## Interviewee 4

**To what extent do you think that the industry is prepared to embrace the transformative technologies of the fourth industrial revolution.**

I think if you would’ve asked me that question three years ago, I would have said we are a long way away from even considering it. Now it's a bit different. I think people's eyes are open. I think AI has become a more common terminology even in public domain. As part of your research, you've probably come across ‘GPT’ and ‘Bard’ and all of these AI chat bots built on large language models and things like this. And every industry has been—I wouldn't say disrupted, that’s not happened yet — but every industry has been shaken up by what's happening.

The question has two parts. One is — do we have the right culture and the mindset to adopt these things? I think that is increasingly, yes. I think people are more interested in doing it. I had conversations with people when I first started my role in my company. Terminology like development, software, technology, AI, data was all buzz words. And from an engineering consultancy’s perspective, technology was something you procured, you didn't develop it. I think that mindset is starting to shift towards — how do we differentiate ourselves from the competitors? and one mechanism to do that is to develop digital technologies and develop new solutions, integrate technology better, have a better mindset towards data, data clarity, data consistency, and data quality. The second part of the question is — are we actually there? Is the data actually there? And the answer to that, I think, is no. It's not to say that we haven't got enough data. The issue is more that our data is never or very rarely, stored in a particularly useful format.  
To give you some context, I did my own AI research a couple of years ago and my thesis was looking at the use of deep neural networks in the context of RCC column rebar tonnage estimation. I trained an AI bot on 200 columns, the idea being — could I estimate with better accuracy than a structural engineer with 25-30 years’ experience how much reinforcement would be required in a concrete column at the end of the project based on the data available to me at Stage 1? I'd got very little information, basically what you get from an architectural schematic design. Interestingly, I was able to find that on just 200 columns, which isn't a lot of data in the grand scheme of things, and it's a bit skewed in scope, because it was focused on tall buildings and concrete frames, so it didn't deal with other structures, but for that particular use case my actual results outperformed engineers by a significant amount. I got an error bracket of ±5 kg/ m3 between what was predicted at RIBA Stage 1 and what was in the column when that building was produced. That's a really good result.

Data-driven analysis and data-driven prediction is extremely powerful. There's no doubt about it. Our industry, structural engineering and civil engineering, is entirely driven by risk, and design codes are based on probability and material strengths are based on probability. We're a domain that's very comfortable with probability. Yet the concept of AI has always been a bit baffling to us. I think it's less to do with whether we are open to the idea of statistics and computing and more about whether we are open to the idea of a computer making a prediction for us based on a result we don't understand. And when you look at even large language models the fear of people is— we don't understand what it's doing under the hood, in which case how do you tell as a user whether something is a hallucination or is a valid result? AI hallucination is when an AI tells you the result, but it is false. It just believes it to be true because it's happened to draw the wrong conclusions from the data provided. If you're in graphic design and the and the AI produces something that is incorrect the worst that can happen, is you put out an image that may be offensive. In our industry, you generate a design and if it’s wrong, hopefully you discover it before you build it and it collapses, but if you don't, people may potentially die. I think the issue here is one that's never going to go away because structural engineers and civil engineers, by training, by their ethics, they agree to design structures and solutions and build the built environment in a safe, secure way. And if you're the person signing on the dotted line, are you going to accept liability for a design you don't understand? No.

**Do you think the data produced by the AEC industry has unique characteristics? What may be the specific challenges that require handling it in a different way?**

A lot of that data historically has been stored in drawings. The question I've asked myself recently is what data is actually relevant? The problem is we generate an enormous amount of data across the life cycle of a project, but for me the only thing that actually mattered is the data that was actually constructed. So, you might have a drawing that's gone through RIBA Stages 1 to 7. But if you take the information at Stage 1, that might be completely different to what was finally developed. So, you could argue, in a strict sense, the first four RIBA Stages of a project are redundant. From an AI perspective, if you want your AI model to be accurate and to draw correct conclusions, you only want to feed it data that was valid and actually built, and the building actually still standing up.

The data that you use to train a model has to be golden data. It has to be correct. It has to be high quality. It has to be clean. That means that you want to be taking the data from the very last stage of the project. You want, basically, the production drawings, the data inherent in those to be captured in your training data. But what that does mean a lot of the times when we look back historically at projects is we have to go through the manual task of looking at the drawings and taking out the relevant information and putting that into a format, usually like a table, that can be parsed by the interpreters and baked into the AI model.

Obviously more recent projects that are created in BIM and in the cloud, their data is often a lot more available. It's a lot better, a lot easier to use because it's already in a useful format. The problem then though is you haven't got that much data because, if you put it into context, how many buildings and bridges get built in a year? Not that many, compared to how many people click the profile picture on Facebook. So we're a different sort of industry. We generate a lot of data, but a lot of that data, is noise because it's not until the very last stage of the project that data would become useful, relevant, and reliable.

**So we do have a problem of data quality and in terms of the data meeting data quality criteria such as accuracy, relevance, completeness, etc**.

I think relevance is a big issue. Let me ask you a question: If someone gave you some data and said “here's the designs for all the columns, and we're at the first stage of the project”, and somebody else gave you the data for all the columns that were designed and the building is now constructed and it's standing. Which data are you going to trust more?

***Obviously, the data from the as-built information.***

Precisely. And it means that the amount of available data actually dropped, if you assume there are five RIBA Stages, and you only use RIBA Stage 5 information or even Stage 6 and 7 information if that's available to you. That data is high quality, assuming you can extract it into a useful manner. The actual reliability of that data is fantastic because that building is standing. You can put a lot of trust in that structural data because the building has not collapsed.

Obviously, that might change. But going forward, say you design a building in Australia and there's an earthquake and that building was supposed to survive an earthquake, but it didn't at that magnitude. One of your obligations if you're doing a data-driven building model is to actually go back and update that data and either you take that data out of it and say that's not a valid data set anymore because it's proven to be wrong or you may annotate it in some way to say actually we predicted that it would survive a 7 Richter scale or whatever magnitude of earthquake and it didn't it. That information can still be useful for an AI because you're prospectively correcting it. You're teaching that we thought this was right, but it was wrong. So, don't do this again.

**So, what do you think are the major benefits that AI can bring to the to the industry? We speak in general of efficiency, but can you speak of specific phases or specific activities that will be benefitted greatly by AI?**

It's hard to say because very few people are actually using AI to generate anything right now. I think you have to remember that there's also a human component to this because there's a general tendency in the world to just automate everything — let's just make sure humans are involved in no part of the process, because then everything is fast, always guaranteed to more or less be correct, and we make a lot more money — only the people that actually have the AI models make the money, everybody else is poor— but that's the mantra.

I'm on the other side of the fence here because despite the fact that I am in a technology role and despite the fact that I've done AI research, I'm not focused personally on the financial capital gains. What matters to me, is, do we do a better job for ourselves, for the planet, for our clients, and ultimately the end users of the buildings and bridges we design? That's kind of the “Tech for Good” movement. We should be trying to move away from the mindset of "can we automate everything?” to the mindset of “should we automate everything?” and then ask the question, if we shouldn't automate everything— because some things we need to do or we would like to do — what do we automate, and where do we use AI?  
If you think about it from that perspective, what jobs do people hate generally? People hate checking things, estimating things where they got very little information, anything that involves uncertainty or repetition that is not enjoyable. That's where people would quite happily push AI. People don't want creative jobs to be automated. I argue nowadays that in a world that's increasingly robotic and automated, if you want to survive this, you just need to be human and focus on what characteristics do we bring as humans to the table — interpersonal communication, conversation, relationship, building trust, understanding how something makes you feel? like how does going into a space or into a building make you feel? There's an aspect of user experience there that AI doesn't care about but humans do.  
The reason I'm playing devil's advocate is because your question is the question everybody asks, but I think it's the wrong question. It's not what AI can do or should do it? Do we really want AI to do everything? Because theoretically AI could probably do everything a few years down the line. We've been through what they call the ‘AI winter’ nothing really happened in AI for many years and a lot of funding disappeared because everyone thought it was just a fad and nothing is going to be achieved. Now we've turned the corner with the inflexion point and we're on an exponential rise. We're now in an AI arms race where every single big tech company wants to have the best AI tool because they think that's going to make the most profit. That's fine, but somebody needs to stand and say, “Is this what we actually want to do?” I don't think anybody's doing that because everybody's so excited about it and no one is actually thinking about the consequences of it.

***But I don't think that we're at that stage where we can automate everything, I mean specifically in the construction industry.***  
  
I think we probably could. I think the problem the only reason we don't is because we haven't got the right skills. Software engineers don't work in construction and if they do, they probably work for a company like Autodesk. That's the big problem. Think about how much stuff in banking is automated, how many financial technology companies have appeared, how many companies in healthcare have appeared, how many companies in pharmacology have appeared. And the reason for that is there's big money in those industries and developers get paid hundreds of thousands of pounds a year to develop something that's competitive. The problem is we are in a slightly different market in the sense that structural engineering and civil engineering specifically, is like a commodity service. The government doesn't care about the best product because it's just a concrete bridge. All they care about is it going to be done on time? Is it going to be built to standard and am I going to have to maintain it for a very long time. and therefore, the concern becomes who can deliver it the cheapest and the quickest. The problem with that is you end up in a race to the bottom. Every engineering firm is developing software basically just be quicker than everybody else. But where's the end goal? We basically developed something so quickly that there's actually no money in it for us anymore and we eradicate ourselves from the market.

One thing I'm heavily involved in now is trying to push the open-source movement in construction and bring back control from big monopolistic organisations that control the speed at which we innovate.

That's how you can potentially change the game, and that's how you start to see increased speed of development. But it's not because it's not possible. To be honest with you, I've thought ever since I joined engineering as a discipline that there is nothing about our industry that can't be automated. In fact, that's the reason I'm moving to technology. Our codes of practise like the Eurocodes are literally designed to be automated. That's how they were written. That's why they're confusing to read because they weren't really designed for humans. It's just simply the fact that we've never had to. I suppose people have just done what they've always done. It's an industry that's very slow. It's never particularly opened his eyes and thought to itself, like, is this the right solution? Are we doing the right thing? Suddenly everybody's asking and the reason is because the generation that's coming into the industry is the generation that's grown up with technology.

So when I joined the company back in 2016 and I was working in Excel spreadsheet to design beams, one Excel spreadsheet per beam and I have to do 200 beams, that was so boring and so error prone. This is not why I studied structural engineering. And that was within the first 3-6 months of joining the industry. And then that's when I decided to start my MSc in computing and transitioned into software engineering because I thought there must be a better way.  
There's only one of me in our company in the structures business. I've got counterparts in building services etc., but it's just not enough people and it's not just about having people that code. This is one thing I have been really trying to get across because software engineering is an enormous field, with so many specialisms. AI is just one of them and even within AI there's so many sub-specialisms. It is one thing, learning how to write grasshopper scripts or learning how to write a plugin for Revit. It's another thing actually building a robust piece of software that everybody could use at an enterprise level. And we haven't got enough people that could write that software or are interested in writing that software because it's not a profitable field for them.

**So, what are the data skills that the industry should focus on building?**

It depends on what companies decide. It's difficult to answer. Obviously, there's an idealistic answer of just upskill people to the point where we've got software engineering development teams working in the organisation. So we move from being structural engineers to primarily being technology companies and we use structural engineers as domain experts to basically provide guidance on what to write. That is probably at odds with what our leadership wants to do. Our leadership wants to grow the business, and the employee base. And ultimately when you start to automate things, you downsize. The problem is, a lot of companies think the more we automate, the bigger our markets. I don't think that's necessarily true. There's contention between wanting to build more to make more money and whether we can actually build more, because net zero carbon is telling us we can't. Even if you theoretically could design more buildings quicker, are there enough clients that want that many buildings? There’s a limit to the number of buildings that get asked for by clients, there's a limit in the amount of carbon we can generate per year, and there's a limit to how much we can produce. There's a triangle, and somewhere in the middle is where we sit. And if you push away in one direction or the other, you throw things out of balance.

Asking what skills we are missing is a very hard question because it really depends on what the industry and the companies themselves prioritise. It depends on who you ask whether they are C-level leadership, an engineer on the ground, or somebody who has just come out of university with a civil engineering degree. Are they going to think about, transitioning to software engineering immediately? Probably not. I did because I was particularly interested.

**You've taken the path of studying software development alongside your structural engineering background. But don't you think that there's a level of data literacy that the common structural engineer or architect or any other practitioner needs to acquire?**

Let me ask you why do you think that?

***To know what they will get when they're using AI applications or tools. Or if they have an idea of an AI application, to be able to communicate with someone with from the IT or software development specialism to create collaborations in which they can create something that's customised for their needs.***

Another question. Do you care how a light switch works?

***No.***

I'm purposely being difficult to make a point. A concept in software engineering that's very relevant is developer experience and it's tied to user experience. And you probably heard the term API (application programming interface). It's a way in which you talk to a piece of code. For example, if there is a function called ‘start’ and you call this ‘start’ function. It's like a light switch. You press it and things happen behind but from the user's perspective, it was one line of code to maybe just run a model or start an analysis. I think that we are going to end up in a space where the individual engineer won't care, where using an AI tool is no different to pressing the light switch. A third party company provides it for you the same way we use ETABS, Tekla Structural Designer, RAM Concept, RAM, Structural Systems, LUSAS, and all the different structural analysis software. We might read the instruction, but most of the time we just blindly trust that it works unless we get a bizarre result, and normally it's our fault because we've modelled something wrong. But we just pressed the analysis button, we get the results and we use those to design a building.

***But is that an ideal situation? I mean, there's the issue of explainability.***

Not at all. I'm an advocate for transparency and that's part of the reason why I'm pushing for open source. One of the GitHub repositories I'm working on is designed to be a robust battle-tested library, for structural engineering that's completely community driven. It's completely open and you can build applications on top of it. And the idea is it's a clean interface that works across various ecosystems. That for me is a transparent way in which people who are interested in it can inspect it.  
I believe that engineering calculations should be visible for everyone and that the things I model should also be ethical. The problem is I am not an economic driver in the business, and I think this is the crux of the problem. I don't make the decisions. If a company comes to my boss and says, “We've got an amazing product, it's going to generate you twice the amount of revenue a year. It's a black box, but don't worry, we know it works”, the engineers don't get a choice in that. We use so many applications right now that are proprietary, hidden from us and we can't do anything about them. We get held back massively by the lack of innovation. ETABS is a great example of an application that barely ever changes. Another thing to add to the list of things that hold us back is the fact that so many of our tools are physically bound to Windows desktop.

Why do you need to couple things so tightly to a Windows machine that I can't communicate with it any other way apart from through your own bespoke interface, or if I export the data into excel or maybe csv format? Trimble's trying to change it a little bit. Autodesk is starting to think about BIM 360 and other cloud services, but so much of our software is literally on the machine. It's a horrible interface to work with, and we can't do anything about it because that's become the industry standard. We just have to move along at the speed at which these vendors sell and update their software. There's been a lot of difficult conversations I've had with people in the last few years. When you're trying to talk about innovating and changing things, you do get a lot of push back.

**Do you think the solution is to have more people with hybrid proficiency who have a strong background in engineering or construction and are also fully proficient in a computer science related specialty like data science, AI, or software development?**

Yes, that’s the world I would like to see but the sceptic in me says that's probably not the world we are going to see. I think we're going to see a world where a company provides a product, and another company just buys it. I work for an enormous company and even in my company there's very few people here. We've got about sixty-five to seventy thousand employees globally now, but as a ratio of that, maybe 1%, 0.5% or even less are working in automation in structural engineering specifically. That's 400-550 engineers or so. We got a few people that do bits of ‘Grasshopper’ and things like that, but we got only one full-time software engineer. Other companies who are perhaps a bit more agile and tech savvy might have a slightly higher percentage, but the reality is that most of the people in the engineering firms are traditional engineers.  
I think there's an age aspect here as well. Younger people generally might have been exposed to programming in their university degrees or maybe just generally had an interest in it. And because it's become available to learn for free on the Internet, people are trying and exploring it. But at the older demographic, those people are not interested in learning to code. To be honest, a lot of the engineers at the older age groups across all companies don't really understand how some of the analysis software works, they just judge the results based on their own intuition. They're not actually checking the modelling configuration to make sure you've got the right settings.

But I think there's a lot of misinformation out there about programming languages, what they do, and why you use which one. Python is slowly becoming a de facto in every industry but in terms of structural engineering, especially in domains where safety matters and correctness matters, there are better programming languages that are much safer to use, that catch a lot more errors. Programming languages have what are called type systems. For example, in some languages you can add a number to a string. JavaScript is a good example, it's a very flexible language. You can add a number to a string of characters and the language will automatically ‘coerce’ it. If you try to add ‘Hello’ and 5, it will convert the number 5 to a string and the result will be the word ‘Hello5’.

And the other extreme, are languages like Haskell, F#, and Rust that are extremely strict. They catch errors at the type level while the compiler which is running over the code. Some programming languages are so strict you can catch most of your errors without writing any tests. So, if you're programme actually runs, it probably works, unless you've intentionally done something wrong. With F# — and it’s the only language in which I’ve seen this— you can capture the units that you’re programming in like Newtons and squared metres.  
Now F# and C# and F# are both built on a platform called .NET, which is an intermediary platform between what runs on the machine in terms of ones and zeros, and the programming language you work with..NET takes your code in F# or C# code, and it converts it into an intermediate language that then gets compiled one layer further and runs on the machine. The idea is to make your programme more portable. So, you write your programme once, and as long as somebody's got .NET installed on their machine, they can run it in that environment, or you can provide the .NET runtime with your code base and they can use that to run the code.

The opposite extreme to that is something like C programming. You compile it for the platform you're running on, but if you want to run that code somewhere else, you have to recompile it. If I compile a C programme to run on my Mac, you can't just pass that on a Windows machine. You would have to take the original source code and compile that back down to something that your Windows machine can run. Python is a bit more portable than that because as long as you've got a Python runtime on your machine and you're not doing something specific in your code that's unique to a machine you can run it.

The point I’m trying to make is that even in terms of AI, I think there's a bigger issue at play here. Because of the hype around certain programming languages like Python. We're in a space where we're running down the wrong avenue. I think we're going to have some big problems in the future when for example, we've got programmes that have designed columns, beams, and walls using code written by an individual in a small company, and maybe nobody's verified it because nobody knew how to, and actually those programmes were wrong. If we can't have that many people to develop and check things, then I do believe we should be using standard tools.

The reason people jump to things like Python is because it's easy to get started with. If you asked a structural engineer to design a building, they would first of all make an assessment of what's the right tool for the job? Should it be in steel? Should it be in concrete? Should be in timber? Should it be a hybrid of all three? But when you ask engineers to write a programme, they don't make the assessment on what tools are right for the job. They just say that they’ll use Python or Excel. The difference between a software engineer and a structural engineer is that the software engineer cares about the quality of what they're building in the technology space. A structural engineer only cares about delivering their work because what they care about is the structural engineering, so they'll use whatever is easiest. But the software engineer will want the software to be of high quality and right for the job. Some thought needs to be given to that and I think it’s relevant to AI as well.

It comes back to data quality and what we originally spoke about, whether we have got the right relevant data. As much as we are quick to build and automate these things, we should also be asking what processes, procedures, and QA/QC checks are in place and being done to make sure that we're producing are digital assetsthat are correct, relevant, and safe.

**So, ideally, we should take it a bit more slowly and be more thoughtful and careful about it. Is that what you're saying?**

Definitely, we need to be more thoughtful. I do quite a lot of training and mentoring in my field. I tell people to think about what they are trying to build and not just jump at the first language that the Internet tells you when you go on Google search. Obviously not everybody should be like me, but we need to bring people with experience, people that actually know enough about what they're talking about. I don't necessarily have the time or the capability to implement every single piece of software in the structural engineering sphere. But if you have, one or two people like me, we can guide developers to build safer products. So, I can't expect everybody to have immediate experience and knowledge in this space, but we can have a few senior people that start to act more like educators or guides.

We're in a bit of a unique position ironically because we haven't actually gone that far as an industry yet. We've still got the opportunity to choose the right tools for the job and I think you could argue people are starting to recognise that a little bit. People start off in Excel, then may be move to Grasshopper, then move from Grasshopper to Python, and then probably to C. What happens is people move slowly up the ladder of increasingly difficult programming languages. They do it because they need to. They realise that there were problems in their solutions because they went for the easiest solution, but that didn't prove to be the most correct or scalable solution. I've gone down that path until I eventually came to F#, which was for me a nice mix of the succinctness and the leanness that Python provides, combined with the speed of execution that C provides, but a type system that was safer than both. It's quite a small niche language. But I actually find that in the particular space I work. It's probably the perfect language.  
And when you look at C, Python, Java, JavaScript, they all come under object-oriented programming languages. But when you talk about F#, Pascal, OCaml, languages like these, are in the branch of what's called functional programming languages. They are very different paradigms of programming. Object-oriented programming languages take the approach that everything is an object. Taking that concept might work for rebar, you might want to create a rebar object, but what about when you talk about data processing? Do you really want to try and map the Eurocode to objects? Functional programming makes more sense in this case because it is more about taking data and passing it through functions.

The reason I'm going off on this tangent is that I think we do have to consider even in the AI sphere whether we do know enough as an industry about how it works to choose the right technique. Because neural networks are not always the right technique, linear regression is not always the right technique, image classification is not always the right technique.

**That is part of the education that needs to happen. Even if someone's not going to go through the full path of specialisation, I think if there is more awareness and people familiarise themselves with foundational concepts, and the conversation expands and happens more often, eventually it will lead to more people specialising in AI or software development.**

The thing is people need to be allowed to pursue their curiosity. We're in a transition. You know, the industry's changing faster than ever and people need to be given the opportunity to explore different ideas. Not everybody's going to want to be a programmer. Not everyone's going to want to be a pure structural engineer. Not everyone's going to work in AI. But if everybody's given at least, the opportunity to explore a little bit what their interests are, that's where the innovation will happen.

If we want to change the industry, organisations need to prime themselves for this period of exploration. It's probably not what they've always traditionally done. Organisations need to consider that maybe we do need to give space and time for people just to try new things and I think that that gets you a long way. And you need a few people like we've said, that do have the experience that can drive this. When I started doing what I'm doing now, there was nobody that could guide me, and it was very stressful actually. I had three and a half years since I took my job on. I was twenty-six when I was suddenly put in charge of the entire of the company’s UK structural digital space. I hadn't even finished my computing degree. I was I doing an MS in computing at the Open University after I graduated from civil engineering. The pressure was phenomenal because people wanted digital innovation but didn't understand how long it takes, and I was saying that we need to do proper quality assurance, proper software design, checks etc. I’m in a big organisation where there's a lot of processes and procedures. I didn't even know where the right IT people are to help me deploy this across the enterprise. Even to this day, there's so much fragmentation and so much misunderstanding. The people asking for the innovation don't understand how it works. So, they don't allocate the right amount of time to do things properly. That's where half of the QA/QC failures are.

Things like reverse mentoring can be one solution, getting senior members of staff reverse mentor with younger staff who are more technologically savvy to explain why these things take so long.

**To what extent do you think that the question of BIM adoption is linked to AI adoption?**

I don't deal with BIM that much. From my understanding, I would I think they are related in the sense that the output of the BIM model forms the data for the AI model. I think you could argue that BIM is the last stage in the construction process, but it's the first stage in training AI at the moment, until you get to a stage where the AI model is the start of the process and it generates the BIM model. So, BIM model is that source of truth where that high quality data is embedded. Once you've got your BIM model at the end of the project that is what was constructed. Theoretically you want to take that data and use that to generate your AI model.