Capgemini

CR050

Wiring your organisation for success, with Gene Kim, Author (Pheonix/Unicorn Project)

CR050

Wiring your organisation for success, with Gene Kim, Author (Pheonix/ Unicorn Project)

Disclaimer: Please be aware that this transcript from the Cloud Realities podcast has been automatically generated, so errors may occur.

[00:00:00] Have you seen, uh, we were just talking about For All Mankind TV show? awesome, isn't it? I love it. I've only just started it. Wait, are you right, are you right, are you right up to date? Yeah, she d s us a s you know, it's so good. No, no, spoiler! No! Spoilers, Jesus. I don't, I don't want to know, I don't want to know who that was.

I don't know who that was.

Welcome to Cloud Realities, a conversation show exploring the practical and exciting alternate realities that can be unleashed through cloud driven transformation. I'm David Chapman and I'm Rob Kernahan.

And you will hear, unfortunately, we haven't got ShSjoukje ouk with us this week. She's on vacation still, so big shout out to you Sjoukje, we're missing you and we'll see you soon.

[00:01:00] But with us this week, we have a very special guest for our 50th episode. We have Gene Kim with us. So Gene is the author of the DevOps Handbook, the Phoenix Project, the Unicorn Project. And most recently, Wiring the Winning Organization, which is his new work, looking at liberating our collective greatness through solidification, simplification, and amplification.

And we are delighted that Gene's joining us, and we'll join Gene in a second. But Rob, before that, 50 episodes, what do you make of that?

I think Dave, just before we continue this conversation, and I don't want to be the pedant on the show, but we have facts. Again, you're already the pedant again. I would like to call it, we've actually done 71, but yeah, yeah. We're not let, let's not get facts. I completely messed that up. Yeah. Sorry. Let you guys aren't very good at this.

You [00:02:00] think after 71 episodes, we might be able to do it, but let's not. What's the phrase about facts? We shouldn't let facts get in the way of a good story. Although it has been an interesting 50 episodes, lots of lives in there actually making it 71, but we're not going to talk about that. No, it's my arcane numbering system that means that this is the formal in studio.

Official 50 good episodes, is that what the way it goes. And thank God we've got a professional guest for it, Rob. I know 'cause it's amateur hour over here, isn't it? We've, uh, we've had lots of fun and one stole a monkey. So that's, uh, that's a chaos monkey. Chaos. I think you're fine. Chaos monkey Chaos.

Thank god you are here, Gene . I'm so glad. Yeah. I'm so, uh, delighted to be here, uh, David, some number of years since we last talked. Uh, so, uh, thanks for having me on and Rob, so good to see you. Good to see you too. Really good to have you, Gene. I'm dying to get into this conversation. So look, let's start with winding back in some of your, you know, earlier and very illustrious works, Gene.

So let's start with [00:03:00] maybe the DevOps handbook and just walk us through a little bit. The handbook and then Phoenix and Unicorn projects. Give us a through line for you about how the arc feels, maybe reflecting on it a little bit. Oh, for sure. Yeah, thanks, Dave. Yeah. So to set some background, the DevOps handbook came out in 2016, and that was really the attempt to sort of codify what I had learned to date studying high performing technology organizations that was started for me back in 1999 when I was the CTO and technical co founder of a company called Tripwire.

It feels like a different world, doesn't it now? I remember Tripwire. What happened to Tripwire? Where did that go? So I was there for 13 years, and I left right after, uh, we had completed our S1 filing to go public. Um, and so I left, uh, we never actually made it to the

public markets. And so I left to work on the Phoenix project full time, which actually came out in 2013.

And, you know, that was, so I'm very So grateful for the Tripwire experience and I can't tell you how many of those lessons learned, [00:04:00] you know, that went into the devil's handbook could have come in really handy, uh, you know, back in 2004 and 2005, the Phoenix project was that business fable about, you know, a technology leaders attempt to, you know, ship products quickly while trying to maintain, you know, reasonable uptime and availability and, you know, achieve security and compliance objectives and failed miserably and what they did about it.

And, uh, so that was, uh, based on one of my favorite books called the goal. by Dr. Eliyahu Goldratt, which was a novel about a manufacturing plant manager who had to fix his cost and due date issues in 90 days, otherwise they would shut the plant down. And so this book sold 5 million copies. Uh, it's taught in most MBA curriculums.

And so the Phoenix Project is really, uh, literally modeled after the goal. Um, and so DevOps Handbook was meant to be the non Fiction prescriptive guide, you know, that would be the companion to the book that says, all right, how did they achieve all those incredible things that were described in the Phoenix project?

And when you look back on the influence of, I mean, particularly the [00:05:00] Phoenix project thing, but the DevOps handbook as well, and you know, you see now that. Many organizations in the world are sort of, if they're not doing it, they're aspiring to do it. And it's a, it's a major part of cloud native culture and cloud native conversation.

What do you reflect on is what you think of the biggest influences that it had? And also, what are the influences that it had that maybe surprised you? Yeah, I, what I am really so proud of in the DevOps Handbook, which I co authored with Jez Humble, so he's, you know, famous for many things, but he was also the co author of the Continuous Delivery book, which I think changed generations of developers in terms of, you know, how do you think about a different way to merge and integrate and test and deploy into production?

Right. Right. And, uh, it was also Patrick Dubois, who famously. coined the term DevOps back in 2009. Uh, John Willis, kind of famous for his contributions in Infrastructure as Code. You know, he was, uh, one of the early chef, people at Chef, and one of the early DevOps days, uh, people with, uh, Patrick Dubois, and also [00:06:00] Dr.

Nicole Forsgren, uh, who I worked with for, uh, on the Accelerate book and the state of DevOps research. She came in and, uh, uh, contributed to the second edition. I think what was remarkable about the DevOps was that I think it was one of the first books to really codify practices that spanned development.

Right. Like, uh, continuous delivery did, you know, QA, like so much of the, uh, you know, the testing and QA books, um, and kind of this next generation infrastructurepractices and platform engineering practices, uh, and put it all into one book. Uh, so one of the things I'm just really proud of in that book is that, uh, there were 50 case studies that weren't just the tech giants, uh, but, you know, half came from large complex organizations, you know, that don't look like.

You know, Facebook, Amazon, Netflix, Google, Microsoft, instead, you know, they represented, uh, some of the largest brands, uh, across almost every industry vertical. So, uh, yeah, I think that was, uh, what made the book unique and, uh, hopefully influential in, you know, people's journeys. The thing that really struck me when I, [00:07:00] when I read the

Phoenix Project and when it influenced me, and it was probably going back probably 10 years ago, actually when, yeah, when I, when I was certainly, and that's when I picked it.

It had its anniversary, didn't it? Cause there was a, yeah, there was a big fanfare about it. Yeah. In fact, it came out, the book came out 11 years ago. Oh, well, congratulations. Incredible and still selling strong, I'd imagine it was the humanistic part of it that really struck me. So, you know, you can, you can talk about these concepts in, in manualistic terms, but the idea that there was a sort of, uh, he's how the humans all interact in these situations.

And you've got that through the novelistic take on it. I thought it was, I thought it was really powerful. Yeah, I think what is, uh, still surprising to me, uh, is just how they seem to identify this universality that happened in any organization. I mean, I think, uh, I can't tell you how many emails I've gotten and how many times people have said, holy cow, it's like you've been hiding in our conference rooms.

You just described the meeting. I came out of it. Just I [00:08:00] know these people. And I think for me, the lesson. And in fact, you know, uh, I say it's surprising, but it's also not surprising because, uh, Dr. Goldratt, who wrote the goal, he said the same thing. He said, how many times do people say, it's like you've been hiding in our manufacturing plant.

And, you know, we just had our boss helicopter in, right. And, uh, you know, described by the, an order that must ship by the end of the day. Right. And yeah, and. It's because I think he identified, you know, these forces that were at work within every organization, right? And so in manufacturing was between sales and manufacturing and, uh, the production scheduling system.

And in DevOps, it was, you know, that chronic conflict between, you know, Dev wanted to ship it. Features quickly into production, right? Ops wanting to, uh, not, uh, ship features because it would always jeopardize uptime and availability and security, right? And so, uh, you know, when you have those sort of forces in opposition, uh, you, you do end up with these common themes and these common patterns of like what goes wrong, right?

And so, uh, there, there are human issues for sure. Although that [00:09:00] point about you hiding in the boardroom, I now have a mental image of you crouching behind a plant taking notes as you record the horror that's unfolding in the traditional ways of working. Just a funny, uh, that's an interesting observation, uh, Rob, because, uh, you know, when people ask where did the Phoenix Project come from?

I mean, it was literally a file I kept on my Palm Pilot called the quote file. Right. And every time I hear something. Yeah, exactly. So it is literally, that's what you were doing. And that's the beauty of it. Is it the anecdotal evidence builds over time to say there's enough of this here now to show that it's an, you know, it's endemic or systemic, whichever way you want to describe it.

But the, um, yeah, it's that, this is the reality. Without doubt, you hear certain people complaining or, uh, you know, you hear and they hear characterized in certain ways, right? And you just realize, you know, there's something really interesting happening here. And so I started that back in probably 1999, 1997, right?

And so, uh, by the time the time. To write The Fiends Project, I had over a decade of, uh, [00:10:00] quotes of things that, uh, I thought were hilarious or insightful that, you know, just had to go in the book. I thought the bit in The Fiends Project that really resonated with me was the, the section of it, and misquote this horribly, Gene, so do feel free to do feel free to correct me here.

But there was a great bit where you showed the lag time between all of the different ticketing systems that are in an organization, and I'm like, yes, that's the quantifiable element that I'm going to try and get to grips with. No, absolutely. In fact, uh, that was something I want to explore in the unicorn project as well.

Is this, uh, you know, what does a worst case architecture look like? Uh, you know, how can you design systems? Plenty of them about. Yeah. I'm probably responsible for a few. Yeah. So the Phoenix project was told really from the ops perspective, right? And the Phoenix project was essentially the same story told from the dev perspective.

And I thought was what, what made the thought experiment so interesting was you take, you know, Maxine, you know, the organization's best developer, right? She's one of these, you know, what, um, You know, some people would sort of think, Oh, no, that's one of the quote, mythical 10 X [00:11:00] developers. You take her, right?

Who's creating greatness all around her. And then she gets marooned in the Phoenix project, which is a wasteland of, you know, technical debt, neglect, uh, of, uh, silos that can't communicate. And, uh, you know, you've basically negated all of her superpowers, right? And she can't do anything. You know, she can't build, can't test, can't deploy, can't, can't see, can't get logs, can't get.

Anything, right? And just as a lesson was really, um, that this is not the fault of the individual contributors. This is a fault of leadership, right? Is that somehow leadership has created a wiring in the organization that has made it impossible for even even the best engineer to do their work easily and well.

It's that it's that thing about creating. How do you create autonomy and empowerment in an organization that still do it in a way that means that people don't go crazy and do the weird stuff? And there is a bit of trust, and it's that you've got to let trust propagate through the system and then make sure you're learning as you go.

And I think so many organizations still [00:12:00] struggle with that. They try and lock everything down because they think that's the way to do it. And it's just So it's counterintuitive sometimes for them to think, actually, no, if you give people the freedom, A, they're happier, B, they're more motivated and C, they do a better job.

Yeah. In fact, can I just pick up that thread because I mean, I think that was really the, the core of the mission going into. Working with Dr. Stephen Spear from the MIT Sloan School of Business on the book, Wiring the Winning Organization, that came out Thanksgiving 2023. And the notion, right, that there's, you know, these platitudes that leaders have to, you know, enable autonomy and empowerment, right?

And yeah, that's true. It's, it's really, how does one do that? Right. And the practical nature of it. Yeah, exactly. And, you know, there was another mystery that, uh, you know, had been on my mind for, you know, going on a decade, in fact, over a decade, uh, which was, you know, both Steve and I. Shared the suspicion.

In fact, let me introduce Steve first. [00:13:00] So he is famous for his contributions to studying the Toyota production system. So he is the author of the famous Harvard Business Review article called that was published in 1999 called decoding the DNA of the Toyota production system, right? Which is that leading to lean and Oh yeah.

No, absolutely. It was, uh, he was part of that, uh, second generation of researchers, uh, looking at, you know, what made Toyota so different, right. And so that was actually based on his, uh, doctoral dissertation at the Harvard Business School. Right. And in part of that

work, he worked on the plant floor of a tier one Toyota supplier for six months.

And so, you know, he extended that word work beyond just the high repetition work of manufacturing to engine design at Pat and Whitney, uh, to helping build a safety culture at Alcoa. And so, uh, you know, after taking his, uh, course in 2014, I felt like there was some, uh, many people pointed out that, you know, the beliefs of DevOps, uh, were so much influenced by the work of, you know, Lean and Toyota Production System, uh, but there are actually very different types of work.

And [00:14:00] so what is in common between DevOps and Agile and Lean and Toyota Production System, and for that matter, safety culture, resilience engineering, psychological safety. And so, you know, after three, um, You know, years of work. Our conclusion was that, you know, there are all incomplete expressions of far greater whole.

And I think the real lesson in the book is that the job of leaders is to enable their people in their organization to do their work easily and well and the right thing. Well, make the hard thing hard to do and the right thing easy to do. Absolutely. And so when you go back to sort of the Unicorn Project and Maxine's plight, right, is how is it that you can take this same engineer, put them in one setting, they are able to do incredible things.

You put them in another setting and they can do nothing. And so who's fault is that? It's not Maxine's fault. It's the leader's fault. And we'll come on, I think to wiring the organization a second, but I do want to dwell on the unicorn project. And yes, I think you were about to say it now. I apologize for interrupting you there, but like, what, what do you think the outcomes of the unicorn [00:15:00] project were?

And are you happy that those have sunk in enough into the industry? Yeah, I, I, I am so. That's actually one of my favorite things I've written. Um, I just, uh, if the F. E. project was meant to sort of point out something, you know, that we've all felt, uh, the Unicorn project was meant to be sort of like the worst case scenario.

Let's, let's see if we can conjure up a scenario. That is so bad. That is so bad. And what's kind of tragic and also. Hilarious. Is that something many people say I recognize the situation. It's like, it's my day job. But it was, it was meant to be a thought experiment of like, uh, and I'm so grateful to that because I don't think I could have, uh, have coauthored the winning the winning organization without that.

Full exploration of the space to say what are all the ways that we can deprive people of their ability to have independence of action right to do their work independently without having to deal with scores or hundreds of other people [00:16:00] for them to be able to work in conditions where you can do experimentation safely to do your work safely without causing even small trouble.

Problems causing giant problems that you have no feedback on your work. You can't see what you're doing, right? And it turns out like those are all the dimensions, right? That, uh, you know, is required to, you know, uh, do your work easily and well. Right. Then if you can't do your work easily and well, there's no way the organization, you know, can achieve the most important things that the organization needs to do.

What I love about that is you've literally theorized the worst possible thing that could happen. And then there's loads of realities that it completely aligns with the people holding their hands up. You just go, how, how is that true? Maybe it could get worse. Who, who knows? But there's like a, I think a lot of people empathized with it and meant it's not just me.

It's not just my situation. I'm in a club of others as well. And I think that there's almost like a catharsis. Through the, this is how it [00:17:00] is that people understand that it's not just them stuck in their reality thinking, how did this happen? It's, there is some commonality with others. Yeah, I'm sorry. Just one of those heartbreaking things is that, uh, you know, because the, uh, uh, Maxine's play, because they have to capitalize, uh, their, uh, development hours, you know, she has a time card.

Uh, and log everything, right? And so not only does she have to face the indignity and frustration of not being able to get anything done, she has to record it in the time log, a good friend, uh, Mick said, Oh my gosh, you know, uh, like we. He's had to deal with that, right? Of course, he wrote the project to product book, you know, CTO at plan view, formerly task top.

Anyway, just, uh, uh, the fact that, you know, you know, I thought that was a bit kind of pushing the absurdity a little bit too far. And I've, I've heard from so many people like, uh, it's happened. It's the reality. It's terrifying. Isn't it? The, the amount of organizations you, you come across that, I mean, literally have sort of textbook versions of this problem, they have technical debt.

That's almost [00:18:00] completely unavoidable at this point. And they have to. Create massive acts of transformation to get themselves out of, out of that situation as something different. And that's not an exception, is it? That is like the norm. And like one of my observations on this is I think that it generally are, and particularly in, you know, kind of let's call them more traditional organizations versus more modern tech organizations.

It had become such the report to the CFO, very financially driven managers, a cost center. And it hasn't had the opportunity to, you know, constantly self modernize and constantly evolve as a function because it's being managed down, I don't know, 5, 10 percent each year in cost. And then they're just going to sweat the environment the whole time and it builds up malfunctioning organizations.

Yeah, that resonates, uh, deeply with me and I think that represents a reality. That I think is, uh, [00:19:00] unfortunately all too common. I sort of almost blame the Millennium bug and then the sort of the initial dot com boom hysteria. Not the work itself, but the hysteria around it, because it seemed like so many expectations in boardrooms were misset by what IT could and couldn't do in the, in the late 90s and some of the issues it might have that then either didn't manifest or they didn't manifest in a big enough way.

And, you know, and boards were almost like You know, like, yeah, another I. T. thing. Let's just shift that over. They're costing too much. Let's push them into the corner, into a box. And actually, we're living with the regret of some of that. Yeah. And if I can just extend that a bit further, and, uh, you know, I think the kind of logical continuation of that is that, uh, technology becomes the order takers of the business.

Right. Just quote, do what the business says, right? Then it becomes a transactional. Uh, in fact, that's an interesting word you use, right? It becomes transactional versus developmental, right? Where [00:20:00] it's, uh, I, I say you do. Right. And, uh, which is sort of the opposite of what we want, which is that, you know, the job of technology is not to obey the orders of the business.

Instead, it's really to, you know, help achieve the. Most important goals of the goals, dreams and aspirations of the organizations that they serve, right? And to un, to liberate everyone's ability to solve those problems. And, you know, hopefully that's a, that transformation is described well in both the Phoenix Project and the Unicorn Project, as well as in The DevOps

handbook and wiring the winning organization.

Well, let's use that as a perfect bridge into wiring the winning organization. And first of all, frame it for us. So tell us about what it was like going into it. You touched on it a little earlier, but let's return to it. What was your thinking going in and where did you end up with its sort of core concept?[00:21:00]

Yeah, right. I had mentioned before, uh, to you that, uh, it was this working on this book with Steve was, uh, the most intellectually challenging thing I've ever worked on without a doubt. Uh, in fact, uh, there were probably, uh, two points or three points where, uh, at least two where. Uh, I was wondering if I was not smart enough to finish this book, whether that, uh, you know, there may be time to walk away.

I feel like that reading books show you. Yeah. Famous five, Dave. How many did you get through? It gets to be a struggle. But it was, uh, it was also the most rewarding thing I ever worked on just because, uh, yeah, I think. Anyone can take something simple and make it look complex. Is it really much harder to, you know, take something complex and legitimately make it simple enough where, you know, you've actually identified something, you know, fundamental.

It's the famous misquoted Churchill thing, isn't it? Which is, he opens a two page letter apologizing for sending a two page letter because he didn't have time to write a one page letter. Yeah. And so, [00:22:00] uh, what was really great was, um, You know, in this exploration is that, you know, I think we came up with a very satisfying explanation of, you know, what is in common, you know, between things like Agile, DevOps, um, and Toyota production system lean, uh, and so much more.

And, you know, the conclusion, uh, is that there's really only three mechanisms of performance. Whenever you see an organization go from, you know, good to great or not so good to great, there can only be three mechanisms at work. And it's because that, you know, essentially those two zones, you know, we call it the danger zone winning zone is that the danger zone is really meant to describe the worst possible, uh, conditions in which to solve problems and, you know, so let's kind of what that is that, uh, you know, you have, you're solving something under enormous time pressure that if you make a mistake, you're Uh, it's catastrophic if it's catastrophic, you can't undo.

Um, and, uh, that means that you can't experiment and learn. So, you know, learning is, you know, and is iterative. It's [00:23:00] experimental, right? And so, like, if you do something, it can be messy and it can be messy. Oh, absolutely. Right. Um, and so if you can imagine solving a really tough problem in production during an outage.

In a highly coupled situation where, you know, changing one small thing can take everything down, right? That's kind of the worst case condition to solve problems, right? And so, you know, let's imagine the opposite. Ideally, you know, what would you want? You want to be able to be doing experiments and learning, not in production, but in planning and practice.

You are working on a small piece of a problem that's You know, decoupled from everything else. That means you can make, uh, small changes. And, uh, you know, in the worst case, it results in a small problem, right? It doesn't serve, uh, ripple out and, you know, cascade into a large problem. Uh, you have fast feedback.

You can see what you're doing. You know, uh, you can see what the results are, which means you can iterate and learn, right? And then you can sort of codify those. You can build up the routines, intuitions, playbooks so that, you know, when the time comes, you know, you can

[00:24:00] actually, you know, do them. Even in the most intense, you know, time critical situations.

And so, uh, you know, that's, uh, what leads into the three mechanisms. So, uh, the first one is about slowification. Right. Right. Solve problems, not in production, but in planning and practice. And so like whenever you see someone doing highly consequential work and they look, you know, they can do it flawlessly, you know, they had to have invested in practice.

Yeah. Uh, the, so the first, uh, slowification is all about sort of moving where that. Problem solving is happening, right? From production to buying practice. You want to push it kind of earlier in time. The second one is simplification, which actually changes the nature of the problem being solved. So the worst case is you have everything tightly coupled to each other, right?

So that you can't change one small piece without impacting the other pieces. So it's really three ways to break the problem up, which is, you know, either you incrementalize it. Right, so small batches, you know, so that's should resonate with anyone doing agile type stuff. [00:25:00] Uh, the second one, uh, way to do it is modularization.

So you take the big problem and partition them into small problems so that they can be worked on independently. And what's interesting is the third one, which is, uh, you can, or you can linearize it. Uh, so sorry, modularization, it should sort of evoke into, uh, uh, your mind, like the API rearchitecture of Amazon in the early 2000s.

It went from a situation where. Even small changes require 3, 000 engineers to have to coordinate and communicate. So a lot of a blast radius on the change, isn't it, where you go, Oh my god! Exactly right. Exactly right. Um, so we can talk a little bit more about that later. But, uh, you know, so they moved into two piece of teams, created, you know, more of a service oriented architecture, which allowed that independence of action, that blast radius containment.

Decoupling. Decoupling, exactly. And so one of the key aha moments, the biggest aha moments is that, you know, the Toyota production system assembly lines. That's like the same thing. What modularization does for parallel processes, linearization does for sequential processes. [00:26:00] So, um, an example of that in our world would be CICD pipelines.

Is that work that was once, uh, all in a big Gantt chart crossing three different ticketing systems, you want to tie together into like an assembly line process, which not only you can automate, but actually creates independence of action, just like modularization. So now the build engineer can do things independently of the QA engineer and the security engineer, right?

And you get this single piece flow. So that's a, like, what a marvelous sort of insight that was. And then so that's simplification and then amplification, you want to create a system that even weak signals of failure can be amplified so they can be decisively acted upon to ideally prevent, but better detect and correct for the problem.

And so, you know, you want a system that is, uh. Uh, rich in feedback that, uh, you know, people can see what they're doing. And let's just imagine the opposite. Imagine a system, uh, where even weak, where weak signals of failure are, uh, suppressed, unheard, or even extinguished, [00:27:00] right? Because I can't imagine that happening.

No, no, no. Generally, things are very transparent is what I find. So, so those are the three mechanisms. Slo fi. Simplify and amplify. And, uh, you know, some of the practices that we

associate with, you know, whether it's lean or DevOps, you can easily slot into, uh, those three categories. Well, let's maybe take each one in turn and have a look at it through the lens of a case study.

If that, if that makes some sense. So let's start out with slowify then. So that was solving problems up to the front of the chain when you're in a position, the way I heard anyways, when you're in a position where you can take your time. Dismantle things and you know, you're not going to have a production impact versus trying to problem solve within the heat of the moment.

Is that have I got the gist of that? Correct? Yes. And you know, what's interesting is, uh, one might observe that slow fi is not actually a real word. So we had to make it up. And the reason why is that We work in tech, we make up words all the time, you're happy, we'll add [00:28:00] it to the dictionary next year, just say it enough, you'll be fine.

Oh, doing this show, I'm so used to made up words, Gene, they just pass me by now. Well, you know, we didn't do that lightly, and our observation was that there's no single word in English that Describes this very recognizable concept or, you know, because we have a lot of adages for this notion that you have to slow down to speed up or you have to stop sawing to sharpen the saw this notion that you're making a short term investment for a longer term gain.

And, you know, we felt that. That maybe the lack of that word, you know, is a reason why that, you know, it's very difficult. You know, there's some people who say, if you can't say it, you can't think it. Well, it's funny that you should observe that because I literally, as you were saying that I was thinking, not only does it lack a word, but actually it lacks quite a lot of implementation in real life.

You know, lots of people do throw those phrases around, but actually the reality of the situation is I'm not sure people really do do that. Or when they do do that, it's such an uncomfortable conversation to get into. [00:29:00] Right. Not exactly. So our hope is by, you know, creating this word, uh, that, uh, it will allow, uh, people to say, Hey, look, uh, we are being forced to solve a problem, uh, under the worst possible case conditions, right?

Uh, either we are solving a problem in production where, you know, uh, we haven't developed the routines and habits and, uh, tooling to be able to even do this work safely, right? You know, uh, you know, maybe we need to slowify, right? And, um. Or better yet, you know, we are potentially entering a situation where we're going to have no idea what we need to be doing right to handle these scenarios or another way that I think comes up so much in software development is, uh, you know, just as neat as even this.

Ability to say, Hey, we need to take a time out here, stop working on features and build some tests for this or, uh, refactor rearchitect so that it is testable, right? Because, you know, currently we have no [00:30:00] feedback, you know, that we're going that we're writing code that's going to blow up in production.

And so that too is a slowification. And I think the important thing is having the presence of mind or the capability to be able to step back and understand there is a structural disadvantage to your approach and those organizations that spot that structural disadvantage and go, guys, this just isn't working.

We need to change your way. Let's embrace the new. Let's think differently. I think that's quite rare still where they can look at. They can look at a thing and go, this isn't working. We need to reboot the way we think about it. And I think there's almost a two tier structure.

There's those companies which will adapt their thinking and they're thriving.

And there are those that don't adapt their thinking and they're dying. It's almost like things like this and the ability to spot it in leadership that's going to create the companies that become the ones that we know the names of the future and those that are assigned to the history books or get bought out or just get crushed.

Because they never changed the way of working and thought about that, those core, real [00:31:00] structural issues. Yeah, 100%. And so one of the goals of the book, uh, was to have this book aimed not just at the technology leader, but also their boss to create a common lexicon, a common set of concepts and terms, you know, to, um, you know, be able to have a.

Conversation to make decisions together to help achieve, you know, the most important goals of the organization. And so because of that, you know, there's 25 case studies in the book of which 20 only 20 percent are technology related. And I think one of the things I'm very proud of in the book is that we're taking these concepts and boiling them down to.

Things that everyone should be able to recognize. And so, you know, the, um, in terms of solidification, you know, something I learned that I thought was just, uh, to help me recognize it was, you know, if you look at, uh, uh, sports teams, right? You know, they go into, you know, uh, in fact, we went to a basketball game a couple of weeks ago, and, you know, they have so many playbooks.

In the coach's back pocket, there are little time cards, little cards that [00:32:00] are quick reference guides of, you know, what decisions are we going to make under these conditions in the third quarter? I mean, you know, these things are not Made on the fly, right? These are the toughest decisions are actually made.

Um, you know, there's like a decision tree that's made beforehand so that you're not making extemporaneous off the fly on the off the cuff decisions, right? Um, you know, during production and so, uh, You know, you have timeouts, you know, that you can trigger, uh, slowification. Um, you know, so all these things are things that you need in sports, but you need also need in manufacturing, you need in software development, and you see these patterns everywhere.

Right. And I think one of the aspirations I have in the book is that this gives a, we'll give technology leaders a way to communicate. Yeah. Here's a signal that we need to slowify, and this is not. I was trying to gold plate something. No, this is, this is required for everyone in the technology organization or team or team of teams to do their work.

Well, there was a point you brought up in the middle of that thought, which was around [00:33:00] common lexicon that I'd love to come back to in a second. So let's just hold that in the air because I think there's something so materially important about common lexicon between business and technology. The fact it's even described differently is an issue in my mind.

So maybe let's come back to that in a second. But I just want to Finish the deep dive into the model first. Yeah. Can I give you maybe a case study? Um, yeah, by all means. Yes. I like two of them come to mind. I'll go, uh, deeper into the second one. The one of the, we had mentioned, uh, uh, we were talking about the Apollo moon landing before we started recording.

And so, uh, studying the landing of, uh, the first Apollo 11, uh, where it has just never been done before. Right. Uh, You know, what do you do when you have no experience going to a situation where no one has ever landed on the moon before, right? How many times a day do

you get that feeling, Rob? I reckon two or three.

So if you can't, if you can't undo, right, uh, where must you invest your effort? You know, it turns out there was a tremendous amount of, uh, [00:34:00] Uh, investment in planning and preparation and simulations in, uh, you know, building high fidelity simulation hardware so that astronauts could get by the time they're doing the first lunar landing, they've actually done it scores of times in production like scenarios.

And it was just so great to see, like, the dynamics that went into that of, you know, just how much was invest invested to make sure that. You know, by the time Neil Armstrong is piloting the lunar lander, you know, 6, 000 feet above the moon is not the first time that he's done something like that. The other thing that occurred to me, because we'd had that conversation when you were describing the elements of the framework was their approach to modularization and simplification as well.

Because when you see coverage of things like Apollo 13, or we were, you know, we were chatting about For All Mankind, the TV show, which have based a lot of their. You know, there's a lot of, you know, real life stuff taken into the drama there. And when they have an issue in that, they break teams off very quickly.

You guys go and solve that element of the problem. We'll take this element of the problem. You know what I mean? Yes. Yeah, yeah, yeah, exactly. In fact, [00:35:00] uh, yeah, because you have to decompose problems so that they can work on separate pieces independently. Right. So that's absolutely. Modularization the for simple for simplification, you know, the, I think the common, most recognizable one for, uh, technology leadership would be the largest, um, in 2011, that was a big AWS, the US West, sorry, the US East.

One availability zone going down. So this is like AWS cloud's largest data centers. Yeah. And so, you know, when that went down, you know, most of the largest customers went down because that's where they put the largest customers. Um, but the one curious exception was Netflix. Which is, and what made it so curious is that they had talked so publicly about running entirely in the AWS cloud.

So how could they be running if they're running entirely in the cloud? And so the, uh, you know, answer of course is, was revealed in that famous blog post on, uh, I think it was April 11th, 2011, when they unveiled Chaos [00:36:00] Monkey, you know. Yeah, that was brilliant. I was just so revolutionary in thinking about actually sending something in to break your systems to test it.

I loved it. Yeah. So, uh, why did they do that? They said for us to, uh, you know, migrate to the cloud, uh, we have to have no single point of failure risks. And they had concluded that the largest single point of availability, single point of failure risk was AWS, right? I think they even said they will never be there when we need the most.

And so what they did was they you. Build this thing called Chaos Monkey that would randomly kill production servers, VMs in the cloud, you know, during office hours. Absolutely tremendous, a tremendous bit of counterintuitive thinking. Absolutely. And so if you are a developer who's responsible for not just building a service, but running the service, and there's something killing your services in the middle of the day, you're going to get very good.

At making sure that in some sort of automated way that you can fail over to something else. And so, you know, that massive investment in [00:37:00] planning and practice, you know, is, is what resulted in this magnificent performance, uh, in production where there were, you

know, running wonderfully well when everyone else was simply down.

Well, let's move on from that to simplification itself then. So building on what we've just been talking about, simplification you talked about in your summary, which was about modularization of problems and breaking it down. What was a case study in that space that resonated with you? Yeah, for sure.

Let's do modularization. Yeah. So we actually studied the, um, AWS, uh, sorry, the Amazon e commerce transformation, the early two thousands and what was so It was actually amazing. Uh, yeah, I had studied that for over 10 years and rereading kind of the, uh, the papers and the interviews of Amazon CTO Werner Vogels, um, then CTO, still current CTO, I, I, you know, it turns out I had missed something really important.

And so, and I actually interviewed, uh, Jesse Robbins, uh, who was a famous, um, [00:38:00] master disaster at Amazon in the early 2000s. So my new understanding of this was that Amazon. In the late 90s, it was a pretty simple software stack. It ran on a Netscape e commerce server. It was written in C The API DOS was written as a C plugin.

It ran on two databases, Oracle for e commerce and, uh, I think it was Berkeley DB for the book database. And it was, it was pretty simple for them to push code into production. They were doing it hundreds of times, uh, a year, but as they added more product categories, you know, they went from two categories to by 2003, there were 35 categories, you know, toys and, uh, music and apparel.

So clothing, right. Uh, so they went from, you know, a handful of SKUs, shopkeeping units to, you know, 50, which each one required a database schema change. Uh, and, and so the result was that, uh, uh, Uh, as the number of teams grew, [00:39:00] the ability for teams to independently, uh, be able to develop, test, and deploy into production went down, right?

In fact, uh, in the interview of, uh, Werner Vogels, he described this ridiculous situation, uh, in the early 2000s where, uh, the digital team, so that's Kindle, Amazon Music, uh, Amazon Video, uh, when a customer would order one of those products, they would still have to provide a shipping address, a physical shipping address.

And the reason was that. Uh, when those teams went to the 35 different ordering teams and said, Hey, could you please change the ordering flow so that you don't need a shipping address for these digital products? They said, we didn't budget for it. You know, we can't do it. So they became stuck. And so the net effect was that they went from doing hundreds of deployments a year to being able to do only tens of deployments a year.

Most deployments did not finish because something would go wrong. And what they, I mean, they had to basically then fundamentally re architect their [00:40:00] system to create more loosely coupled approach. Absolutely, right. So that is what led to the famous, uh, 2003 edict from Jeff Bezos that said, we're going to reorganize the company into these two pizza teams.

Um, you know. No teams can be no larger that can be fed by two pizzas, which can all work independently and autonomously be thrown at Amazon's largest problems, which then required the re architecting of the code base, uh, and the services so that, uh, these hard modular boundaries could be created so that.

Teams could make changes to their parts of the system without having the need to communicate and coordinate, let alone schedule, prioritize, escalate, you know, up eight levels, you know, down eight levels to even do small things. And it's quite a courageous thing to do, to look at something that you built and put blood, sweat and tears into and then say,

nah.

We're going to completely recreate it from scratch to make it do the job that we now know we have to do. Because our understanding of what our problem is has [00:41:00] evolved because we've learned on the way. And it takes a lot of leadership courage to be able to stand back and go, nah, we're going to start again because it's going to be better when we do it.

It's a great point because it's not just companies like Amazon that have this problem. So it's, it's, it's visible in so many problems across the world. And that's whether it's, whether it's built on COTS products, commercially off the shelf products, or whether it's built in, um, kind of stuff that they've built from the ground up these problems that.

Are the technology getting in the way of innovating and moving fast because of decisions that might have made complete sense 10 or 15 years ago now need to be resolved properly and my observation is that so many organizations just try to round the corners off on these problems rather than going right we gotta take a step back.

Oh, actually, we've got a, we've got to do some fundamental re architecting or transformation of the environment to allow us to work in the way that we want to work. Absolutely. So, uh, I love to just bring up two things. One [00:42:00] is like, what was the reward of Amazon doing that? You know what made it was the juice worth the squeeze, they say.

And then, you know, how, how common this problem is across so many organizations. So the Amazing outcomes of that decision from Amazon was that they went from doing hundreds of drawings a day in the late nineties to tens of deployments in the early two thousands by 2011. I think many of you will remember this, uh, John Jenkins said, uh, in 2011 that they're doing 15, 000 deployments a day.

And we're all shocked, right? You know, at one deploy every 11 thousand deployments a day. Seconds. Um, and then in 2015, you know, they came out and said we are doing 136, 000 deployments a day, right? And so it's nuts in it. It shows it works when you think about it. Absolutely incredible. So this is the notion of, you know, creating independence of action so that teams and work independently of each other.

And so let me give a counter example. I just heard last year, which is yeah. So heartbreaking and, uh, and horrible and yet awesome. [00:43:00] Uh, someone said, uh, I'm part of a, uh, a mobile phone company and the top initiative of the year is to put a in front of every one of our 20 million customers a checkbox that allows them to opt into a 5 a month service for email, for music, uh, for movies.

It will take a 40 million. Uh, it will take a year because it has to transit across 40 different teams. Across all four channels to the customer. So three retail, you know, that's physical stores, e commerce support, uh, and so forth. And, uh, most people, uh, we will require daily war room meetings, CEO minus one level support.

And most people will estimate that it, you know, give it a 20 percent chance of success. Why? Because the last two times that tried it did not work. So it doesn't feel good. I wouldn't be the one who got that. So hospital pass. It's unthinkable to think that today we ever stand back and go, what you're actually trying to achieve from a [00:44:00] business concept is relatively simple.

And it's like, I see we've gone back and gone. That's good value for money, isn't it? Lads. Come on. Let's have a think about that for a second. It's that, it's that, that they're plowing

on with it and thinking, hang on a minute, maybe just, maybe we should make some changes to make this sort of thing easy. So what's interesting is that.

What is, so by the way, does this problem resonate with you? I mean, maybe, have you seen a friend, a client struggle with this? So horribly entangled operational structure and change has to transit so many commercial vehicles and team vehicles that it just kills it on the way. And at the end of it all is a project manager who sat in the corner with a tinfoil hat on wondering why the world hates them so much and why they have to.

to tackle this thing, and then when they get it in, there's this huge sigh of relief when it eventually works. But the amount of effort expended to do what is pretty simple, and nobody actually stands back and go, yeah, that wasn't great value for money, but they keep doing it and history keeps [00:45:00] repeating itself.

Yeah, exactly. We see it. Yeah. It resonates. This is the same problem that Amazon found itself in the early two thousands. And so what is magnificent about this example is that technically this is not. A terribly challenging problem, right? I mean, it's more than just, you know, a couple lines of HTML code, you know, I'll, you know, granted, but, uh, you know, this is So a year's worth of efforts, paying 40 different teams, hundreds of people, right?

It's not the technical work that is dominated by it. This is all coordination cost. That's right. And so the language that we put in the book to describe this is that there's really three layers of which work is done. You know, layer one is, you know, the work in front of us. So that could be the code we're working on.

It could be the binary running in production, right? You know, it could be the patient in the hospital. Uh, this layer two is, you know, the tools that we use, you know, the IDE, the, you know, Kubernetes or, uh, you know, the, uh, you know, the MRI machine and layer three is the social circuitry. It is the way which, uh, nodes in the system interact with each other, right?

How do teams interact with each other so that [00:46:00] they can do their work? And so in this mobile telco example, Microsoft It is a failure of design in the layer three organizational wiring. And so, uh, what's incredible, right? If you look at Amazon before and after, right? If you look at, uh, Netflix before and after, right?

The only thing that changed, uh, was layer three, right? The management system. And so, it is not the project, you know, sure, the project manager, uh, is responsible for Trying to get this checkbox projects done, but ultimately it is the responsibility of the technology and business leaders for the wiring that they created that is making work so impossible for people in the organization to do their work.

I'll give you a personal example of that when I was I run a. Large scale cloud transformation program of 10 5 10 years ago is when I came across a phoenix project. First time we were having a problem is just that it was. It was a basic refactoring migration to the cloud, but it was on a very large scale and we were finding that we [00:47:00] could not get to the cadence of migration that we needed to get to.

Fast enough and we were having all the same, you know, kind of sticky board conversations and logjam conversations that you reflecting in a number of your books is exactly like that and actually what we got to is one of the fundamental problems is that it to use your similar version of your words that the wiring of just the IT organization and the tick speed.

The organization had been designed to run out was actually running too slow and it was the tech speed of the organization to solve the organization around it that was slowing down the migration an example of that was you know a decision had been made that any. Things

that get added to the active directory and the objects that actually the directory have to be almost personally signed off by a team of three people is so highly skilled.

It's a 12 month lead time to get another person into that team. And it was like this, right? Then it was [00:48:00] like layer after layer of the onion that we're all getting in the way of actually trying to thread a needle somewhere else. Is that, is that the sort of thing you know? Absolutely. And so the, we chose the word.

social circuitry, organizational wiring, right, as layer three. I like them very much. Those terms are great. And the goal, it's metaphorical, not figurative. We're saying that the organization is like a, uh, electronic circuit, a, uh, like code, like a hydraulic circuit, mechanical circuit. And what a circuit does is it gets things that are stored in one area and moves it to where it's needed.

And so when, to your point, Dave, uh, there are circuits we can build that, uh, where when you need something, you never get what you need when you need it in the right format. And it's always too late. Um, right. And so that's, um, that's a, the circuit is doing exactly what it was designed for. Or you can rewire it so that everyone has what they need when they [00:49:00] need it.

In the right format in the right place, and they're not having to talk to everybody. They can talk to, you know, one person or better yet, you know, they don't have to talk to anyone at all. Right. That too is a function of how leaders design the organizational circuitry. So, you know, we can talk about like how poorly.

Uh, social circuitry can be created, you know, that can be fixed modulization. You know, I think so many of us have also lived through, you know, poorly designed linear sequential processes, you know, like code deployments where, uh, you know, in the Phoenix project, uh, it required 45 different teams, you know, that, You know, you add up all the work to get a VM, you know, uh, test environment to say a developer, right?

And it takes nine weeks, right? And the reason is that, you know, there's three different ticketing systems with three different priorities, right? And that, you know, to get what you need through all of the different work from the different functional specialties requires massive escalation because the coordination mechanism of the, uh, of what was there is just not sufficient.

And so, you know, why is CICD, uh, [00:50:00] so safe? Much better is because those different activities were linearized, you know, we found exactly what we need from all of those people. We put that we tie those work centers together, right? And ideally, we automate them so that the interfaces between the systems are known.

And, you know, suddenly you can get independence of action. Uh, just like modulization did, you know, even though these are sequential activities. And so, you know, again, we can wire our deployment processes so that we get really horrible outcomes, and it takes a long time, or you can wire them differently, you know, like in an assembly line, uh, and get, you know, fantastic outcomes that, you know, take, you know, not weeks, months or quarters, but instead take, you know, minutes, worst case hours, um, and deliver, you know, better outcomes, not just for operations, but for developers as well.

Well, let's use that as a jumping on point to the third part of the model and amplification and where you've seen either good or bad instances of what that means. Yeah. Yeah. I think the, so [00:51:00] again, uh, with amplification, the goal is to create a system so that even weak signals of failure are amplified so that they can be decisively acted upon to better detect.

Correct. Uh, uh, and ideally prevent. And so I, you know, I think for me, one of my favorite examples of this, uh, in the technology space would be, you know, the notion of, you know, what is required, you know, to run, uh, you know, great infrastructure and operations. And, you know, we call that blameless postmortems or, uh, you know, post incident reviews where, you know, when something bad happens, you know, we can, Engineers are trained or, uh, to lead, you know, these blameless postmortems that we, the goal is to create this chronology, uh, not to blame someone, but, you know, to figure out what actually happened, what were the mental models, what did they see, uh, what actually was happening, you know, that led to people, you know, making decisions that they did, you know, that led to, you know, outcomes that weren't so great.

Right. I think, uh, you know, for me, uh, One of the stories that made it into the DevOps Handbook that [00:52:00] I loved was from Randy Schaub, who ran, uh, Google App Engine. So he was the engineering director there. And he said that, uh, you know, whenever they had a customer impacting incident, you know, they would lead these blameless post incident reviews.

And the outcome was that the number of incidents went down, right? Because, uh, they were learning from these, they were putting in good countermeasures, uh, to the point where they didn't have enough Customer impacting incidents to have postmortems on. And so what do they do about it? That's a good problem to have, isn't it?

They get out of the habit of doing postmortems very well because they didn't have any reason to do postmortems. No, no, they made a different decision. They said, instead of doing it for just customer impacting incidents, let's have, do them also for team impacting incidents. So. Uh, when we find ourselves in a situation where there were seven safeguards that were designed to prevent a customer impacting incidents and six of them failed, all right, what happened there and how to make sure that doesn't happen again.

So this is a great example of amplification. And did you look into the, like, the open and closed system [00:53:00] theories around that? You take the open system, which is the airline system we talked about before the podcast, where when a mistake's made, it's understood, there's rules and regulations about, pilots declare it, it's safe for them to do so, safe environment, they talk about it, they understand it, processes, procedures, redevelopment, guess what?

The airline industry is very, very safe. Compare that to the medical System, which is closed and when mistakes are made, they're not discussed and not talked about. And consequently, the rate of preventable deaths in medicine is much higher than it is in the airline industry. And if you actually had, you know, there was this stat that said if the airline industry behaved like the medical industry, two jumbo jets a week would crash and nobody would want to fly.

And it's like completely different systems showing completely different outcomes. And you know which one you'd rather Yeah, in fact, of I'm so glad you brought that up of the, you know, 25 case studies and one of the winning organization, I think a third of them are medical related and they show up mostly in the solification chapter.

And this is heartbreaking. One of the case studies is, uh, we call it the Miss Morris versus Miss Morrison, [00:54:00] where there were two patients in the healthcare system and, uh, the wrong patient was operated on despite 15. Yeah. Uh, events, uh, that, uh, said something is wrong here, including the patient saying you've got the wrong patient, right?

We're still gonna operate on you. Is it still operating? Horrible. And we're laughing. There's nothing funny about it, but it does evoke this horrible reaction. It does. Um, where Instead

of pausing in production, right? Uh, they said, let's keep going or we're going to ignore the signal, right? And, you know, we, uh, described this other case, uh, study where, you know, they were able to decrease the rate of, uh, infection rates for C labs, uh, for this particular, uh, surgical operation, you know, by three orders of magnitude, uh, by doing the opposite.

Whenever something was, uh, Um, not going as planned, they would call it, you know, just like they would in the, uh, in a well run, uh, airline or normal operations and, uh, crisis operations [00:55:00] and, you know, slow flying to figure out, you know, what is wrong, either in the playbook or in the situation, you know, and how do we make sure that we, uh, Don't stay in the fast moving production mode, right?

The fast thinking environment and, and revert to a slower thinking, more methodical analytical mode, you know, to make sure that, um, you know, we don't let habits and routines, uh, take us to where we don't want to go. Right. Particularly if you've got a patient on the stretcher shouting, it's not meant to be me, it's not meant to be me.

It's farcical situation that that can happen. It's just unbelievable, isn't it? But it has happened. You just go. Well, it is a great, um, you know, I know it's a real life scenario, but it's also an excellent metaphor for how many times that actually probably happens in thousands of different scenarios. It's that it's glaringly obvious that something like you're going to operate on the wrong patient, but people press on anyway, let's consider oppression deployment where, uh, you know, involves, you 3, 000 steps, [00:56:00] right?

And someone at one point might say, Oh my gosh, that's odd, right? But let's keep going, right? And two weeks later, you have a massive failure of an ETL pipeline and all the revenue generating servers stopped sending, uh, transaction events, uh, you know, into the big Hadoop cluster. I mean. That's never happened, right?

But that's the consequence of if you're in the middle of a very complex process and you're the one who puts their hand up and says, is this the right thing to do? The consequence of your actions are, it might stop all the pre works, but it's the right thing to do. But then it's that structural disadvantage again, that says the system's stacked against the odds.

So people just go with the flow because the, the concept of them stopping that flow is so great that they, they're just, they, they fear the consequences of the. In other words, they made a decision to, uh, they succumbed to the operating tempo and, you know, instead of pausing, uh, the operating tempo sort of prevented them or [00:57:00] people felt like, uh, they were prevented from doing the right thing.

So, uh, the amplification is based on information theory. Uh, Cloud Shannon, uh, described how signals have to be generated, transmitted. Most importantly, received, acted upon, right? And then, you know, ideally, you confirm, you know, that the correction, uh, corrective action was actually satisfied the need. And I love that because it actually says, you know, these are the things that must happen.

So when you think about, you know, Dr. Westrom's organizational typology model and psychological safety, really, this is a, uh, Represents failures that could be at the generation mode. People can't say what they really think because they, you know, if you do, you get fired or punished, or it could be a problem with transmission or the receiving of it, right?

They're not heard or could be failure upon acted upon. So I'd love this clinical treatment of the social circuitry in this way, because it gives us even better tools to diagnose. You [00:58:00] know, is this a single generation problem or is this a signal transmission or reception problem? Um, and, you know, as, uh, uh, Dr.

Claude Shannon said, right, uh, the most important part, communication success is measured by the receiver, not the transmitter. Absolutely. And there's another dynamic is that, you know, you want not only all these critical actions in, in the, the value stream, but, you know, they have to be done quickly. And I think one, you had mentioned airlines, uh, Rob, the, the application chapter opens up with.

Something that happened in winter of 2022, which is a Southwest airlines cancellation failure. So this is when the big winter storm hit in the United States and, you know, it shut down everybody. There were thousands of airline cancellations, but something really peculiar happened is that, uh, within three days, most airlines resumed normal operations.

In south its airlines, uh, the number of cancellations kept on climbing. Right? And so what, what happened, uh, as, uh, documented in the, uh, uh, business press was that it was this crew scheduling system. [00:59:00] Hmm. So at the end of each day, whenever a crew was not. Where they were supposed to be, uh, and the plane, they would call, they would have to call this phone number, uh, to the crew scheduling service and tell them where they were so they could redo the schedules.

But they were often on hold for, you know, uh, hours or sometimes in one case, like 22 hours. And so, uh, the When the time, when the time to, when it came time to resume operations the next morning, right? The planes were not where they're supposed to be. So they had to cancel more flights. And so essentially they had to reboot operations, right?

So they would fly these empty planes to where they're supposed to go, right? Literally have that same problem in the UK with train strikes. So they'll have, you know, the train unions are quite active at the moment and you'll get days where, you know, your seven or eight services an hour will be down to one or two services an hour because a lot of them are out on strike.

But then actually that lags over into the following day of disruption because all the trains are in the wrong place. [01:00:00] Exactly. So what does it have to do with our work? It's this is now an example of where the control overlay. Layer three cannot keep up with the production environment. Yeah. And so, uh, in the worst case, you have to stop operations, right?

To get, uh, planes to where they need to go, trains to where they need to go, or get the parts in the assembly plant to where they need to go. And so the, the mark of a great Mandarin system is that, uh, it is faster than what it is controlling, right? And so to your point, when you have those active directory authorization processes that take 12 months, right?

You have a control process. Um. You know, that is probably profoundly slower than what it needs to go, right? Somehow you need to reorganize the social circuitry so the decision making can happen much, much more quickly to match the tempo of what is needed.

Now let's maybe [01:01:00] bring our conversation to a bit of a conclusion today by returning to the subject of lexicon and common language between. Business and technology. And I think within that there is something in all of this to me about the role of leadership itself and maybe modernizing leadership thinking to be able to deal with, you know, to be able to deal with increasingly tech driven organization.

So maybe let's start with your thinking on Lexicon and where you got to that. Yeah, that was a great point. In fact, the reason why I laughed is that when you said we need to modernize And I was, I was thinking, boy, wouldn't it be a shame if leaders thought we have to modernize technology, the layer and layer one and layer two parts of the system.

Like, no, no, no. What needs modernizing is, um, leadership thinking the layer three

activities. Right. And you're, you're so right. Uh, and so the absolutely agree with you. Our goal of the book is to help [01:02:00] create. A vehicle for technology leaders to have better conversations with their leadership and their business counterparts and not talk about microservices and CI CD pipelines and open telemetry and Cooper Nettie's right instead, you know, here is how organizations work.

Here's why they work the way they do, right, and, uh, here's what we can do about it. And it's my genuine hope that by creating a common lexicon, uh, by describing how systems work in so many different industries, uh, in the different phases of value creation, technology, non technology, you know, manufacturing design, so forth, that we can help people have an aha moment of like, Oh, here's why it takes a year for us to deliver this checkbox initiative.

You know, that seems so easy, right? Then say this is not a layer one or layer two problem. It's a layer three problem. And this is our responsibility. [01:03:00] And then drives that courageous decision that says, actually, we need to stop and take a step back and look fundamentally at how this organization is wired.

Yeah, exactly. You mentioned the Winston Churchill quote. One of my favorite Winston Churchill quotes is, uh, we shape our buildings and thereafter they shape us. So too, we shape the wiring of our organization and forever after, right? It shapes us. Um, and something that we didn't put in the book because, uh, it was, it seemed a little bit mean spirited, uh, which I'm all for, but, uh, Steve thought it was, uh, it was very important that we didn't make anyone look stupid.

But, uh, if I could just put this here, it's like, this is like what I, okay. What's really in my head. I love this. Um, Dr. Westrom introduced me to the notion of the socio technical maestro, right? A great leader has usually five characteristics, high energy, high standards, you know, great in the large, uh, also great in the small and they love walking the floor.

And I just like, Oh yes. Now that is absolutely resonate with me in terms of like what I've seen, [01:04:00] but he also introduced me to this other quote. Yeah. He called it a rabbi knows law number 23. Um, uh, if you have a dope at the top, You will have, or soon we'll have dopes all the way down. Right. Right. Right.

Right. This has a lot of explanatory power for me as well, because it simultaneously explains, you know, the best systems I've ever seen designed, you know, that are just are enabling people to do extraordinary things and do it easily. And well, also explains my most horrible experiences where personally, you know, uh, not only could.

I not do my work easy and well, but I also hated my job, right? Because even small things required like super heroic amounts of effort. Right. And, you know, even when you did get it done, the outcomes were nothing to be proud of and I deeply resonant, I think, well, look, we end every episode of this podcast by asking our guests what they're excited about doing next. And that could be, I'm excited about a great restaurant. I've got booked at the weekend, or it could be something in your professional life.

So Gene, what are you excited about doing next?

[01:06:00] Oh my gosh. You know, I gotta tell you, now that the book is done, I've been having so much fun. Without any guilt working on generative AI projects. I just, I thought you were going to say sitting on the sofa, staring at the wall. There's been some of that, but you know, I just, I haven't been having this much fun, uh, programming in decades.

It's just, uh, there are so many things that are now within reach, you know, and forget about the things that, you know, not just like these kind of copilots that are available, but, uh, just

to see how good these, uh, you know, LLMs are at, you know. Categorization, summarization, and so I'm just amazed at how many problems can be solved with so little effort these days.

And so I've been spending a ton of time just analyzing emails, you know, summarizing the experience reports from the DevOps Enterprise Summit. And I got to say, I think what's so interesting is that. You know, there's no doubt to me that something amazing is happening in technology, but the job for the technology leader just got [01:07:00] a little bit harder.

Uh, if I can just make a little segue here, one of the other sort of mind blowing things I learned working on the book was, um, Steve telling me that apparently healthcare systems, which are so notoriously complex and, uh, to run these days was actually somewhat easy to run in the 1950s because there were only two functional specialties yet doctors and nurses, whereas you compare that.

You know, 70 years later, now there are scores of functional specialties just within the clinician space. You have, there was no technology back in the 1950s, right? Now you have technology everywhere, each one of them with a layer two technology team supporting it. So even, uh, radiology is not just radiology.

It's, uh, x rays, it's MRIs, it's CAT scans, right? So they now call it imagery. So think about the, the responsibility of the layer three circuitry. Yeah. Which if it's looks similar to the 1950s, it is so insufficient to do what it needs to do now, which is why you have so many horrendous experiences in health care these days.

Interesting. [01:08:00] So technology. So, uh, the layer three social circuitry. Is has to get increasingly sophisticated, the more functional specialties you add, and so the job of the technology leader just got a little bit more complicated because now you have not just dev QA ops information security and, you know, quote the business now you have, you know, the MLOps, AI ops, AI engineers, right?

You have to move dev further to the left. To these ML researchers and these people working with the LLMs, and you have to move those groups to the right so they can work, you know, if you are delivering live results to customers, right, they can't be in the isolated silo, they need to be, you know, coexisting and living with the LLMs.

Right. The production services in production. So, you know, fascinating, fascinating that observation because Rob and I have been kicking the ball around a little bit and we're going to do more on this both on the show this year and sort of generally, generally our day job about what macro AI transformation begins to start to look [01:09:00] like.

Yeah. So at the moment there's a lot of use case dotted use case. And each of those are quite good in their own right. And some of them are actively very impressive in their own right. But it's like, what does the aggregate of that begin to look like? And how does that reform organizations? Yeah. And I think one, there's one conclusion.

I don't claim to have any ability to prognosticate the future, but you know, one thing is so clear. I mean, I think it's indisputable as you increase the number of functional specialties that have to be integrated. You know, into a common purpose, the layer three organization wiring must change. Yeah. Yeah.

And so, uh, and as you increase the number of functional specialties, the job of the leader has just gotten a little bit more difficult or maybe a lot more difficult. And so, you know, I would say. Anyone who thinks that this is going to be an easy change, I think it's going to be, you know, mistaken and which is the reason why we've actually renamed the DevOps Enterprise Summit.

We've done 19 conferences since 2014. We've renamed it to be the Enterprise Technology Leadership [01:10:00] Summit. Oh, brilliant. Because. Well, it becomes the, that becomes the challenge. Yeah. Like if you, if you get the thinking at the leadership level. In the right place, and it's going after the right sort of goals, you get a lot of sort of knock on effect down the structure that you were describing before.

And the reverse is also true that if you don't do that, it's not quite as you might as well give up and go home, but you ain't achieving any macro, um, differently dynamic outcomes. Yeah, absolutely. And you know, that's what DevOps is all about. And that's what I think many of us had in our head in 2010.

But it's funny that over the years, um, now people are saying. I don't, why would I want to go to a conference about CICD pipelines and automated deployment? So it's like, no, that's not what the conference about. And so, uh, I'm super excited to the, uh, conferences because, uh. You know, we've had over 600 organizations, 1, 300 speakers present, and the last year we had a third of the talks [01:11:00] be around generative AI.

Just show us what the frontier looks like and tell us how you're, you know, integrating this into, you know, how you shape your organizations. And, you know, we need to learn because I think, uh, generative AI is, uh, you know, has the potential to impact how every. Parts of technology in our organizations run.

Thank you for that. And good luck with the repositioning of that because actually it is, it's a small naming change, but it's not a subtle movement in the conversation that needs to be had. And I wish you very well with Gene. Oh, thank you so much. And thank you for having me on. And I so much look forward to the next one.

It's been some years since you and I last talked, Dave. So again, congratulations on all your successes. Thanks, man. Roll on the next one. So, for this special 50th episode, a huge thanks to our guest this week, Gene. Thank you so much for being on the show. Thanks to our sound and editing wizard, Ben and Louis, our rejuvenated producer, Marcel, and of course, to all our listeners.

We're on LinkedIn and X, Dave Chapman, Rob Kernahan, and Sjoukje Zaal. [01:12:00] Feel free to follow or connect with us and please get in touch if you have any comments or ideas for the show. And of course, if you haven't already done that, rate and subscribe to our podcast.

See you in another reality next week !

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided every day by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of nearly 350,000 team members in more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering, and platforms. The Group reported in 2022 global revenues of €22 billion.

Get The Future You Want | www.capgemini.com