TECH REIMAGINED
AN ENDAVA PODCAST

What's at the core of efficient software architecture?

00:00:01
Speaker 1: Tech Reimagined. Redefining the relationship between people and technology. Brought to you by Endava. This is Tech Reimagined.

00:00:11
Bradley Howard: Hello and welcome back. I'm Bradley Howard, and you are listening to season three of Tech Reimagined, the podcast that explores how technology influences the fabric of our society, changing the ways that we work, we live, and we do business. Every week I speak to the leading lights of the technology world and dive deep into the biggest topics in the business. You can find us on every podcast platform that matters. Today I'm joined by our CTO or Chief Technology Officer at Endava, Eoin Woods. Hello, Eoin, how are you today?

00:00:40
Eoin Woods: Hello, Bradley. I'm very well. How are you doing? It's very nice to be here.

00:00:42
Bradley Howard: Thank you for joining us. So as well as Endava's resident technology guru, Eoin is the author of a couple of books and many articles on software architecture. As both a researcher and a practitioner of technology architecture, Eoin has a dual insight into the discipline's evolution. His recent book, called Continuous Architecture in Practice, is a guide to a specific, recent approach in software architecture, written for practitioners and advanced level students. Today, we're going to talk a bit more about some of the themes that are explored in the book, alongside Eoin's wider perspective on the discipline.

Eoin, thanks so much for joining us. If we can start diving straight into it, so you've been writing about software architecture for almost 20 years now, and practicing it for even longer than that. So in that time, how have the core fundamentals of architecture changed?

00:01:31
Eoin Woods: That's a great question. The key thing really that's changed is the context we practice architecture in. When I started doing software architecture, which I suppose was in the late 1990s doing it in earnest, we had to make a lot more decisions about the system up front. And that's for two reasons. Firstly, we didn't have as much evolution and change to the system's requirements, and constraints, and goals during the development and deployment of the system as we do today. We tended to fix an idea, we decided what it had to do. We delivered it incrementally, but none of the fundamentals tended to change very much.

The other thing that's happened is that the technology we now have, such as cloud computing and modern languages, the fact that we can distribute systems without really even thinking about it, the fact we have a lot of technology, especially from the Internet giants, which allow us to build flexible, scalable systems means that we've got reusable solutions, which are in themselves flexible. The industry's learned a lot of lessons about that.

And the third thing is, we reuse a lot more than we used to. There's an awful lot of open source, there's an awful lot of very powerful cloud services that we get immediately as soon as we deploy on a cloud platform, that there's an awful lot of reusable shared knowledge in the industry. So all that's a slightly different context.

So what's happened is that architecture needs to evolve, too. Today we need to build digital platforms, not one- off systems. These things last a long time, but it's very hard to predict what it's going to be used for in the long term. My canonical example is Google Maps. I'm quite sure when they sat down and the geniuses started hacking out the initial C++ with Google Maps, they didn't think, " You know what we need? A really good way of finding restaurants." I think that was probably a very long way from their thoughts when they were thinking about map data, routing people, annotating it with weather or whatever they were doing in their first application. And Google Maps is a great example of platform application. It provides some fundamental facilities based on big complicated data sets, highly connected to other things. And then on top of that, they've provided APIs to their services and lots of Google teams have built all kinds of creative things on top of it.

So that means that architectures have to become much more incremental, because we know a lot less when we start out about where we're going. So therefore the architecture process has to be one that is much, much more attuned to discovering and evolving as we discover more, rather than making a lot of decisions at the front and setting a very firm direction, which is then difficult to change later.

00:04:08
Bradley Howard: And will those principles continue to shape how software architecture will change in the future as well, do you think?

00:04:14
Eoin Woods: I think so. I don't think everyone agrees with me, but certainly Murat Erder and Pierre Pureur, who wrote the original Continuous Architecture book, and then I joined them for their second book, Continuous Architecture in Practice, the three of us, and actually, we find many other people we talk to in the industry. Gregor Hohpe would be another person, I think, who views this as very much how it happens, the team at Thoughtworks, people like Pat Kua and Rebecca, their CTO, and Neal Ford, I think with their book Evolutionary Architecture. There is a swell in the industry towards viewing architecture as something that fundamentally is about evolution as opposed to fundamentally making all the big decisions up front. Which frankly, when I started, that's very much what architecture was seen as.

00:04:56
Bradley Howard: So has architecture moved alongside agile software development in that way?

00:05:02
Eoin Woods: Yes, very much so. I think really what happened was that architect was in denial for about 10 years. The agile development not even existed really, but when they admitted it existed, a lot of traditional architects went, " It's a terrible idea, that'll never work." And then over the last 10 years or so, this has been rapidly dawning realization, not only does it work, it works really, really well, and it's the best way to develop software.

So some people are still very resistant to this, but many, many architects, especially architects coming in the last 10 years, can see that what they do has to be fundamentally aligned with and embedded into agile delivery. So that needs a rethinking of a lot of the architecture thinking that went before. Not so much the individual techniques, not so much the goals of architecture, not so much the fundamental approaches, but how architecture work is actually performed on a day- to- day basis in a project.

00:05:54
Bradley Howard: There's several different types of technology architects nowadays. I think you wrote an article about it in 2014. Don't mind, I'm not going to pick on some minute piece of detail

there, but the different types of architects are enterprise architects, infrastructure architects, application architects. Which one do you think you are more closely aligned to, and why?

00:06:14
Eoin Woods: Personally, I think I'm very aligned to application architecture. I'm probably 80% application architect and 20% enterprise architecture bluffer. I've done a fair bit of enterprise architecture over the years, but really my expertise, and really what I enjoy, is the architecture of software applications.

00:06:30
Bradley Howard: Do you think there's a personality difference between the three of them? And that's a slightly loaded question, because if I think about the three different types of Endava, there's a very clear personality difference.

00:06:40
Eoin Woods: Well, I think there's an old joke, isn't there, that EAs think that they rule the world, application architects think they're the center of the world, and infrastructure architects know that they actually run the world, because they underpin it all.

I think there are slightly different characteristics. It's sometimes experience- based. EAs tend to be very experienced, have a huge historical perspective, but also be further from the day- to- day reality of writing software. Application architects tend to be very close to that but often don't know a great deal about the infrastructure, just to classify them all perhaps unfairly. And then infrastructure architects tend to have been in a different professional career path, so tend to be the people who are always thinking about what's going to go wrong with the infrastructure. So yeah, maybe they're not fundamentally different people, but I think they do behave in slightly different ways.

00:07:27
Bradley Howard: How much do each of them require some soft skills, as well? Things like people management, et cetera. Because essentially, I've always thought of architects, not just software technical architects, but even building architects, et cetera, as trying to sell a bit of a vision, a bit of a dream, into the rest of the organization or to eventually the rest of the scrum team, to say, " This is where I conceptually see that we're going to be heading. This is my dream and how to bring them on board."

00:07:56
Eoin Woods: Yes, I think soft skills are terribly important for architects because as you say, the whole point of having architects is to set direction, align, unify, understand the need, make sure that the disparate activities of all of the different teams are aligned in the same direction, achieving the goal. Without good people skills, it's very difficult to do that. You need to be able to understand complex problems, you need to be able to simplify them and communicate them. You need to be able to empathize with people who disagree with you, perhaps for good reasons. And you need to be able to work very cooperatively with people. So yeah, soft skills are very important.

As you go up the, if you're at the conceptual levels, if you go to be EAs, soft skills are crucially important for them because they tend to be working with senior non- technical people, stakeholders, as we would call them. Infrastructure architects arguably don't need quite such great people skills. It's more of a technical job. But they also, as you say, they need the skills to sell the vision, to sell their decisions, and to constantly communicate to people why the infrastructure architecture is the way it is and how to work with it.

00:09:04
Bradley Howard: How do you promote best practice across the different architects within Endava, then?

00:09:09
Eoin Woods: Within Endava, we organize our technical staff into professional disciplines and we have 12 of those disciplines, such as development, such as testing, such as creative services, where our UX and graphical design people, for example, are. We have an infrastructure discipline, we have a project delivery management discipline, and so on. In this context, we have an architecture discipline. It's run by Chris Cooper- Bland, our head of architecture, who's in my group. And Chris and her team of leads are constantly working to establish what we think of as good practice in architecture in different situations, because it can vary, it's nuanced.

And also they work across the disciplines, because for example, we have infrastructure architects in the infrastructure discipline, we have data architects in the data discipline. It's a bit of a cross-disciplinary thing, too. In Endava, we have a delivery framework we call TEAM, The Endava Adaptive Model. And in there, the disciplines capture, and if you like, their collective wisdom as to the best way to go about delivering certain kinds of projects from that discipline's perspective. And the architecture discipline's constantly thinking about how they should organize themselves, how do you work with teams, how do you build models, how do you address system qualities, things like performance, scalability, security, and so on, and how to go about evolving the very practice of the discipline.

00:10:27
Bradley Howard: Can you share any specific projects or examples with us? You don't need to necessarily name the client, but any examples to bring this to life?

00:10:35
Eoin Woods: Yeah, sure. So I mean, it's very common that we'll go into a client project and the client's got a reasonably firm view, or I think they've got a reasonably firm view, of where they need to go functionally. They've often really not thought very much about the system qualities. " So does this have to be secure?" " Well, yes," they'll say. " Okay, so does it have to be secure to the point we're prepared to degrade usability?" " Oh, no, no, no. Usability's more important." " Well, that's a really important point you didn't mention before. You're quite sure, you're prepared to have less security for more usability?" " Oh, no, no, I didn't say that." " That's exactly what you said." And you go through this whole process with all of them, because it's a multidimensional thing. It's often quite hard for architects to keep all that in their heads. There's a lot of quality properties, and every architect's got deep knowledge in a few of them and a bit less knowledge than some others, and there's some they haven't worked a lot with. So the architect discipline did a fantastic job. Chris led them to create what's called the, I think it's called the Quality Property Matrix, but it's effectively a big lookup table for all the properties. What are the things you've got to be aware of? How do you capture requirements and how do you test for them? What kind of architectural tactics do you use to meet them? And so on. It's a terrific knowledge store, which is a reusable artifact.

So when we go into the client now, it really helps the architect think, " Right, which of these are important, and what do I have to remember about each one? Because actually regulatory compliance is really important here. I've not worked a lot with that." So there's the collective wisdom of the other architects in the table for them to go and look at. And of course, we're a very cooperative and quite sociable company. The discipline talk all the time. So they often also will contact another architect who knows a lot about that quality property.

00:12:10
Bradley Howard: You've written a lot about the need to move to a continuous model of software architecture. Can you explain what that means?

00:12:16

Eoin Woods: It goes back to what we were talking about at the beginning. You remember you said, " How has architecture changed?" Well, I think this is the most fundamental change. It's that when I started architecture in the late '90s, it was pretty well a case of the architects understanding the requirements, working with the stakeholders, making lots of trade- offs in their heads on bits of paper to work out performance versus security and all those kind of things, and then coming up with a model for how the overall system is going to work. And really good architects were prototyping it. We created something called a walking skeleton, which is, if you think of an architecture model as boxes and lines with all the functions inside it, effectively a walking skeleton is enough code to implement the boxes and the lines, but without all the complicated code in between. So you can see roughly how the system's going to work.
And then iterative delivery would deliver all of the functions incrementally across all those components. That's fine, as long as not too much changes. And it tends to produce, back then it produced systems made up of a relatively small number of quite big pieces. And that's been flipped on its head. It's hard now to predict what the system's going to end up doing, where the system's going have to evolve to.
So we now tend to build systems out of rather smaller parts, using such buzzwords as microservices and serverless technology, and that gives us more flexibility in how we combine them to solve problems and gives us more flexibility in solving the problem. But the reality is, we can no longer at the beginning write down a big diagram and go, " That's it. It's not changing. Just do that."

So continuous architecture is how you rethink architecture practice in order to meet those quality properties, and to make sure that we do things that manage our technical debt, make sure the system is sustainable, we can carry on making change to it. How do you do that when you can't do it all upfront?

And the answer is, we do it in an incremental fashion such that every delivery period of the system, what we call a sprint, two- week increment, in every sprint, we're doing the architecture work that's most important at that time, rather than trying to do it at the start. The tricky thing is working out what the most important and in time means. It's very easy to leave stuff too late, or to do stuff that's too early and you don't yet know enough to do it. But that's the idea, is that rather than being a big architecture, if you like, hump at the start of the project, we flatten that right through the project. And it means that we know a lot more before we come to every architecture decision.

00:14:41
Bradley Howard: And what are the key business benefits from the continuous model?

00:14:46
Eoin Woods: What we're doing, really, is we're abandoning what used to be, honestly, a fallacy that because we did it all up front, we understood everything, we had all the risks managed, and everything was under control. In reality, as soon as we created our architecture model and started development, things started changing horrendous. So actually, we were always having to react to change continuously, it's just that we weren't really set up to do it very well, because we thought the architecture was finished. And we have to abandon that. And so the business benefit is that actually you've got a much better view of what you know and what you don't know, because you haven't made a lot of decisions without any context. You've got a better idea of what you know or you don't. That enables you to manage risk better.

But I think the key thing for businesses, it also makes it clearer what is possible and what is not possible. When we had done the architecture up front, there was a tendency for any big business change to be greeted with the word, " No, that's not possible. We've done the architecture." Agility tried to turn that on its head. And the problem without the architecture, the answer's always yes, but we don't always understand the implications, and actually we can do a lot of damage to the

system trying to change it in a tactical way. An architect tries to get in the middle of that and go, "Maybe. Here are the trade- offs that you are going to incur if you bring these changes into the system at this point." And so I think it gives people visibility, better risk management, and also much better understanding of what is really possible, and the costs and the implications of their decisions.

00:16:16
Bradley Howard: Thank you, Eoin, for giving us an insider view on a fundamental aspect of software engineering. It's quite a feat to make these discussions accessible for the layperson, so thanks for being so clear with us. To all of our listeners, I hope you're enjoying season three of the Tech Reimagined podcast. You can find us on every podcast platform that matters. Please remember to hit those Like and Subscribe buttons, and see you next week for another deep dive into the new frontiers of technologies. Until next time.