157 Building with code with Jeremy Tammik · BIMrras Podcast

[Speaker 3] (0:00 - 0:05)

I'm so scared, man. I don't know if we can make an episode in English, man.

[Speaker 2] (0:05 - 1:28)

Why not? This is BEAMREST. The podcast about BEAM and technology applied to construction. A podcast that even Chuck Norris doesn't dare to listen to.

Welcome to BEAMREST podcast, the place where we chat about BEAM netology and technology in the construction sector. And you haven't made a mistake, have you? No.

This is still BEAMREST, and we're still trying to get Chuck Norris to listen to some of our episodes. But this is not the reason to torture you with my Galician English. It's not even because we're testing an AI to make BEAMREST in English.

I assure you we can do it. The reason I said hello in English is to put you on guard. This will be our first interview in English.

Well, the first interview in which we'll interview someone. Petru has already interviewed us on some occasion. The first interview we'll do in English.

And this, despite the fact that our guest today is speaking fluently in six languages. I don't know if seven, among which Spanish is, of course. English is the language of code, as it is in many aspects of BEAM.

Or as it is in many other fields. So we've decided to dust off our average English and let's go.

[Speaker 3] (1:29 - 1:30)

Dust off, he says.

[Speaker 2] (1:31 - 3:37)

Dust off, indeed. Dust off. Taking out the dust.

Whatever. I must say that today I'm very, very excited to dive into the world of AEC technology with someone who has been and is behind some of the most popular CAD and BEAM tools since a lot of years. Jeremy Tamek, also known as the building coder, is consulting analyst at Autodesk.

Has been a cornerstone of the company since 1988. And his journey began as technology evangelist spreading the gospel of AutoCAD development across continents. After a stint away working on HVAC application, Jeremy returned to Autodesk in 2005 and ever since.

He's been deeply embedded in the Autodesk developer network specializing in the Revit API. So, yes, Jeremy is the man you blame on when something doesn't work in your plugin. Or when you don't understand why in the hell there's no way to do whatever in an easier way when struggling with the Revit API.

Jeremy has an amazing, very, very amazing academic background in mathematics and physics. He has worked as a teacher, translator, C++ programmer, pioneering the graphic user interface and multitasking projects. Well, as I said before, fluent in six languages.

He's a cultural savant who also enjoys playing the flute, reading, traveling, and engaging in theater improvisation. But he also is a yogi, carpenter, and used to enjoy climbing cliffs or navigating oceans. All those things while maintaining a family life with four children and involvement in community-focused projects.

I can say, I just can say, what have I been doing my entire life, Jeremy? Welcome to WordPress.

[Speaker 1] (3:40 - 3:48)

Hello, thank you very much. That's nice to hear. I'm impressed.

Well, I am impressed. I recognize myself.

[Speaker 2] (3:49 - 4:44)

And I even didn't say it, yoga part, your meditation skills, your commitment with music, and a lot of things that are in your resume or around the internet, which makes me ask if you are the same guy or there are a lot of Jeremy Tameks all around the world doing things related with the construction sector, because it's amazing what you have done and what you are doing. It's all a question of marketing. Then congratulate your marketing team, because it's quite amazing.

And please let me introduce my partners. Today, just my partner, before continuing, because I must say hello to Rogelio, who is helping me to destroy Shakespeare language.

[Speaker 3] (4:45 - 4:48)

You don't need any help from me. You don't need any help from my partner.

[Speaker 2] (4:50 - 4:51)

Well, as usual.

[Speaker 3] (4:53 - 4:59)

You don't know what you've been doing with your life, but we do. We do.

[Speaker 2] (5:02 - 5:04)

By the way, Rogelio, how are you doing?

[Speaker 3] (5:06 - 5:06)

Fine, fine.

[Speaker 2] (5:06 - 5:08)

Are you very nervous?

[Speaker 3] (5:09 - 5:12)

Yes, I'm very scared to death.

[Speaker 2] (5:14 - 6:21)

Well, going to the point directly, because, you know, being not my mother tongue, perhaps in English is more difficult, at least for me, making introductions or quite ample speeches in advance. So let's go straight and starting with your background. You've been with Autodesk since 1988.

Most of our listeners, well, you know, they were just a dream in their parents' head or even not a dream. But this means that you've been there since 2010. I've been checking, and that means a lot for many of us, because it's time to ask something I've been asking myself for years.

Come on, guy, what's wrong with our hatchet boundaries? For real.

[Speaker 3] (6:21 - 6:22)

We all want to know.

[Speaker 2] (6:24 - 6:48)

I think I've asked myself this question, I don't know, millions of times. Many people are aware of your maths and physics background, but that is just a tiny piece of your story. I'm not talking about your philosophy or martial arts interests, but it makes me wonder what was the way from your early studies in humanities to be a computer guy?

[Speaker 1] (6:49 - 6:54)

I actually never studied humanities. I just studied math and physics.

[Speaker 2] (6:55 - 7:03)

You started humanities in high school? Ah, yes, high school.

[Speaker 1] (7:03 - 9:56)

I went to an old fashioned German high school called Gymnasium in Stockholm in Sweden. So I grew up in Sweden. I read it.

The nice thing about this school was that it was sort of traditional focused on humanities. So there was more stress on languages and philosophy and culture. And I was very happy about that.

And another thing that I was happy about was that it's a German foreign school. So Germany subsidizes these schools all over across the planet in various places. And for the teachers, they come from Germany for a limited time, like four years or something.

And it's quite a privilege to go there. They get economic advantages. They are selected.

So we had very good teachers. And another aspect was that for people staying in Sweden, it was quite attractive to leave the school for the finishing years and go to a Swedish school instead to finish off because they could save one year's schooling. This gymnasium system had 13 years of schooling in total until you reach the baccalaureate, the Abitur in German.

And in the Swedish system, you had 11 or 12 years, depending on what kind of or 13 for the really hardcore technical branches. But then you would be specializing in technical stuff or in simpler stuff. Very nice system in Sweden.

I like it. All the kids are together until the end of the ninth year, and then they can choose to specialize in any kind of direction. And even if you want to become a carpenter or a mechanic or something, you still stay in the same system.

So you have a certain amount of schooling and a certain amount of practice. And the people aiming for more academic directions, they have more years and no practice, more academic stuff. Anyway, so lots of people left my school and our class shrunk and shrunk and shrunk.

And we were actually at the time the largest finishing class in the history of the school with 10 pupils. The class before us had four pupils for one year and the class after us had six. So it was quite special and very high quality, a very pleasant going to school.

[Speaker 3] (9:56 - 10:22)

Do you think because here in Spain we have some kind of a big debate about humanities in high school and our programming about how the education system must be. And do you think that background in humanities added up to your formation, to your studies in maths and physics? Do you find it useful?

[Speaker 1] (10:23 - 10:54)

Yes, yes, very, very useful. Absolutely. I do think it's a shame that schools tend to scare young people away from mathematics and physics and rational thinking and logical thinking.

That's really a big fault in the part of most of our didactical systems, I think. But at the same time, humanities take a much more time to shape.

[Speaker 3] (10:54 - 11:05)

I agree. This is something that you really find the taste for it when you're mature and you grow up.

[Speaker 1] (11:08 - 12:53)

Yes, the foundation is laid in the schools, of course. I'm a great fan of early schooling or early caring for children. There's a study from 1962 proving that it is economically sensible to invest in schooling or caring for young children.

They took 192 people in a ghetto in New York, children, and half of them got caring in the afternoon and the other half didn't. In 1962. And they followed these children since then.

How long is that? That's 38 plus 24, over 50 years. So they could see what effect did $1 of caring per afternoon for one child have in its life, in employment, health, criminality, everything.

And it pays off hundreds of times over. And the prizes for this study were not given to the social workers or teachers who explored this concept, but to a professor of economics. He got a Nobel Prize for economics for proving the return of investment on this initial childhood investment.

Proving something? And still all politicians ignore it up until today. They say, no, we need more money for police.

[Speaker 2] (12:54 - 12:55)

We need more money for prisons. No, you don't.

[Speaker 1] (12:56 - 12:59)

No, less money for schools, less money for kindergarten.

[Speaker 3] (13:01 - 13:07)

That's crazy. That's criminal. The endless confusion between spending and investment.

Yes, yes, yes.

[Speaker 1] (13:09 - 13:23)

Yes, you can. And there again, there you come back to rationality. We would need a rational look at investment and spending, and we would need rational politicians, and we would need rational voters.

[Speaker 3] (13:24 - 13:26)

And the problem is that...

[Speaker 2] (13:26 - 14:33)

That's a very easy goal to achieve. But perhaps it starts with the previous part, the investment on schooling. Yes.

So, it's like going again and again on the same mistakes. I don't know if nerd or geek is offensive in English. I really don't know.

I mean, I use it in Spanish as, well, it depends on the context, as something good, but I don't know if in English is something bad. But I'm pretty sure that there's something that happens between being a humanities student in baccalaureate, to start maths and physics studies in university. How is the way to do that?

[Speaker 1] (14:33 - 15:44)

I would definitely identify very strongly with geek and nerd. And I always was one, I'm afraid. And I think that most people who decide to study mathematics are nerds and geeks in a way, because, at least from my point of view, mathematics is a way to deal with an area with some certainty.

Is it the way to understand our world? No, definitely not. The world is much more complicated.

Mathematics is no way as complicated. Then the way to avoid it. And even more complicated than just the world are human beings and emotions and cultures.

I can say, yes, they are. Yes. So I think many, many people who go to study mathematics are sort of seeking something simpler than the normal, the human world and even simpler than the humanities.

[Speaker 2] (15:47 - 16:46)

Well, as I said before, I researched your resume and your entire, well, all which is published in Internet about you, or at least many of the web pages that says something about you. And many of them has been written by you, by the way. And everybody can read that you're fluent in six languages, which we really appreciate.

And not only speak six languages, but you're an avid reader too. And that's only the peak. And what is in my head is how do these skills, I mean, being a carpenter, being a musician, being a climber even, how all those skills influence your technical work?

[Speaker 1] (16:48 - 17:14)

Where are they when you are working? My work is mostly pretty separate from my private life and other activities. So there's the influence is just I'm interested in everything.

I'm interested in solving problems. And that's one common factor between all of these activities.

[Speaker 2] (17:14 - 18:22)

I bet when you're climbing, perhaps solving a problem could be the difference between a good, a good end or what? Not so good. Well, perhaps most of us and most of our audience has a story to tell about how they ended in BAM and BIM or how they discovered a more productive way to work using BIM tools.

I must say using your BIM tools, because, you know, we all assume that Revit is on everywhere and is being the king of the market. But your background is anything but the usual. I mean, you're a mathematician working in the IEC, an IEC related company.

And how was it even linked? I mean, perhaps there was no construction workers to solve problems or to programming tools. I bet there weren't when you started.

[Speaker 1] (18:23 - 19:43)

Yeah, well, Autodesk has always had a focus on these different domains and tried to address all of them. So the mechanical, the architectural, the generic drawing and modeling, and in the last decades, also the creating worlds in general. And when I started working for Autodesk, I was actually employed to do developer support in Europe when we didn't yet have any developers, because John Walker had only just integrated AutoLisp into AutoCAD.

That was the first real sort of hardcore customization possibility within AutoCAD. Before that, there had been some very minute possibilities, but Lisp really revolutionized the way that AutoCAD worked and was used. And even the way that Autodesk looked at AutoCAD and the developer community as an area where unimaginable things could be done and the tools could be used in ways that nobody had thought possible before.

[Speaker 2] (19:45 - 20:55)

And that was thanks to John Walker. I'm one of the first on claiming that one of the smartest things on Revit is the API. I've said many times that it's very, very clever to launch an incomplete product that can be finished or adapted by the owners.

Well, owners, you know, the users. And I think it's fair to say that you came from AutoCAD where a lot of people have, as you said, learned functional programming, even if they didn't know Lisp was functional programming, by the way. But how was born the idea of opening in so open way the app to be completed by the users?

I mean, when somebody realizes that perhaps the best way to make our tool Revit more and more popular is not to make tools for everybody, but opening it.

[Speaker 3] (20:56 - 21:02)

Are you saying that the best thing to do, it was not doing anything at all?

[Speaker 2] (21:02 - 21:12)

They did the best thing, which is open your doors and let people to interact with your tool. Yes.

[Speaker 1] (21:12 - 26:08)

Basically, that's what Autodesk proclaims today, being a platform company, this platform idea. And the idea of a platform grew step by step. And that was also all due to John Walker.

And back in the last millennium, because he implemented Autolisp for his own purposes. He wanted to do things in AutoCAD that were difficult to do manually. So he wanted to do it programmatically.

And for that, he needed a programming language. And then when Autolisp was available and successful, the software landscape started to change and more and more packages, components became available with powerful functionality inside of libraries. And also Autolisp showed that it was powerful for interaction and manipulation, but for performance critical tasks, it was not sufficient.

So the next package or functionality that John Walker became interested in was a software package called AutoSolid. That's a CSG, Constructive Solid Geometry Modeler. So you could do solid modeling in this program.

And it had a C interface. So after Lisp, John wanted a C interface in AutoCAD, because with a C interface, he could hook up the source code for AutoSolid with the source code for AutoCAD. So he went and implemented ADS, the AutoCAD development system, which is just a C wrapper around Autolisp.

And then he packaged AutoSolid into AutoCAD and suddenly we had 3D AutoCAD with real solid modeling capabilities. And quite soon afterwards, object-oriented programming became a big topic for large software projects. It was clear that the complexity of a large software project cannot be managed at all without compartmentalizing things, separating them clearly.

This is one piece, this is another piece, this is a third piece, and the interfaces are clearly defined. So people started thinking about converting AutoCAD to an object-oriented environment. And that happened with another pretty genius programmer, Bill Atkinson, who got the task of making AutoCAD object-oriented.

He initially wanted to do it in Smalltalk. Smalltalk. Smalltalk, yes.

Very pure, pure, pure object orientation. It seemed a bit too complicated or challenging. And he ended up choosing C++ instead and implementing the AutoCAD runtime extension, ARX, which later became Object-ARX.

And then with ARX, AutoCAD was just a kernel, a very small, small, small kernel, and everything else was plugged into this kernel. And that meant you could take other things that were not originally planned and add them as well. So AutoCAD suddenly became this huge, unlimited programming platform.

And in Revit, the situation was completely different because there was recently a blog post on the history of Revit. I don't know all the details, but it was originally written completely without a programming interface and also without any programming interface in mind. And when Autodesk acquired Revit in, I don't know what, 2005 or something, Autodesk said, we are a platform company.

We understand the importance of end users and developers being able to adapt the product and adding new functionality. And the Revit team said, never, never, no, absolutely not. That is not the idea of Revit.

[Speaker 2] (26:08 - 26:16)

And there was a fight about that, a struggle. There's no way to success with that orientation.

[Speaker 1] (26:17 - 26:27)

Anyway, so Autodesk, as the owner of the product, got to do as it wanted and an API was added.

[Speaker 2] (26:28 - 26:55)

And you started to release the API as it's release of Revit more and more open. Is that because it was not so API focused? I mean, because what you said, it was not designed to be so open or not to be a platform itself, which is nowadays.

[Speaker 1] (26:55 - 28:52)

Yes, well, it's still sort of halfway between platform and user interface. Originally, the Revit API was a com API. So based on the Windows com interface, and that was a bit shaky.

And quite early on, 2009, 2010, it switched to the .NET API, which was more stable. But as you say, Revit itself, it's kernel is not designed to be an API driven kernel or a platform for programming. So it is still internally more focused on the user interface.

But the developers have set it up so that every single piece of user interface functionality comes equipped with a script. An internal Revit development language script. So every button, every command, every option is somehow linked to this scripting language.

And if you click a certain button, then this scripting language will produce the .NET API for it. So nowadays, the requirement from the program managers of Revit is if you write a piece of functionality, you write the user interface and you must enable this script so that the API is automatically also produced. And that's why the coverage is broader and broader.

And for quite a number of years, there were old pieces of Revit that were not equipped with this API scripting. And all the new pieces, all of the new pieces were always equipped with scripting. And therefore, the coverage would grow and grow and grow.

[Speaker 2] (28:53 - 29:27)

And then the migration, the recent migration. I mean, it's well known that the Revit API was based on .NET 4.8 framework. But you've done, I think, a huge jump in Revit 2025, changing from .NET 4.8 to .NET Core 8. This is a significant upgrade, isn't it?

[Speaker 1] (29:27 - 29:59)

Yes, it is. Yes. It was actually overdue.

Many developers have been asking for an upgrade to a more modern .NET platform. Several years now. Yes, absolutely.

At least four or five years, people have said, I'm using new .NET functionality. Why are you not? And the reason is simply that Revit is making use of a number of additional components again.

And the dependence.