we're using the most basic manual means to do it, and it's not really the most efficient way to tackle these things. So, there's a better way.

And that better way is low code platforms. So what are low code platforms for folks that are not familiar with them? They rely on visual declarative abstractions. So instead of writing code like traditionally, we tackle it by writing abstractions in a WYSIWYG environment. And, that allows you to drop components on the canvas to make a screen or process engine that allows you to define a business process in visual notation. And, the platform will do all the automation, all the management of the code. It will do the compiling of it so that it all runs the way you want it to run as an application.

And you would do this for three reasons. One is speed, to increase development speed in particular, we find that there's a 70% average reduction on coding from scratch when building with low code, depending on the use case. So, that's a 70% improvement on speed; to time to market for low code.

Second, it's to increase the adaptability and flexibility of the software itself, both through the increased speed due to the built-in quality checks in the platform itself. And then finally, it's expanding the developer pool beyond what you were talking about earlier.

Because; when you aren't writing code, aren't using code to write software, you can have more people writing applications and more people become developers. And, low code allows even business people outside of IT to develop. And, those are called citizen developers.

In addition to that, there's a very clear benefit of low code that it's a full stack technology. It's everything in a box needed to do modern development, which is fantastic. There's tooling for data and integrations. There's process automation. There's reporting. There's user experience. All this stuff is included within the box, which makes it a great tool for building new applications; features you would normally have to add on, even like DevOps toolchains and collaboration capabilities, they're often built into the platform itself.

And lastly, governance, administration, security, and extensions, these are all baked in as well. So, they help manage risk and ensure compliance. So it's a great way to get a platform that does just about everything you need to build out modern applications without having to add additional applications or systems or platforms in order to achieve this.

Forrester breaks the market into two segments. We have the low code for citizen developer segment. This is for business people outside of IT to deliver applications. These products focus on providing a simplified development experience to appeal to the non-technical developer. And then we have low code for professional developers. And, these enable professional technologists to deliver a wider range of enterprise applications, including mission-critical apps, apps for data management, process automation apps, and all those various pieces. And these are powerful products but are often too technical for the average business person. So oftentimes, one will need to choose between these two types of platforms, low code for citizen developers, low code for professional developers.

And I'll switch over to John with a critical statistic. 71% of developers say their organizations have adopted or plan to adopt low code. That means that if you're in an organization, chances are you're already adopting low code. Talk to your developers, talk to your business people, see if they're already starting to leverage low code platforms. They probably are. And if so, you're on your way to all the benefits we just described regarding speed, adaptability, and expanding the developer pool as well. And for that, and to continue, I'll switch over to John. John?

John Bratincevic (Forrester): Thanks, Chris. I'm going to start by hitting on some data around that adoption to kind of make the points that Chris made around the value of low code and how it's being used, what its functionality is.

So the first point is this. This data point from our annual developer survey goes out to a couple thousand developers every year. And of those developers, we asked the ones who are adopting low code, that's 71%. And we asked, well, what are the use cases you're using? And you can see we give them a lot of options. And the chart isn't very slanted,

right? The bars, except for the very bottom one, which is small automations, which we threw in there just so people had something to select for small, because there's this misconception about low code being used for small things.

You can see most of the use cases are actually pretty close. And, the top use case is actually customer-facing applications, which in general, in most companies means it's a pretty important critical application. And, as you can see right in the middle, just a few clicks away is core business applications, which no matter how you define that, means the work is non-trivial. So this chart tells us two things. One is these platforms really are general purpose, because you can see these vastly different ways of building stuff are all pretty close in the percentages. And secondly, it's non-trivial, virtually by definition, because the top use case is customer-facing. And right here, not that far away is core business apps, which that's about as critical as you can get in an enterprise context.

So, the platforms are being used for a lot of stuff. And a lot of that stuff is important. Also, this is really interesting. Again, we're asking people in our developer survey who are adopting low code, why their firm is adopting it. And, Chris went through the benefits of platforms. And you can see some of the stuff there, like around making processes more efficient or making it faster, what have you. But you'll notice the two largest bars, the ones at the top, focus more on the who than the how. So in general, when people buy these platforms, while it is much faster to develop and it does bring quality benefits and so on, in general, the sort of driving impulse is more on that idea of expanding the developer pool. And as Chris said, that could be all the way up to citizen development and enterprise, or it could be even, say, for professional software companies where they have decided, we're going to adopt these platforms so we can expand our talent base. We can hire people out of college and train them to be engineers quickly on a platform and make our software product or whatever.

So many benefits, but I would say this aligns with our own experience as analysts. The ‘who’ tends to be the driving impulse in the industry, getting lots more people involved in the development process is driving adoption. And, what you can see when you get in the more grittier part of our research is that this even translates into development practices and the practice of agile.

So, most shops say they want to be an agile development shop and people apply the different frameworks like Scrum and so on. But we published an entire report on this topic, so as companies are adopting these products and going about using them, we found that these shops tend to be inherently agile because the technology itself leads to those things.

So this chart appeared in our report on the subject where we basically laid out the agile manifesto and we showed how there are emerging development practices that can only

really happen in low code that sort of embody the agile ethos in a way coding never can. So it could be something as simple as the ball on the top right.

Like, you know, you can do shorter sprints of two or three-day sprints instead of like two-week sprints or whatever, you know, which that's pretty good, you know, coming back and getting your feedback more quickly from your end users and so on. But it goes a lot deeper. There are things like deeper collaboration. So it's very common in the local world for professional developers to sit with end users, not go through an analyst, sit with end users and create an MVP or prototype something or work through a problem on the fly, right? And maybe create a whole working application in an afternoon. That's a very common practice. The teams also tend to be smaller.

So, because of the efficiency of the platforms, you can go faster and then a lot of the tools are baked in like Chris said. You don't need a specialist for every task of the SDLC. And then different members of the team can do multiple things. So it's not uncommon, even for enterprise apps and enterprises for the scrum master to be the developer and the analyst and the trainer on a two or three-person team, right? which is an agile aspiration, the fungibility of roles. Also, documentation tends to be reduced because the tools themselves tend to be somewhat self-documenting based on the way they work, their visual. And, then you get into this idea of working more with end users as developers themselves. Like may be they create an MVP or a portion of an application collaborating with a professional. Or maybe you do full-on citizen development where you're allowing end users to make what they need in a department or a business unit or whatever. So, the big point here is that the technology itself allows for better development practices and a pure manifestation of what agile is; which I think most companies say they want to be.

And, if you extrapolate this out and you look at the data and you look what people are doing, there's a very interesting thing happening, which is the nature of development at least in low-code shops and the nature of a developer is shifting. So, on the one side, you have kind of the citizen developer phenomenon which is when business people start developing software. And, I think people kind of get that, right? You add technology to your business skills and now you can make apps, right?

And they start working their way up the spectrum towards being a really good developer in addition to whatever their business skills are. But then there's another interesting thing which is on the technologist side, the professional developer or the IT person who doesn't have to be IT, just the technologist; applies low-code and learns it.

Their job starts changing too. So, kind of consistent with those points on agile, we have a bunch of data on this and a whole report on this topic. If you look at the professional

developers, okay? So, technologists who adopt low-code and you look at all the data on them and compare them to coders, we summed it up by saying they're business-focused and technology-advanced. So, they're way more concerned about like the business outcomes of software, right?

So not how many story points did I complete or whatever, but like did we save money or improve customer experience or that kind of thing? They really care about that stuff and that's how their teams get measured and so on. But they're also not the dummies. There's a misconception about developers that if you're using low-code, it's because you can't do better or it's for trivial things. We already killed the trivial thing myth. These developers, the ones that are using low-code are actually the ones working on what you'd call advanced use cases or using advanced techniques or tools.

So they're much more likely to be building on microservices, which is a complex thing if you know anything about it. They're much more likely to be using container orchestration. They're much more likely to be building IoT apps, all of the discipline instead of tech and so on. They are business-focused and they tend to be doing a lot of difficult, cool things. So if you net out both sides of this spectrum, what's happening is it's blurring. And, this idea of the low-code developer is this business-first, multidisciplinary digital problem solver, which is exactly what we've always said we want engineers to be. We always say the best engineers are the ones who know a lot of things and can make those trade-offs and accomplish the software solution.

And, that tends to happen in low-code because of the nature of the tooling itself and all the techniques we outlined. So what does this mean? There's a lot of benefits, obviously. Faster is good. More developers is good. More adaptable software that you can change as you need it and not break stuff, all good. The changing nature of development, and the agile techniques, all good. There's a big impact this has when it comes to software sourcing generally. It affects build versus buy.

So again, we wrote a whole report just on this topic. It wasn't just about low-code. It was about different technologies and how they're affecting build versus buy. But low-code was kind of at the center of it. And there was a quote during that research where we spoke to an architect at a very large firm and he pointed out that the boundary between build and buy is blurring massively. Right?

One of the reasons it's blurring massively is because the basic economics of getting bespoke software out the door, of writing the applications you need and changing them has changed radically with things like low-code. Like it's just a very different algebra to create a custom software solution now.

It's much cheaper and easier. It's different economics, basically. And, when you combine

that with the desperate need to be digital, of digitizing everything all the time, like Chris said at the beginning, that means the way you look at build versus buy has to change. And, you should use low-code to help you accomplish that.

So, in the course of that research, we interviewed a whole bunch of people, looked at a whole bunch of data, and we found out there were four assumptions that most companies have around build versus buy that are no longer quite right. So, they think this, but they're wrong.

The first was that most companies or many companies underestimate the complexity and cost of buying an off-the-shelf piece of software and implementing it. And, I feel almost embarrassed saying this like we should have learned this by now. We had many examples where in the real world, the supposedly easy off-the-shelf solution was much more difficult and much more expensive and much less suitable in the long term than the bespoke solution.

Like that's the reality, is the assumption that off-the-shelf will be better and easier is often not the case in the real world. You need to change it at a minimum and change it easily, and that's often hard. The second assumption we found that is flawed was that many shops overestimate the cost and complexity of creating new software. And, because of things like low code and cloud platforms generally, that sort of reflexive concern that creating anything customer bespoke will be too hard or too expensive, it's no longer true.

Chris cited our rule of thumb earlier about low code, for example, is about 70% faster to get something done on average than coding. But think about all these other economic levers we talked about, more developers, right? A collapsing of roles where people are fungible, where you don't need a separate analyst from developer or designer or scrum master or whatever, right? So, both because of the people who are involved, because of the techniques that are emerging, and because of the technology itself, the basic economics of app development with things like low code, are completely different. It may be much faster and cheaper than you realize. It may even be much faster and cheaper than buying something. And then the last two, these assumptions that are wrong in the digital era around build versus buy, we found that a bunch of firms underestimate the value of platforms and ecosystems. So, they treat each software need as a discrete need. That's something that has to be set apart, and then we got to find the exact solution for this particular problem.

But there's actually a lot of value, the correct way to think about it is there's a lot of value in having a platform with a shared experience and shared security and shared administration and skills leverage and development and that kind of thing, with their ecosystems of maybe pre-built solutions or whatever to build on.

But, that's an important sourcing consideration that most people omit.

And then finally, the really big thing is if you really understand the meaning of digital, then the usual assumption that all businesses are basically the same as just a few differentiated details where they're different breaks down immediately. Because, if you turn everything about a business into software and then change it in real-time, that means every business is actually really unique when it comes to the software they need. Everything is at least somewhat custom because otherwise, you end up with all those analog workarounds or poor analog metaphors like Chris was talking about. And that means you're not digital. So you need to lay aside the presumption that all businesses are so similar that you can just assemble this architecture of off-the-shelf stuff and you'll be good. It doesn't work in the real world, and real digital companies don't do that.

They digitize the heck out of everything, and that means bespoke. So, you got to make it. Thus, our assertion now, and this goes far beyond low-code, low-code, as I mentioned, is at the center of it, is that companies don't have a build versus buy decision, where you've got build on the one side, encode from the primordial soup, and buy on the other, just this nice, happy solution. We'll just take it off the shelf and install it. We'll all be happy, and it'll be great. Often doesn't work that way, especially in big companies.

The real decision you have is customize or compose, i.e., you start with something that is made, but it has to be a modern application that's architected for change, i.e., for example, being built on a low-code platform. So you can change it easily after you bought the thing that you needed. You can continue to change it and keep it digital so you don't end up accruing these workarounds at other places.

Or you can compose, not code from scratch, not necessarily. That often isn't necessary for a lot of things, but using stuff like low-code platforms, and of course, going to other sources, components, and so on, compose the solution you need from pieces, which is much faster and easier than coding from scratch, as we articulated earlier.
That's really the two options you have, and there's kind of a spectrum in between those two options. And, that's how you should think about any software sourcing decision. And, low-code is a big part of it. And, our other assertion out of that research is because of low-code and some of these other things, and because of the need to be digital, that in general, the pendulum is swinging.

That the need for bespoke software is becoming more common. And the closer you get to the customer, the more important that becomes. If you were to go read the report on this, you can see that when it gets close to the customer, it's very acute. Things like e-commerce, where there's a lot of differentiation, has a lot of real and immediate value and impact on the customer. Pretty much all those are custom projects now.

Nobody goes with off-the-shelf suites anymore. But then you go back in the customer operations, all the business stuff that supports customers, all the stuff behind the scenes that we do in every company. Again, traditionally, a lot of people thought of that as very generic. The reality is, to be digital, it's extremely bespoke. And that is swinging towards custom in many cases.

There's a lot of low-code work there. But, even when you get to core applications, recall that chart I pulled up earlier that showed core applications at 34%. We have this idea in our minds that those can be kept vanilla and never changed and protected carefully. But the reality is, sometimes they got to change. Stuff like billing processes, they got to change. And that's part of your accounting system. So you got to change it. Because otherwise, you'll end up with all these workarounds. And that's what you see in every business. So in general, we see a lot more custom development in companies and less off-the-shelf sourcing.

But again, it's made possible by these modern technologies that make it more economically viable. And in this context, we've basically written that in the digital era, there's five pragmatic solutions for modern delivery on the customized and composed spectrum. I won't go through them all. But the point here is that low-code is the one in the middle.

So it's certainly not the only one. I'm not saying that. But it is a very pragmatic one. And we see this play out in a variety of industries and contexts and use cases. Just so it isn't all theory, I want to give you some examples of where low-code has directly affected build versus buy. So these are actual case studies we've studied. Some we published. Some we haven't.

I'll start with the banking one. I have seen both ends of the spectrum in terms of delivery personas dramatically affect build versus buy decisions in banking. One example is a pretty well-documented case study about one of the large European banks. And, they basically had a large core modernization project they needed to do. Very painful. And, they evaluated off-the-shelf software, some of the name brands you're probably familiar with. And, they evaluated coding from scratch. What they settled on was using a low-code platform.

And, the net result of it was, it was better, faster, cheaper. Significantly cheaper than off-the-shelf. It didn't have nearly the risk of coding. It was a fairly short delivery time given the size of the use case. There was a big mainframe involved and so on. The net result was they modernized 70% of their core.

They retained the mainframe for some of the record-keeping going back decades with

pretty much all the business logic and processes and experiences and channels. Everything else was built on low-code.

And they got so good at it that they got different financial products. So, they created new channels for selling stuff to their customers and being able to get into new businesses quickly. And, they got so good at it after that that they started selling the technology to other banks. But the crux of it was they decided to use low-code to compose what they needed rather than buy an off-the-shelf package or code it from scratch. I've also seen citizen development programs. That one was done by a professional shop as you'd imagine given the significance of it. I've seen citizen development affect build versus buy in banks too where they have hundreds of people developing apps and they just start buying fewer SaaS apps and rolling more of their own.

Manufacturing is another really interesting one. There's a wonderful case study of a luxury goods manufacturer that bought a manufacturing execution system and tried to roll it out for three years and it was one of the best on the market. And then they gave up because they couldn't make it behave the way they needed. They couldn't be digital in it. Every plant wanted to continuously improve their processes and they couldn't because the software worked the way it was. And it wasn't easily changeable no matter how good it was. So they threw it out. They gave up several million euros. And they bought this platform at like 10% of the cost. And they had the new manufacturing execution system across all the plants written by the people who knew what was going on.

It was the process engineers and the line supervisors and the kids out of college who were doing continuous improvement. They're out of college and they're learning the business so they go to the plants. Like those personas, people who are out in plants who really knew what was going on.

Citizen developers wrote this core system across all the plants. And, it was better and it was faster and it was cheaper and it was democratized. And, it was this merging like I talked about earlier of the developer who also understands the business. And, now basically every plant's a little software factory. They all have their own software practices where they prioritize their own needs and continuously improve in software as dictated by the manufacturing experts. So, democratized implementation, this is very interesting, and actually affected a core system sourcing strategy. Very interesting stuff.

And there's other examples we could go through. So here's our recommendations in that context. And these recommendations apply almost irrelevant of the industry. They're generally applicable.

The first is you should add low code to your landscape. If you don't, you're behind.

It's table stakes at this point. You can see it in the data, but that's also our view and our analysis. The second is that you should include compose as a key option in any software sourcing decision. So, don't just reflexively assume you should buy something off the shelf. Right? Buy applications and suites that are modern and highly adaptable and a close fit for your needs. That's fine at a price you're willing to pay. So they check all those boxes. They're modern. They can change easily. Right? And, they're not overpriced. That's when you buy something. But otherwise, perhaps you should consider composition because it's a very viable option. The economics of this have changed.

Also, find more developers. You're going to need them. As we mentioned, low code is a practical way to find more developers because you can expand your lens of who can be a developer. We also recommend federating and democratizing development work on local platforms. We didn't get to this too much in the presentation, but this is a major trend where development is being pushed out in companies and more and more people are becoming involved. It's a formal strategy. And then finally, I mean, the big point of this is to become digital. Turn everything about the business into bits and bytes and change those bits and bytes in their real-time. That's what being digital means in the real world, in a practical, concrete way, which means express the business and software. That's the point.

And low code is one of the very practical, pragmatic ways to do that. So with that, I think we'll hand it back.

Hashim Toussaint (FIS): All right. I think it's over to me. Thanks, John. Can you see my screen?

Chris Gardner (Forrester): Yes.

Hashim Toussaint (FIS): Perfect, perfect. Well, good morning, everyone. Thanks for joining us. So, my name is Hashim Toussaint, I am the general manager and division head of digital solutions at FIS. Digital solutions is a business segment that sits as part of the broader banking solutions business within the company. For those of you who are not familiar with FIS, we are one of the largest fintechs in the world. We provide banking and capital markets, products and services to financial institutions. We also have payments processing as a business under the WorldPay brand. And, as some of you have seen in the news, we're actually spinning off WorldPay into its own company.

So, today I wanted to share how we're using low code, the benefits we're starting to realize, the areas of opportunity we see, and then where we're headed with the technology. So, I just want to give some real perspective in terms of how we're thinking about it and using the local tools within the company. So, as it relates to digital solutions,

right, and digital solutions business, it includes a set of Digital One products, which provide mobile web banker and teller solutions. And also the business includes our CodeConnect product, which is a platform that exposes our banking APIs to our clients, to third parties and to other fintechs as well. As it relates to our use of low code, we're using it in a number of places across the company, including retirement and our card businesses. But specifically to my business, we have used it in one of our newer products across in digital one. Digital one is a suite of solutions that is about eight years old, and we have over 600 institutions using one or many of those products.

But, first I want to start by taking a step back in terms of looking at our journey of the company. So, FIS, we have been on a journey and where we are today is there is an intense focus across the company on delivering with speed and quality for our customers. We are doubling down on client-centricity and we are accelerating innovation. So, our use of low code where it makes sense is one of the many approaches that we're taking to accomplish those goals more broadly.

So again, just want to set a bit of a context there for us. So let's zoom down a little bit further. So specifically, we are using low code with our consumer studio product. Consumer studio is one of our newer digital products geared towards our largest clients. It is more of a one-to-one model that allows our clients a bit more customization than they would get from our one-to-many solution, which actually is not represented here in this view. Our journey with low code began about three years ago with one of our partners, with a partner, Wavemaker. And, so, we've been working very closely with them throughout the last three years along this journey.

So in terms of, you know, so; how are we using the technology? Right. So, within Consumer Studio, we have built a number of widgets and modules that essentially are made up of certain components. Right. So, if you think about the types of products we build for our customers, which are digital products, there is an element of a UI, right? There is a set of APIs and there is there is data and then there are services to support the overall solution.

And so we've built widgets and modules that essentially package those components together for reuse across the organization. And so if you take a step back and look at the customer journeys that we support across the business, almost envision, you know, those journeys being comprised of many different widgets and prefabs stitched together to actually drive an outcome for the end user.

There are a number of benefits we have seen by going this route. So we have seen much more of a seamless and faster integration of the various front-end components to the back-end services that support them. And, we also have seen from a development

perspective, a lot more ease in terms of binding data elements to those back-end services. So in the past, you've got to sort of code it every single time.

And, so now our developers can effectively take the various modules and drag and drop into the new experiences as needed. One of the other benefits we've seen, obviously, that leads to really lots of time and cost efficiencies in our development cycle. We have also leveraged quite a bit of reuse as well, which is important. So as we think about our broad ecosystem, we can reuse and repackage many different modules and components and other parts of the business, which, again, going back to our focus around delivery and execution and speed certainly helps in that as well.

One of the other benefits we've seen out of the gate was the ability to have a single code line, regardless of the form factor we're surfacing our experience through. So web, mobile, you name it, right? One code line, which, again, is really important as you think about unified experiences moving quickly and again, building in a way that's flexible for the end user. One of the crosscutting benefits I sort of mentioned this before, but again, we can take components and modules and the prefabs and extend those across many different businesses and channels as well. So, for example, you can have a login prefab or module that you can use not just for Consumer Studio within digital, but also within Wealth, within Payments and within any other sort of business or use case you're trying to accomplish.

So, connecting the dots a bit from the previous slide. So, one of the things we are accelerating here is we're going to be leveraging what we call experience as a service.

So, if you think about, again, those customer journeys we have to support and then the suite of prefabs and modules we have available to us, we can effectively allow composers who are internal users. So John talked about how do we extend what we call sort of development. We can federate it out more broadly to the organization. So we can have essentially business users where it makes sense. We can expose it to our clients. So they and their SI partners can leverage as well. Citizen developers, again, other users across the business. And, so you have a set of prefabs and widgets that support various journeys you can pick from. And, you can plug and play and really create lots of bespoke experiences depending on your need and depending on the end user need. So, again, it's all about reusability. It's all about leveraging the UI, the API components.

It's about slicing them by logical business function and then also slicing them by the journey and the end user outcomes you're looking to get to. So, again, really, really flexible for us in that way. Now, there's quite a bit of upfront work to actually code the modules and components. But once you get there, again, it's really about customizing where you need to and plug and playing with the right components to build out the experience. So, where are we heading? The two items on the right are actively being

worked on within my team and our teams.

We are establishing the first LCD Center of Excellence. John talked about some of the trends out there in terms of companies using the technology are actually getting a bit more serious about how do you put some guardrails and frameworks around how you use a technology. With any technology, we have to ensure that we have the right structure. And, the focus is to make sure we maximize the use and also ensure that we have flawless quality when the products get out into the hands of our customers.

And, it starts with just proper development of the widgets and the prefabs and the governance and structure around those. So, we're going to be putting a lot more focus on creating best practices within our Center of Excellence. And this will be over time a firm-wide initiative for us.

Secondly, we owe it to our customers to raise the bar in terms of modernizing and delivering new capabilities. So we're leveraging low code as we execute on a number of strategic digital initiatives that are underway. And so we're really assisting in the transformation journey, allowing us to, again, move faster and really compose new experiences that are meeting the needs for our customers.

The other two items on the left are places where we think we want to go and will be assessing in the near future. To summarize them, I would say that at some point we want to expose our widgets and prefabs to clients and allow them to self-serve and compose more bespoke experiences for themselves. The other item that is a potential for us is exposing widgets and prefabs in various marketplaces and extend those externally to developers and make it really easy to to leverage what we have, but also, more importantly, connect into the FIS ecosystem. And then last but not least, we want to get to a place where we can and John touched on this. We again, we can expand to much broader personas within the organization. Right. And so, again, where it makes sense, allowing much more self-service and allowing our teams to non-technical teams to compose new digital components that can be extended to our customers.

So with that, I think so. That's the end of my presentation. Samir, I will hand it back over to you. I think we're going to jump into a bit of Q&A at the moment.

Sameer Sainani (WaveMaker): That's right, Hashim. Thanks so much. Thanks very much to all the speakers. We will now open it up for questions and answers. Please enter your questions in the Q&A window and we will pick these up as they come up. And then while we wait, I have a question for Forrester. Based on your research, what is the impact of AI on local development? This is something that comes up a lot. So I'd appreciate if you could shed some light.

John Bratincevic (Forrester): Sure. So we're working on this one right now.
OK. So there is no published formal opinion. One thing I can say is I think in the short term, every low-code vendor I know of is adding, whether they're bringing in their own open-source model or they're using one of the large models APIs, they're adding some sort of natural language to application functionality. Right. So, you don't drop right into the canvas. You can put in a prompt and then it'll render the application, the components. So there's going to be a big shift in terms of the experience in these products, potentially, based on all that work. Right. So that's definitely happening, just like it's happening in other parts of software.

And then we're adding it as pre-baked connectors and all that for putting the functionality in your apps as well, as you'd expect. I think also in the short term, if those features are strong because only a few companies have released them already, it will probably help the industry a lot, especially on democratization. Because, what we've seen with smaller releases in the past is, I don't know, think of even very trivial things like you take a picture and it turns into a screen. A lot more people just started using the platforms, right, because maybe they weren't interested before, but maybe that removed; that last barrier. They could try it out. And then they got the success. So, they were motivated to continue. So I think the companies that do it well and get it into the hands of their clients, their user base will probably grow a lot because we've seen that before.

And this is much more substantial. Longer term, it's kind of an open question because AI kind of starts making you think architecturally; like, OK, well, low code versus code. And then there's natural language going to become the core mode of interacting with the tools. And then, so that's like the long-term impact.

And like architecture generally is kind of the thing I don't have a foreign philosophy on, but in the near term, every vendor is working on it. And I expect there'll be a lot of near-term adoption by less technical people. Make sense?

Sameer Sainani (WaveMaker): Yep. Thanks so much. We have a few questions that have come up. We have a question about what are the governance mechanisms that the COE must put in place? Can someone speak about the governance?

John Bratincevic (Forrester): Yeah. So there's a couple of things. First off, you want to choose your structure, right? And then from your structure comes certain implications. But the most common structure is what we call a federated structure where there is a COE. But then the COE isn't the only group involved. There are coaches or evangelists or some sort of next layer down that spread throughout a company, right? They're often called coaches.

And then there's all the end users who can now become developers. You have kind of a three-tier structure. So that's kind of thing one is you got to sort of ferment that over time. And we have case studies of how firms have done that in significant scale. And then the second thing is you have to distinguish between use cases. So what a lot of developers or a lot of technologists, they make a mistake where they see all development is the same, right?

They think they have their SDLC. It works the way it does. And all development must work that way. And that's not how citizen development of scale works. You have to say this class of use cases has this level of rigour. But this class of use cases has this level of rigour, right?

Because you're talking potentially hundreds or thousands of people building hundreds or thousands of things. So, you have to distinguish between those things and have different rules and processes for them. For example, no process for very small, low-risk stuff, right? Because otherwise, nobody will participate. Then the third point is once you kind of have that org structure side and the rules you want to put in place, right?

And then you've iterated on them and figured them out. You really, really, really want to automate them. It can't all be based on training and audits and documentation and things. And, there's a big DevOps component to that because pipelines can do a lot of this where they can look at the metadata of what's being built and then apply different business rules, right? But it's also just often just configurations in the platforms themselves. So, like rule-based access control saying you can or can't use certain components or have access to certain data or do a certain task or what have you. So, come up with a structure. Federated is the one most companies do at large scale.

Don't treat all uses cases the same. There's going to be a lot of art and work on that and how different use cases get treated differently as lots of software work gets done. And, then the third is to automate the heck out of it. Don't make any of it sort of training or manual intervention dependent because that's not digital anyway.

Sameer Sainani (WaveMaker): Makes sense.

Hashim Toussaint (FIS): I think to add to that, I think one of the really tactical pieces when you get down to how you structure the prefabs of modules. One of the things we certainly focus on is around documentation and ease of use. So again, making sure that as you're publishing to the broader suite of prefabs, all of the components that allow it to be accessible, easy to understand, have to be there. So again, we're really focused on that foundational sort of view of good quality documentation that helps further use the components across the organization.

So again, certainly John talked more 10,000 foot view, but I think there's some real tactical things we're hoping to accomplish from the COE as well.

Sameer Sainani (WaveMaker): Awesome. We have a question about examples of UIs created using low-code tools. Are there any publicly available examples of apps created? I guess we could follow up on that with examples, but there's an interesting question that's which is low code development seems like a good fit for standardized domains like websites and mobile apps, et cetera.

But do we think it will be a useful tool, non-standard, that is highly specialized software environments? Maybe the panel could comment on that.

John Bratincevic (Forrester): If I understand the question, I think I agree, right? Like the box of components is basically around web and mobile apps, and there can be a lot of depth and sophistication on that, but we're basically talking business, web, and mobile apps. We're not talking video games, right? Now, there are examples of platforms that are even in this domain that could be used to make video games. It's almost like hacking them, but that would be a very specialized domain that is not normally what we're talking about, right?

There is the equivalent of low-code in those video game engine builders, right? They're essentially WYSIWYG, but we're talking business, web, and mobile apps, and potentially maybe desktop fat client, a few companies do that as well. So if that's what the question is referring to, like really totally different types of software that don't fit that, yes. You're probably not going to write a word processor in one of these. Yeah, that's how I would describe it.

Sameer Sainani (WaveMaker): Makes sense. Another question that's come up is, what are the drawbacks, challenges of low-code versus traditional coding?

Hashim Toussaint (FIS): I can probably jump in there. So, I would say less of challenges, but more of just where we are in the cycle. Certainly, as we think about scaling, the use of the product, right? And the tools, when I think about things around testing, right? We want to be able to not just only develop within the current product but also test it end-to-end as well. And so, we've worked through some challenges there on our side. The other piece I would say is that you think about sort of the tool and product that we use and the upgrades that we do there, the more scale you have in terms of prefabs and products that are sitting on the platform. We're being very, very cautious. So when those upgrades happen, there are lots of folks who are focused on it. So you see your sort of scale go up in terms of the team.

We're trying to obviously over time to get better there, but there's a lot of focus to make sure that nothing breaks and nothing goes wrong. I think for now, it's heavy operational when those events happen, but the goal is over time for that to become a much lighter thing. So again, I don't think it's really a challenge.

It's just where we are in the cycle and where we're trying to get to. So we've just been really cautious and focused on things like upgrades and looking forward to being able to do more of the end-to-end development and testing all within the same platform as well. So, those are the two things that sort of come to mind that I would call out.

Sameer Sainani (WaveMaker): Makes sense. We have a question about, okay, I'll get into it. Isn't there a danger of opening up shadow IT again with data getting created in silos across the firm?

John Bratincevic (Forrester): It never got closed. You've got shadow IT whether you know it or not. So the most conservative firms in the world that are the most concerned about shadow IT and invisible applications are often the ones asking me about how to do this because they realize the only way to mitigate the risk is to put things on a managed platform. So, the useful metaphor I heard from one architect was I realized that a black market is worse than a well-regulated open market.

So, the way you should think about this is shadow IT exists. You cannot possibly stamp it out. It's impossible. And as such, it's a risk that needs to be mitigated practically. And the most practical way to do it is actually to give an acceptable way for those people to make things. And it makes it visible. You have a platform where you can see what's going on. You can put some rules around it.

You can create that federated structure with the relationships where people are helping and coaching. And I'm not saying that in some empty fatuous way. I mean, I've seen this happen in the real world where they've really transformed the relationship between IT and business and they have addressed shadow IT practically. But it's basically accepting you can't get rid of it.

So how do you rehabilitate it? And you kind of work from that. That's kind of a starting philosophy. But actually, I'll add one more thing. Yes, you can make the problem worse if you do it badly. I will say that. You could just give everybody the tool and say, yep, we did citizen development. And then they end up building 5,000 things you didn't know about. But don't do that.

Hashim Toussaint (FIS): Yeah. Yeah. I think this is where the COE right and the governance comes into play. Um, rules of the road will be really critical to making sure

to your point, it doesn't go off the rails.

John Bratincevic (Forrester): Which means you're going to have to eliminate friction.

So everybody participates, right? It can't just be the rulemakers coming saying we're from the government. We're here to help because then they'll ignore you. So there's a change. But like I'm saying, this isn't theory. We've seen people do it and they go through it. They do the work and it has a good...it mitigates that risk and it brings business benefit.

Hashim Toussaint (FIS): That's right. Yeah.

Sameer Sainani (WaveMaker): Thank you for that. We have a slew of questions that we can't get through them all, but I'll just pick a few.

Wondering if this group has thoughts on monetization, pricing approaches that successful application organizations using local development ought to consider or pitfalls to avoid?

Sameer Sainani (WaveMaker): I wonder if that's something we can comment on.

John Bratincevic (Forrester): It's a hard problem in general for the industry because when a tool set can do so many things, a simple pricing structure sometimes doesn't work for all those things. Right? So vendors across the board will try different things out. Sometimes the most common pricing is per end user. Sometimes that works great. Sometimes there's a lot of friction, right? Because of the nature of the users you want to extend your applications to and so on. So it really depends on what you're going to do with it. In general, there is a slight trend towards consumption kind of pricing in the industry as some vendors run into the issues with the end user and they'll be like, okay, it's based on consumption of these services or number of apps you build as a proxy for value versus number of end users and things like that.

So it really depends on who you're going to open it up to. Right? And what use cases you know you're going to build. And that will narrow down the list of models that work for you pretty fast.

So there's no like general advice. When you go wall to wall, things tend to average out. But there's all those transition stages before you're wall to wall. Right? So it kind of depends on what you're starting with. It would really depend on your situation.

Sameer Sainani (WaveMaker): Right. Thanks for that.

Sameer Sainani (WaveMaker): We have a question on generative AI. So as generative AI and local development gain traction in various industries, including banking, what ethical considerations should guide the adoption of these technologies within the context of banking security and regulatory compliance?

John Bratincevic (Forrester): No opinion on that. That's a big one. Yeah. That's a heavy question.

Sameer Sainani (WaveMaker): So I'll skip to another question. Maybe we can come back to that if you guys have any thoughts.

Sameer Sainani (WaveMaker): You talked about build versus buy shifting. Are you saying enterprises should stop buying applications?

John Bratincevic (Forrester): No, no. But you shouldn't reflexively buy them either. Right. Which is, I was trying to get that across in the recommendations. Buying is fine. Buying is good. If it's something that fits what you need, actually fits what you need. Not we buy because that's safer and better. And then it can change. So you can be digital. So everything is rendered in the software versus accruing in all these other places. Right. So that was the idea of something that's easily customizable by design.

Maybe it's built on low code, for example, where that's baked into the intent. You're not just buying a big suite and it works the way it is. And, if you want to customize it, it's really hard to do. And then you break it and all that stuff. But something that is intended to change that fits your requirements. And, is at a price that's reasonable. Right. Then buying is great, obviously. But all those boxes should be checked. Don't reflexively buy things because composition is a really practical alternative, is the point. Because you want everything in software.

Sameer Sainani (WaveMaker): Makes sense. There's a question for Hashim.

Sameer Sainani (WaveMaker): What were some of the challenges in implementing a low-code approach?

Hashim Toussaint (FIS): Yeah, I think initially, certainly just getting our teams trained up. Right. With the new tool and the process. I think we're in a much better place now. Again, all of the work that we're putting forward in terms of COE, I think, are learnings from that period of time. But yeah, I think for us, it was a huge learning curve. In many ways, it's a sort of a mental shift in terms of how we think about software delivery. And so, certainly that takes some time to do as well.

But again, we've been on this journey for a few years now. I think initially slowly. And I think in the last six months or so, really accelerating it. So again, to me, it's just more of the growing pains of using a new technology and scaling that very quickly.

Sameer Sainani (WaveMaker): Awesome. Thank you so much.

Sameer Sainani (WaveMaker): Thanks, everyone, for your attendance. We appreciate it. And this is all we had time for today. There's a bunch of questions that are still on there. We will get back to you on those individually.

Please keep an eye out for the webinar link. And if you have any other questions, you can actually reply to that email, and we will get those answered as well. This has been a very rewarding webinar. And, I thank the speakers once again for their time.

And with that, we'll sign off. Thanks so much.

About **WaveMaker**

WaveMaker is the most open, extensible and flexible Low-code Platform that complements your enterprise application delivery while keeping in mind therequirements of Software Developers, Citizen Developers/Business Users, IT Architects and CIOs.

Write to us at **info@wavemaker.com**

