00:07
I was born to be a mainframe nerd. And it's funny. I was actually born the same year that Tim Hortons was founded, which happens to also be the same year that the IBM mainframe was announced, and that's 1964. And IBM announced their system 360 mainframe on April 7, 1964. That was after about two decades of the whole world trying to figure out what a computer is supposed to be. And then a decade before that, of people such as Alan Turing really putting together the foundational theory that, you know, built on, you know, hundreds and thousands of years of human thinking about automating stuff that people did as machines. And so Turing had this idea of there being this computational mechanism, and even invented the idea of the artificial intelligence test, you know, the Turing test for artificial intelligence.

00:57
Can you tell whether you're dealing with a human or mainframe or.

01:00
Sorry.

01:01
Or human or computer? And so all of this thinking led up to the 1930s. And then, of course, this little thing called World War two happened. And on the one hand, it got in the way of the advance of technology. On the other hand, it also led the groundwork for some of the most important advance going forward, including, you think about, what is it called, the simulation game, the Turing movie. And so at the end of World War two, basically, IBM had really built a lot of punch card machines for the war effort, and suddenly they were no longer needed. And so what you have is all this suddenly available technology, and this will to create computing, but really not a first real computer, just a whole lot of mechanisms that they're computing, like things such as punch card machines.

01:48
And one of the interesting things that came about that I hope you don't mind me going way before I was born, just can't put this in context. But one of the interesting things that came out of this was all these people inspired to do stuff with computing before the computing was even there to do it. One example was a fellow, a jesuit priest named Doctor Roberto Busa from Italy, who wanted to do a complete index works of the works of Thomas Aquinas, which was something that would take more than a lifetime to do manually. And so he approached Thomas Watson Sr. And said, can we have some of your punch card machines you have leftover from World War Two to do this index? And Watson senior said, yeah, I don't see a problem with that. And Busa said, but one little thing.

02:29
We know the punch cards are capable of having characters, not just numbers, but your machines only had the numbers. Could you reconfigure them to handle letters as well? And Watson said, well, you know, my people have advised me that this would be very hard. Well, Busa had a little card up his sleeve, literally a card punch card shaped in Bernice that had been at a waiting room at the IBM headquarters in downtown Manhattan. That said, the difficult we do right away, the impossible takes a little longer. And so Watson laughs and says, okay, just don't make this the international boost of machines company. And so he gave the orders and they reconfigured a bunch of these machines to start processing letters as well as numbers.

03:07
And so based on this, then Busa made it his life's work to create what is now available on the web as Indextemisticus.org dot. And so he was one of the contributors of really humanizing computing. So there's more than just calculation. Now, a next person who did that, and this is leading up to me, believe it or not, that did this, was Rear Admiral Grace Hopper, who in the early fifties was one of the people who came up with the idea, which echoed Lady Ada Lovelace's idea back in the 18 hundreds, that a calculating thing could be programmed to do something more than just calculations. And the case of Lovelace, it was doing Bernoulli numbers. In the case of Hopper, it was doing it using human language. And she was on the adventures of the idea of a compiler.

03:50
And so over the 1950s, as organizations such as Share, which was founded in 1955, and other users and organizations gave input to IBM about what a computer needed to be for business purposes, people like Hopper put together how we could be more human. So all this leads up to the ultimate 360 degree machine that IBM put together, which in my own things, ended up being one of the ways that humanity got to the moon was the

system 360 mainframe. And so along with that, I happened to be conceived just before IBM announced the system 360 mainframe and born the same year. And so I grew up in the era of people who had been born after computers had really been established as a thing, but before everybody kind of grew up as a computer person, you know, that they're digital natives, if you will.

04:39
And so I and all of my contemporaries kind of had to, you know, grow into computing. But for whatever reason, it was just first nature for me. I'm just exactly the kind of nerd who fits not only with computing, but especially with mainframe computing. So as I grew up, my first computer that I actually got to deal with was in my last year of high school, grade twelve, when they brought in Apple II computers. Apple II plus computers, in fact, and I just found my way right there. I thought I was going to get a degree in math at all to university, but discovered computing. And so I went straight into computing in my university degree. But you know, it's funny, the university I went to, the University of Calgary, had mainframes, but they weren't teaching on them.

05:16
They were teaching Unix and distributing stuff and all these things. C and so I got to learn all those things, but I didn't get to learn the mainframe stuff, which everybody was already saying, oh, that's old, you know, and funny because my first year of university was the year that Time magazine named the personal computer, that back then they called it man of the year. And sort of this whole idea we're going to move off of these big centralized mainframes. It'd be almost 20 years since the do not fold spindle and mutilate movement had really taken, especially the west by storm. And so there's this whole moving towards personal computing.

05:51
And I got initially swept up with that and was just really happy of being able to have my own piece of computer that I could do whatever I want with you program and have fun with it. And this was sort of my journey as I took my computer science degree. And then I got to my first real computing jobs during my summer jobs. And I got a job ready cobalt for heli hiking and heli skiing or outfit based in Banff, Alberta, canadian mountain holidays. And they were taking a COBOL course, so I had to learn it on the job, which is apparently how people learn a lot of legacy and main trade and such things on the job.

06:27
And so I learned COBOL on the job and then came back in my fourth year of university and took a COBOL course, which of course I did quite well since I'd already been working with COBOL. And it really struck me how COBOL thought differently than other programming languages because it really thought in terms of business. It was about doing practical business things. And we may forget that. CObOL, in fact came out five years before the mainframe. And I've written articles about how basically the mainframe architecture on COBOL are like twins separated at birth. They think the same way they think in terms of business data, decimal math, massive amounts of character data, you know, and so they process them with just incredible eloquence.

07:07
And I say eloquence because one of the beauties of a COBOL program is if you read a COBOL program out loud, it's nearly self documented. It tells you what it does because it's so English like now obviously that's the challenge of English, isn't allowing to speak, but otherwise. So CoBOL, which was born five years before me and was really led by Doctor Grace Hopper and then the IBM mainframe, which was led by people such as Doctor Fred Brooks and Eugene Amdahl. And they're like, these all were kind of established by the time that I was getting my computer science degree. And so much so that by the time I graduated in 1986, the mainframe was 22 years old. And I still didn't know that much about it other than I'd written some cobol.

07:53
But I didn't know that I didn't know about it until in a funny turn of fate I put on my business cards I was using to try to get a job experience from micros to mainframes. Because of course back then PCs were micros, microcomputers. And I thought that the big Unix computers I worked on at the University of Calgary were mainframes. And so on the strength of that I got a job interview and offer to become a CICS or kicks a trainee system programmer at the government of Alberta. And it was only when I got there was like, wait a second, you don't know anything about me. I said I didn't know. I didn't know.

08:24
It reminds me of Rumsfeld, you know, there's like the people who know and know that they don't know, and then there's the people who know what they don't know that don't know. And then there's ones who don't know that they don't know, you know? And I was don't know that I don't know. I didn't know that. I didn't know about

mainframe. But fortunately, on the strength of my ignorance, I got to learn mainframe. And it was just no turning back. I was just so utterly taken by this amazing platform that so different from every other platform was not consumer electronics, commodity computing, it was serious industrial strength computing that had to do with the heavy lifting of the world economy. And I was just so impressed that I just dug and dug into it.

09:04
And so after a few years in Evans, working with the Alberta government, came back to Calgary and there's a whole family story parallel to this and worked for the city of Calgary. And they had a mainframe back then, did all kinds of system programming and network programming and security, then went to work for a very large mainframe software vendor who at the time was known as computer associates. Then became known as CA, and in the last few years of being acquired by Broadcom. But I worked for them for 14 years and did every different kind of both security and computing job, including being in a position to have the global responsibility for their mainframe services strategy.

09:42
So based on that, in 2004, I wrote a white paper about the need to get a new generation on the mainframe because of the fact that the average mainframe was getting one year older every year, not people. The whole average itself was going up by one year every year, because they had this weird idea they were moving off the mainframe, and they still haven't figured out after y two k and everything else that the mainframe wasn't going away because it was the backbone of the world economy. And so I wrote this white paper, which is still out there describing the need to get to a new generation to get new people in place. And I'd already been going to the ChER conference for about five years at that point.

10:20
And at the Cher conference, a couple of young ladies, Christine, Ben Harper and Iris Rivera from IBM, co founded Z NextGen, which was a project at Cher for new mean framers. And so I got involved with that. And so basically, for the next decade, I was involved at CA in stuff that was related to the future of the mainframe workforce and all these things. And then I was given the opportunity that eventually everybody at CA was given to find my own way forward. And so I co founded, with my late wife, an organization called Mainframe analytics. It was all about understanding and communicating the role and value of mainframe.

10:58
And so I've been doing all kinds of writing and presenting and teaching and co writing books, and even had the opportunity to come and IBM champion, in fact, first person in Canada to get the IBM champion for Z mainframe designation. And so that's just, believe it or not, brief summary of how I got to this place. Any data you'd like me to fill in that I might have left gaps on? No.

11:22
That's a great background, right. And that's a great history. And I think I'm in the Rumsfeld area of I know what I don't know, and I don't know what you just talked about from a history standpoint around the mainframe. And there's pieces of it, right, that I picked up along the way. But I think one of the things that's always interesting to me is learning something new about the mainframe, because there's so much to it and it's so important in the world, right? And what it does from running our federal government, running airlines and hotels and credit card transactions and banks and everything else that, you know, most people don't understand, right? And I often, when I tell people, right, I actually work for, as my day job outside of the podcast, a mainframe software startup. They're like, is it not an oxymoron? Right?

12:11
And I'm like, no. You got to understand, there's actually a lot to the mainframe and a lot of benefits to the mainframe. So I guess my next question for you, and you've given me a lot of interesting facts. So if you want to move on from this question, that's fine. But is there an interesting fact about the mainframe? Right. I often like the amount of transactions processed, or often people say, look, if AWS goes down, it'll have a big impact, most notably, like, a lot of the social media sites or Netflix or something like that. But if somehow the mainframes were to go down, the world's economy would be on hold until it came back up. Now, obviously, the mainframes are not sitting in a common data center like a lot of the cloud providers.

12:52
But is there something to your, you know, about the mainframe that, you know, is always enticed you, or a fact about it that you think our listeners would find interesting?

13:01
You know, Doctor Fred Brooks, who was the fellow who led the development of both the mainframe hardware and software, in fact, he wrote the mythical man month as a reflection on creating the mainframe operating system, and then decades later, wrote the designer design as a perfect reflection about that. You know, he referred to the mainframe as a platonic ideal, that they had taken all the thinking about computing technology and all the thinking about what a business and anybody else needed for a computer and put it exactly into something whose theory and practice matched, which never happens. You know, I love the joke. You know, in theory, practice and theory are the same thing. In practice, they're not. You know, the whole mythical man month was about theory of programming and how that didn't turn out to be quite so smooth.

13:44
But just the fact that the mainframe was designed to be the most perfect manifestation of what computing could be. And every other platform has been built as a commodity from the ground up in terms of meeting a specific need at a specific price point for particular type of consumer. But the mainstream was literally built to meet a theoretical requirement that made it stable and founded and designed to move into the future. And that it's done that so well that no other platform has been able to compete with it. And so every other platform in competing with the mainframe starts by refusing to discuss any of the mainframe differentiators.

14:25
And so much so they've gotten such market share that people don't realize that all the differentiators that make the mainframe the only suitable platform being the system of record for the world economy, are invisible because nobody wants to talk about them. But they are absolutely essential to the running of the world economy. And no other platform but the mainframe has.

14:45
Yeah, yeah. I remember when I was first working with some of my peers at Accenture, right, talking about mainframe modernization, there was actually some pretty senior folks with a good amount of experience who didn't understand that the mainframe is refreshed every two and a half years. Right.

15:02
Well, it's so leading edge. It's way ahead of every other platform in addition to being way below and the depth of its foundations.

15:09
Yeah. And it still runs. Right. I mean, what's difficult sometimes about the mainframe are a lot of these applications are 40, 50 years old. Right. But that's also pretty impressive that our forefathers, if you will, were able to write a program that still runs an airline, still runs a credit card, still runs a hotel. Right. And the impact that has today and why, yes, it's 40 or 50 years old. It still runs fairly well.

15:38
And I'd like to say something about that. Bill Gates, who is, of course, the flag waiver for distributed computing, once said, automating an inefficient process will not make it efficient. But he was talking about the other side of the coin from the mainframe. The mainframe automated a process that was tried and proven through thousands of years of humanity. The mainframe manifested how we do business property. It manifested professionalism as established before computing ever was invented. It did it the way it was done and then made it more efficient, more reliable, more trustworthy, but it did it properly, professionally, the way the accountants, engineers, doctors, lawyers demand. This stuff stands up in court. This stuff does accounting to the penny.

16:21
Reliableness of the same penny you have today was on the same mainframe, you know, 30, 40, 50 years ago, that, you know, you can literally trace that penny if you needed to because of the integrity of the mainframe, you know, and so that you've got this system that was literally designed by humanity before computing even existed and is embodied in how the mainframe does its stuff. And that's really interesting, I think.

16:45
Yeah. I mean, one of the challenges that's very difficult to replicate are the, you know, historical compliance and regulatory things that everything that the mainframe tracks. Right. Today. Right. And trying to modernize those applications and replicate that, I mean, there's not a whole lot of institutional knowledge on those things

anymore. Right. Because we haven't, to your earlier point, done a really good job of refreshing the talent on the mainframe.

17:08
Right?

17:09
Yeah.

17:09
And part of the issue is that refreshing is not always what's needed. You know, we have been sold this bill of goods in the world of computing that, you know, only new is good, that innovation is better than established, proven, and, you know, and the legacy is a bad thing when legacy basically is a gift of value from former generations. The mainframe was designed to house legacy. And although none of the programs that ran on the first mainframes had previously existed on the mainframe and such, they had to be rewritten to some extent. I think it made a promise in 1964 when they came up with six different models of mainframes that any program you wrote for any one of those models would run out, all of them, and into the future, and they kept that promise.

17:48
That means that people didn't have to rewrite their code, they could build on it. And that's what's been happening is the people being able to build and don't have to keep rewriting it, you know, putting a new version, installing a new version. If it works. And by the way, not only does it work for the past 2060, whatever years, it's worked for the past 2000 years, because this is how we've always done these things, because it works. It's a legacy. It's tried and proven, and we just happen to be able to do it properly on the main trick as well.

18:13
Now, yeah, I know, the backwards compatibility, right? As you continue to upgrade, you go into new versions of operating systems. I think 3.1 or 3.2 is coming out this year or has already been released early released, which is a huge step forward, a huge migration for companies. And those same applications we're talking about that are 40, 50 years old are going to run and they're going to run even faster and better. If you recompile and do all the things you need to do, even if.

18:41
You don't recompile, hardware is going to better. But you're absolutely right. One of the cool things is they continue to optimize the hardware to the application requirements and you've got this amazing thing happening where now you've got, for example, vector decimal math, where you take your cobalt programs that you wrote 30, 40 years ago even, and just recompile them. And these new instructions make them run orders of magnitude faster. It's amazing.

19:06
Yeah, exactly. So let's focus a little bit on the talent because that's a big thing, you know, when, you know, we speak, when I was speaking to clients about, you know, the need to do something with their mainframe, right? Is it to get off? Is it to, you know, focus on modernization, modernizing in place, right? What is the right approach for them? You know, it really came down to three things, right? Business agility, trying to get new things out faster, respond to things in the market, because businesses and the business processes are changing. Talent was a big one and in cost, right? And there's different ways for solve is solving the cost challenge and everything else. And those business cases can be complicated to get off, but there's also ways to save money by staying on the mainframe.

19:49
But talent was always really the hard one, right? I mean, there's a lot of solutions out there where you take COBOL and convert it to Java, but you just took 500 COBOL Java developers and now you gotta convert them to Java developers, right? But at the same time, it's really hard to find, you know, 500 cobalt developers and continue to refresh as folks retire and everything else. So what are your thoughts on talent? I gotta imagine you've got some, you know, some good thoughts on how to make the mainframe, but I'm cool again. But what do we do with the talent? Right, because you are working on the leading edge then some very important programs. Right? And software.

20:28

Well, as you might have a lot of thoughts on that. And the first thing I have to do is own the fact that for much of my career, I've had the mistaken idea that you needed a degree to be a computer person. And the thing that has really disabused us on that is the fact that there haven't been enough people with degrees, but also that academia has built into most computer science series a very explicit bias against the mainframe. And I have a lot of thoughts about why that's the case. But regardless, the fact is that it turns out that in order to be a good mainframe, you don't necessarily have to have a degree, you just have to be smart and hard working and have a really good work ethic. Well, that is even more true about Cobalt.

21:08
Now there's literally hundreds of billions of lines of CObol running the world economy. And so, I mean, you can take your Java conversion and go to your most leading edge, CObol, that's currently being maintained and developed and changed and change that to Java if you want. If you want to hire Java programmers. But, you know, Cobol was built before there was such thing as a computer program. We forget that CObOl was designed in 1960. You know, they didn't even have, I mean, until 1964, we didn't even have a definitive, you know, ultimate computer design yet. They were still on that journey.

21:37
And so Cobol was designed for clerks and, you know, people, you know, people whose job title was computer, you know, people who were good enough at thinking that they could do bureaucratic stuff and, you know, to type, and they would type COBOL. And so it's literally designed, you know, in its english style, verbose way of doing things to be done by a bureaucrat, not by somebody with a high end computer science technical degree. In fact, as somebody who has a computer science degree, I have to say that the one thing about COBOL is it's just a little bit boring. You know, you could do interesting stuff on COBOL, but you shouldn't. That's not what it was there for. You know, CObOl is not for being a genius. CobOL is for getting the job done day in, day out, doing good, quality work.

22:19
And guess what? You can get a hard working, smart person off the street who's proven their character and their intelligence, and they can make a very good Cobalt programmer with very little training, because that's how the language is designed. We don't need to go to universities to get thousands of Java programmers and then convert all the cobalt into Java. We need to get smart, hardworking people who want a real career paid well, that they can do the same thing day in, day out that they feel good about doing and get them to learn CObol on the job. So it's important to recognize that doing a conversion of 800 billion lines of cobalt in Java isn't in anybody's interests. It works. It's legacy in the best sense, and there's really no need to go out and change it.

23:02
But if we want to bring in some java, we can have the Java interface to it or convert the cobalt to Java. But that's not about the workforce. Now, I'm going to show you the secret, and this is after 20 years of thinking and working really hard about this. And the secret about getting new people in the mainframe workforce is so obvious, it's painful pay to. That's it. That's all, literally. Just give them a good career level salary and train them to get into the job, and you will have all the people you need. We have so deprecated the mainframe for so long and required so many savings from it when it's done all. But, you know, the cost of the mainframe is a fraction of the cost of a not even equivalent workload on any other computing platform. And that includes the people.

23:45
There's fewer people. And they're not paid like crackerjack programmers, super spectacular programmers. They're paid union level wages. The good news about union wages is you can make a living off that, but they are also not superstar wages. And so if you want to pay people better, then you'll get more people. And it's really important to recognize that it's just about recognizing and paying for the value, and that's all it is.

24:11
Yeah, I will counter that, though. And I did not release a survey. This is not official survey I did. But I have asked a few recent grads that want to go work for a snowflake or a databricks or AWS or Google or Microsoft. I said, look, what's your starting salary going to be? And they're like, oh, it's going to be around $120,000 a year. And I said, well, what if I paid you double that or even triple that? Would you go learn cobol and become a mainframe developer? And they said no.

24:44
Yeah, and you know why? Because academia has gone full press for decades, basically, ever since IBM and Harvard had a falling out over the Harvard mark for computer. When Harvard pre announced the computer that

IBM invested all the time money in, IBM and academia just had this falling out ever since then. And, you know, when I took my computer science degree, I was fed the gospel of mainframe. Bad. Mainframe. Bad. I'd be in bad. I'd be in bad. We know that's not true, but especially those of us in the mainframe ecosystem. But you literally get attitude from your non mainframe technical colleagues, like, what is wrong with you if you even say mainframe? Because they've been fed this dark anti gospel of anti main trade that pervades the world of consumer electronics.

25:27
Commodity computing, that is just one of the most inexplicable anti human things that's happened in the history of technology. Because meanwhile, all along, the mainframe is the only computer that's actually reliably being processing the world economy while everybody's being excoriating. It's really problematic. And it's reminiscent of how the original people who staffed the mainframe, you know, did not happen to be, you know, these young university graduates, they happen to be hardworking career people from every different walk of life. You know, think about wonderful movie hidden figures. Those were the original mainframes. Those are the people who are hardworking, smart, competent, ideal for the job, and we don't have enough of them, and we should.

26:05
Yeah, so I. Yeah, and I 100% agree with you, Fred. I mean, I think anybody you talk to and what they see in the market as well. Right? So when they turn on the news and they see, you know, a company go from nothing one day and in 18 months, they're worth, you know, the IPO and they're worth $10 billion. That's what they want to be a part of. And I, I can see the lure of that. I can see what's attractive about that and what excites people about that.

26:31
Like being a star basketball player, you know? And, and there's, you know, thousands upon thousands of non star basketball players out there for everyone who has a happiness and salary, saying, I was about to say that.

26:41
Right. The odds of that happening for you are like being a professional athlete and saying you're going to be a pro when you're in 8th grade. Right. It's. It's. The odds are below 1%. Right. No, but you bring up an interesting point, though, too. Right? And I think we're seeing this more and more in the technology world is less recruiting at the. Must be an Ivy League, must be a prestigious college or must be a college degree, we're seeing more and more going into, straight out of high school or even junior community colleges recruiting there. And I think IBM's put out a big push around at Accenture, I think put out something big around that as well. Right.

27:23
I mean, when you got companies like that saying, hey, look, you don't necessarily need a college degree to be successful in it, I think that's going to be interesting. Right. So how do we make that more attractive and everything for those types of folks that see the value and doing a good job and getting paid very well, to your point, right, where they might not have and if he, you know, the current situation.

27:43
Right.

27:44
So.

27:44
And I think it comes down to basically recruit people, give them an attractive career level offer and then test them and train them really well to make sure you filter out the people who don't have the work ethic or the character, you know, or just their intellect is focused on different strengths, you know, and, you know, let a lot of people come through the system like you would do with any system. Because right now, the mainframe. You know, one of the weirdest things about the mainframe is that the average mainframer is a person of inexplicably extraordinary integrity. Like there's no other organization or group of people I met. And I'm a person of faith. I've been involved with religious organizations and even the organizations of that type they go to. You're going to get a lot of people who aren't for real.

28:25
You go to the mainframe and we filter them alone. It doesn't make sense. It's not possible. And I think one of the reasons, because we haven't gotten new ones in, and that's an issue, but the fact is that the average mainframer is

a person of extraordinary intelligence, competence, integrity and work ethic. And as we start to bring new people in, that value is going to go down. And we have to recognize that program for filtering to help, encouraging, keeping the people valued and being willing to let the other people move on to something else. But we have to restart that whole process.

28:56
Yeah. And I would add every single serious mainframe where I've ever met is very humble. Right? They know what they don't know and they're not going to overstate anything or anything else. They're very confident in what they know and what they've done and the impact they've had on a business. So where's the mainframe going? I try to explain to people the different aspects of the mainframe, and again, I'm not one that knows it inside and out, but you think about the mainframe and its ability to compute and you think about all the challenges around energy consumption and AI and the demand for energy consumption for these new chips and data centers required.

29:43
And you look at the architecture, Linux, one that IBM has had out there for a while now, and its ability to run a significant amount of Linux workloads on the mainframe architecture, you know, honestly replaces a significant with the same power consumption as a significant amount of x 86 servers. Right? So, I mean, I'm probably stepping in here and maybe leading you to water, if you will. But maybe you could tell me more about where you think the mainframe is going, the impact it could have had, and maybe even how it should participate better with cloud in the future.

30:19
You know, the strangest thing about the mainframe's greatest weakness is people who are going to be retiring soon, who have attitudes about the mainframe that are holding it back. And most of those people are non mainframers, and they've been trained in the traditional academic thing and have been taught this ironclad balance against mainframe that is just anti society. But there's people on the mainframe that are also, you know, they have some very fixed ideas about the mainframe and are willing to change it. And those folks are willing to retirement right now. And I, in theory, around the age of retirement, I don't plan to retire for decades yet to come. And there's a lot of us on the mainframe, we're going to keep working.

30:55
But the thing is, those who have some very fixed ideas are bigger challenges to the mainframe's success than anybody else, because the mainframe literally already has all of service that are not even available on other platforms, and yet are absolutely essential for doing cloud computing and all the other leading edge stuff properly. That matter is so far behind us that it's just a matter of getting, you know, letting the people who are standing in the way move on to retirement and letting the new generation just discover what's already there. So on the one hand, IBM is not staying still. The new technology they keep putting out a mainframe is just beyond mind boggling. And now they're even being humbled themselves and putting a mainframe at a rack mounted mainframe who thought that you could do that? And they have.

31:44
You've got just this constant innovation on the mainframe. On the one hand, that is just position for everything from being able to do quantum proof encryption, a real time AI rack model, and that's just the beginning. There's so many things that I didn't even know about. That game just won't stop doing more stuff. But for mobile, let's just forget that stuff and just take what goes already there. The mainframe is the only competing platform on earth that is literally already positioned for the distant future. That it is so well established and so proven that, you know, 1000 years from now, you know, whatever is running world economy is going to have something at the heart of it that is based on the system 360, because there's just no advantage to reinventing that wheel.

32:26
That 360 degree wheel was invented 60 years ago, and it works so well that it's just going to keep on turning unless something that is so vastly unexpected and it just completely devastates all of humanity in society and changes everything. And of course, you can't plan for that if it does happen. So if there's any continuity at all, even the tiniest bit of continuity whatsoever for the next thousand years it's going to be the IBM mainframe that is at the heart of whatever we call computing. It's important to recognize that first of all, it's foundational. It is the foundational computing platform. All the other platforms come and go and disappear. Everything that really find its ways to the mainframe that includes Linux. But at the same time we have to say, well, what is that? Because the IBM mainframe can be emulated.

33:16
But if you run software emulation of the IBM mainframe, it's not really that mainframe because of the hardware strength. It goes all the way down to the most basic hardware, which is optimized and powerful and high volume. For me, it's almost like my job is not to defend the mainframe. My job is, for lack of a better term, and I'm uncomfortable with it as a religious person. We're going to say it anyway, to evangelize the mainframe, just to let people know the good news that it's already out there, it's already working, and they just got to get over their biases. And the nice thing is IBM is increasingly opening it up so you can get a bite sized slice of medium training to do a smaller workload on it.

33:55
If you need something that's powerful, reliable, trustworthy, and is going to continue working the way you design it, you know, into the future so you don't have to completely throw everything out and start again over and over again. Which is so funny because for the first 20 years of computing, that was one of the challenges. You had to keep throwing stuff out and redesigning it. System 360 said you don't have to do that anymore. And on the mainframe, people haven't had to do that. Lesson was in their business interest and other platforms still doing that, you know, and so, you know, all these other platforms are going to come and go and we're going to have other things to come.

34:25
And, you know, one of the things that they were going to have in common, I like to say I'll take a green screen over a blue screen any day, that I had a clunky legacy. And there are people around the mainframe, including Frank Dijilo, IBM's CIO of declunkification. You got all these wonderful people who are declunkifying the mainframe. But clunky or not, the fact is it works and it's going to keep working because that's the secret. That's what is going to make it the computer that when I read a line of COBOL code on the mainframe. There's a very good possibility my great, great granddaughter will be maintaining it, you know, and so that's the future of the Bayframe. It's the future of humanity.

35:06
And the question is, how soon can we take hold of that steering wheel and get involved with it instead of constantly kicking at it as if somehow we can make it go away when it's the one thing we don't want to go away, it's the one thing that we can trust.

35:18
Yeah. Yeah. It's funny you bring up Frank Desolio. I was just thinking about that in my head, right. Because he was actually one of our first, he was our first episode of this podcast. Yeah.

35:28
He's an absolutely awesome dude.

35:31
Yeah, he's great. He's great. And he went into, well, Java's been on the main for him for a while. Right. But, you know, if you're a Java developer, there's a lot you can do on the mainframe as a Java developer, and then Python's available as well on the mainframe. Right. So, you know, there's a lot that, you know, they're doing a, you know, Tyson, new talent or, you know, folks that maybe don't want to go to COBOL, don't want to learn assembler. Right. There's still great things you can do on the mainframe, bright. And to your point, I mean, I think there's, and I find myself. Right. You know, talking about the benefits of the mainframe on a regular basis because I think there is a big gap there, right. And I don't think people really give its credit.

36:08
Like you've, you know, you've mentioned a few times, right. And it does have a place in the future. It's, it's ability to scale, its ability to compute, its ability currently to continue to run complex legacy workloads is going to be around for a while, especially with the distraction of data and AI and everybody trying to get access to data on the mainframe. They're really not. I think there's going to be a point where people aren't focused so much on the application. It's more how do I get either a data and AI platform to run on the mainframe or how do I get that data over to my data platform? I'm not sure if you have any thoughts about that. Accessing data and modernizing data on the mainframe or, I mean, that's a, you know, for me that's a. Because of virtual z.

36:52
That's somewhere I'm heavily focused.

36:54
Well, it is. I mean, there's no question that the data is in so many ways at the heart of the mainframe. The data and the processing that the mainframe posts are its whole raison debt. The reason it exists, you know, that you, the mainframe is a business machine, and the business that's there is to do the business of the world economy. And that business is hosting that data and making sure it stays secure, reliable. You know that it has integrity. So that when you draw that data, whether using AI, you know that using AI, that you don't get hallucinations because garbage in, garbage out. You know, if you give AI garbage in, it's going to get hallucinations that the data of the mainframe is the good stuff. It's the pure, reliable data with integrity.

37:36
We need that place where the integrity is without question built right into the platform and the processing continues that. And so to make that available, you know, basically, as you sort of hinted, the mainframe of the platform that has all these other qualities of service, including the small footprint, the low energy usage, it's just green. And even though it does not require unnecessarily large amounts of staff, you know that if you take a look at the growing of various platforms, the mainframe has grown just like every other platform has. But every other platform, the number of staff needed to operate it has just skyrocketed. And the mainframe, the number has gone down as the amount of mainframes gone up because it's just done so efficiently, so even frugally sometimes. But the major has this incredible centralization.

38:25
And the nice thing is it's got room to expand indefinitely. When we realize that's there, we can move all these other platforms, all these other workloads onto the mainframe either directly on the z os environment, or, you know, one of the other legacy environments, TPF, VM. Or we can move it under Linux, you know, we can move it under one of these rock operators or put it on to containers. You know, you can run it under containers and you can even render containers under z os. So I mean, it's literally positioned for any relevant business or even business related workload you could possibly need into the indefinite future. So it is really just waiting. It's the platform and waiting for everything else that we've tried every other platform and have not been satisfied.

39:11
Yes, that makes sense. That definitely makes sense. So, you know what, is there anything I missed or I should have mentioned or I should have asked you? Right? Like I said, right. I know I don't know. Right. So if there's something that you think is really relevant and you think our listeners should know, please. You know, when I was telling you my.

39:30
My story to this point, I didn't tell you about my master's degree, and I did a master of arts interdisciplinary humanities. That's English, history and philosophy. And my focus was the humanity of the mainframe. And having been working on the mainframe since 1987, so. But I come from a humanities family, not a technical family, actually. So I've been always paying attention to that. And the thing that we forget because of the do not fold, spindle and mutilate movement was really about punch cards more than duty and all this clunkiness. I like to joke the mainframe is that technology has been around for so often that it's often operated by a crank.

40:11
We are clunky on the mainframe, and we forget, therefore, that in fact, the mainframe is, in many dimensions, one of the highest manifestations of humanity in human history in a way that has not been exceeded. You know, that all of the aspirations that found their way to it, expression as business or even academia or science are suitable things to be processed on the mainframe and therefore are not merely its business. It's not personal. No, it is personal. It's personal not just to us as individuals, but to us as a species, that our humanity has been moved forward by computing in a way so definitive that we are literally a different species now.

40:54
We are computing beings and not just thinking beings, and that there is simply no platform that is more responsible for that, even though it's invisible to most of us than the mainframe. And it will continue to be, and so, as we always need to be, re examining what it means to be human, you know, because that is our highest calling, is to be human and to discover what it means to be human and to build what it means to be human. And I'm not a proponent of you got people like Ray Kurzweil talking about transhumanism, how our technology is going to display. So I could go on and on about how technology doesn't even want that job, trust me. And it wouldn't know what to do with it if we gave it to it. It would just power down.

41:31
Because it's like, well, who do I serve then? That the technology's whole meaning is humanity. Everything we've ever invented, the whole meaning of it is humanity. And that is certainly true of mainframe. And so I think that the one thing that we have not really begun to plumb is just the incredible depth of the humanity of mainframes past, present, and future. What it says about who we are and help us become more and more who we can be as becoming more human, not more technical or more something else, but to truly be human. And the mainframe is a definitive place for that to happen. So that's, I guess, the thought I really want to leave you with.

42:09
No, I mean, that's a very interesting point. Right. You kind of say in a poetic way. Right. You know, it created. I mean, if you look at the success of the mainframe, which ultimately was created by humans, and the impact it had on the world then and now, it's pretty significant. Right. And impressive. Right. So I've never even thought about it that way. Right. I mean, that's. We often talk about, you know, going to the moon and. And things like that. Right. But, yeah. The impact of the mainframe on society and humans is very large. Yeah. And it helped us get to moon. That's right. That's right. Well, reg, I really appreciate you taking the time today. This is great. As usual. I learn a ton about the mainframe every time I do one of these calls or one of these podcasts.

42:57
So nothing short of that today for sure. I think I learned a great amount, and hopefully our listeners learned a great amount as well. So thank you so much for that.

43:07
Well, thank you, Mark. It's been a real pleasure. Thank you for the opportunity.