SRE-S5-E1-SLOs-w-Alex-and-Brian.1.1

MUSIC PLAYING] SPEAKER 1: Welcome to Season 5 of The Prodcast, Google's podcast about site reliability, engineering and production software. This season, we are continuing our theme of friends and trends. It's all about what's coming up in the SRE space, from new technology to modernizing processes. And of course, the most important part is the friends we made along the way. So happy listening and may all your incidents be novel.

STEVE MCGHEE: Hey everyone, welcome back. We're on Season 5. Count em, five. I can't believe it. This is The Prodcast, Google's podcast about SRE and production software. As always, I'm joined by a few folks.

Today we have our friend Matt, and then two other folks who are going to introduce themselves in just a second. But Matt, did you know that not everything is perfect all the time? That sometimes we have to accept problems, they just happen and that's OK? What do you think? Is that OK or should we--

MATT SIEGLER: I've heard of this. I've heard this. This is a thing. Don't let the, was it, perfect be the enemy of the good? Is that the thing?

STEVE MCGHEE: That's right. You can't just demand perfection and be like, work harder, darn it. Come on, people and computers. Anyway, our friends here will have more to say about that. Why don't we have them introduce themselves? Let's go in, I don't know, alphabetical order by first name. That's always fun. Let's do that.

ALEX HIDALGO: Hey, everyone. My name is Alex Hidalgo. I was a long time SRE at Google. I spent the last part of my time there on the CRE team, the Customer Reliability Engineering team, where our remit was to help Google's largest cloud customers learn how to do SRE. And one of the things we learned pretty early was that not everyone speaks the same language.

Right? Different verticals, different industries are going to speak in different ways. And so we had to establish a common vernacular of sorts. And what we figured out was that vernacular had to be Service Level Objectives, SLOs. So a thing I had done at Google previously, but that's when I really fell in love with the concept. I moved over to a company called Squarespace, where they kind of asked me to do SLOs, and eventually, I was tricked into writing an entire book about the concept.

MATT SIEGLER: Sucker.

ALEX HIDALGO: Yeah.

STEVE MCGHEE: Way to go.

ALEX HIDALGO: It was not my intention, but it somehow happened. And now I am at Nobl9, which is a company that helps you really better understand the reliability of your platforms, of your services, and how you are operating in terms of your customer experience.

STEVE MCGHEE: Cool.

BRIAN SINGER: I guess that makes me next. I'm Brian Singer. I'm the co-founder of Nobl9 and the Chief Product Officer. Like Alex, I was also a Googler. I had a company that was acquired by Google and spent some time at Google Cloud, which was my introduction to SRE and SLOs.

I fell in love with the concept also. As a product manager who had also been the founder of a startup, who was the person who got all of those customer complaints over the phone when things weren't working correctly, it was amazing to me that there was an engineering concept that we could implement that could basically say, well, this is what would stop me from getting customer complaints. Can we just manage to this?

And as it turns out, SLOs, Service Level Objectives, are actually fairly difficult to adopt in practice in enterprise. And so that was really the concept for starting Nobl9. Could we use a little bit of software and tooling to make these, the concepts, which in theory are pretty straightforward, but in reality require a little bit of work to adopt, easier to use in the enterprise?

STEVE MCGHEE: Yeah. So SLOs, they're actually, I think, one of the more, not even popular, but it's like one of the things that people are most aware of in the SRE space, when they're new to it. It's like one of the things that people grab onto, like early. They're like, aha, a thing. Let's do that thing. Has that been good or bad for business?

Like, what do you think? Has it been a great success in people just nail it or is it problematic in any way? How have you seen this work?

BRIAN SINGER: We've seen the full spectrum, Steve, believe it or not. So we've seen cases where companies are incredibly successful adopting SLOs. And we can talk about what that means, because you can have success in a number of different ways.

We've seen instances where the adoption hasn't gone particularly well, or where it's been hard to just get the sponsorship needed to get that initiative off the ground. I think the common thread, though, when we've seen successful adoption with customers is that the teams that are responsible for writing the code and running the code are also responsible for creating the service level objectives and using the service level objectives. And I know Alex has some strong feelings about this, but where it gets off the rails is where these things are imposed or created by others, and they become more of a compliance or check the box exercise.

ALEX HIDALGO: Yeah, I mean, the whole approach is really about thinking about what does the user of your service need. And I always try to say user instead of customer, because it's not just about the people external to your business who may be interacting with your company. It's, if you are in charge of a database, your users might be the team down the hall, right? And you need to make sure that you are being reliable in the way that they need you to be.

And so at a certain level, SLOs really need to be bespoke. They need to be artisanally crafted for the best possible outcomes. You need people to really think about this stuff, right? Just slapping an SLO on every single microservice, every API endpoint you have and saying every response has to complete within 200 milliseconds, that's just a good path to failure because not every API endpoint follows the same code path. It doesn't talk to the same back end.

You really got to make sure that you're spending time thinking about how to actually implement this stuff. And so yeah, I agree with Brian, some of the biggest failures that we see are actually companies that really, really want to do SLOs, but it's such a mandate that individual teams and individual engineers even don't have the time to communicate with each other to figure out what these things should actually look like. What should the SLI, Service Level Indicator, be? What should the SLO be?

How are we going to use our error budgets? Because SLOs are communication tools as much as they are anything else. And so if you're not communicating with the people that either depend on you or the people that you depend upon, you're not going to make the right choices.

STEVE MCGHEE: OK, this is kind of pulling a thread in my mind here, the different roles. And let's say your organization has some roles defined. You have production level roles, monitoring and responding to production. You have some product management goals, helping set expectations for what this product is supposed to be doing and how it's supposed to behave and what its users need.

You have some developer roles, which are actually building and crafting and releasing, and they're all working in some ecosystem, big or small. It doesn't matter. And we have these SLOs and they're hopefully being communicated in concert between them.

And let's talk at some extremes. A healthy organization, how is that working out and how are they doing that together? And then perhaps think in the opposite. When does that go wrong and what are some of the smells that you see?

ALEX HIDALGO: Yeah.

BRIAN SINGER: I don't think that there's a one size fits all approach. Every organization we've been into is a little bit different. And I think, just give you an example, you go into certain financial institutions or banks and you just try to log in. And I've seen diagrams where it's hitting 50 or 60 systems on the back end.

So an SRE that's trying to support a login journey that is traversing 50 or 60 systems on the back end has a very different set of things to worry about than maybe say, an SRE that's working at a place like Nobl9, where it's pretty easy to understand who owns each of these systems and issue an edict and whatnot. So I just want to preface by saying, there's no one size fits all approach that we've identified.

But in larger companies, what I've seen work really well is where there is a team that is responsible for the practice of using SLOs and some of these SRE practices in general. They are not necessarily creating the SLOs for every team out there, but they are acting as consultants and advisors, and they are ensuring that teams are following those best practices.

They're almost, in some cases, like arbitrators between the different parties that have a stake in the SLOs. Because SREs in many enterprises just fall into this, a lot of the time, this incident responder type role, where they're not really doing the SRE work of reducing toil necessarily. It is just incident response. And that's where having some of the centralization and oversight to say, well, these SREs, they're going to spend some of their time creating SLOs and making sure that you have the ability to do canary deployments and things like that.

So just that's where I've seen a lot of success, where you have the ownership of creating the SLO with an SRE or developer team responsible for the code base, responsible for the incident response, but this oversight role that is maybe responsible for some of these user journeys that cut across different teams. And that's, I think, the most challenging thing in many companies is where you have responsibility for the reliability of a user journey, but you don't own the underlying code that user journey traverses across a number of different systems. And that is so common, I think, in many companies.

ALEX HIDALGO: Yeah. I mean, like a thing I've seen so many times is people not assigning the responsibility to be an SLO evangelist to either an individual or a team. And it's not like it necessarily has to be the only thing they do, but it does has to be understood that it's part of their role. So you have to carve out time in your sprint planning or however you might manage things to say, this person or this team is spending x percent of their time explicitly being an internal consultant in terms of helping other teams adopt SLOs.

The other thing that Brian touched is once you move beyond a single team, a single SRE team that might own a single service, how do you get ownership of these SLOs of higher level user journeys? And the best way to do this, and I've seen this. Like, it works, is at some level, perhaps the product manager owns that SLO. That doesn't mean they necessarily have to implement it. It doesn't mean that they necessarily have to figure out how to emit the correct telemetry from the code that will be able to inform the SLI, which then informs the SLO.

But they're the ones who look at it and make the decisions based upon the status of that SLO. And as you move even higher, perhaps a director owns it. Perhaps the VP of engineering needs to own it. And again, it's not that they're the ones even picking what the target is, but they need to be the ones owning the decision making process of what the data of that SLO is telling you.

BRIAN SINGER: Now, one thing I would add is just what I would really caution is once you have a central team of SREs that have some oversight across the organization, it's very easy for them to come in and say, I'm going to just create the SLOs for each of these teams or for each of these critical services. And that, I think, is a very slippery slope. Because when those teams don't have ownership over the SLOs and don't create them, it's very hard to get buy in for them to actually use them and maintain them. And SLOs are not a one and done, I'm going to create it and it's going to be the perfectly artisanal, as Alex said, SLO, and we're never going to have to touch this again.

The reality is, they're changing all the time because the conditions of the service you're delivering change, the features in the service change, the customers might change, and you're having probably incidents all the time, which are giving us new data to implement new SLOs or adjust these. So being able to go back and the teams that are responsible for them saying, well, this isn't telling us what we thought it was telling us, we're going to go spend some time on changing this, is incredibly important. Because if that's not happening, then those teams are not going to make decisions based on what the SLOs are telling them. And that really is the crux of what we're doing.

Yes, there are operational use cases for SLOs and we can get into the alerting and what that means. But Alex likes to say these are tools for making decisions, and if the people that would be making decisions aren't bought into what the SLO is telling them, or what it means for it to be read, then it really is a wasted effort. And that is, I think, the biggest risk.

STEVE MCGHEE: So I heard a good take recently, which was that at its core, SREs, one of SRE's main jobs, is turning trailing indicators into leading indicators. Like, taking a graph and being like, I think what this really is telling us is we should look out for the thing that's coming up soon, or at least looking at the past and predicting the future, which is kind of what we're doing as humans all the time anyway. And I imagine that the way that this works into SLOs is like, without SLOs, you have a dashboard of 10,000 graphs and you pick one. Which one is the one that's important?

It's at least a way to centralize, when we're talking about, is the stuff good, and is it going to stay good, and do we need to freak out or prevent or take on a two quarter long reinvention of a thing? Which is the graph that we care about or can we construct a new one? So this is one of those critical, pick the right thing that represents both the technology and what's our bottlenecks, but also how the customer is feeling and where we expect to go with this product.

And a good example of this is like we'll see, sometimes customers will have a service where the SLO is unhappy and they're like, it used to be fine. We didn't change anything. We don't know what's going on. And really what happened was the customers changed. The way that they use your service has changed. The world has changed around your product, and now you're just measuring the wrong thing.

It used to be X and now it's X.2 or something like that. So how do we fix this? Apart from just being flexible and being ready to adjust on the fly, like, do you advise companies to get in front of changes and provide alpha SLOs or something? How do you do this?

BRIAN SINGER: Yeah. Yeah, I can add a couple things and Alex probably, I'm sure, has some thoughts. I think one of the areas we identified is having consistent revisit dates for SLOs, especially critical SLOs, where the definition is revisited. We look at, is it still telling us the thing that we thought it's telling us? And that can be quarterly, or it could be bi-annually or something like that.

But having that consistent cadence. And one of the challenges that we identified with actually running an SLO program is for these central SRE teams that are trying to understand, do we have high quality SLOs, sort of high signal to noise? Are the folks that implemented the SLOs revisiting them?

Are they looking at them on a consistent basis? That's a really good indicator of whether they're stale, so to speak. So that's something that we can help with in product, of course, but also there has to be a culture of doing that as well.

ALEX HIDALGO: Yeah. I mean, a thing I see a lot is that people often get attracted to SLOs, especially in the operational side, like, on the production side. Because they're like, oh, we're going to get better alerting. We're not going to be woken up at 3:00 AM.

That's a very easy story to tell. Right? Like, SLOs are superior to threshold based alerting. We don't care about CPU usage. We care about, is this service doing what it needs to be doing?

And that's a very easy story to tell and it's something that attracts people. But where I've actually seen SLOs be most useful is kind of what Brian was saying, as a historical review. On most of the teams I've been on where we've used SLOs successfully and efficiently was looking at them in our weekly sync. So Steve just mentioned the 1,000 graphs.

You might still need those for investigation later, but SLOs let you converge those into a smaller set of graphs that maybe you just scroll through those and you say, wait, that looks a little weird. Something's going on with that service. It's burning some of its budgets and we don't exactly know why.

And then you assign someone in the next week to go take a look at it, and maybe it's a big deal, maybe it's not. Maybe it's just some dependency acting up. But it gives you a signal to at least say, let's go look at that, right?

BRIAN SINGER: Yeah, I really like it. I use the example of pod restarts. We had a customer recently, where there was always sort of a consistent error budget burn when they had Kubernetes pods restarting, but nothing that burned enough error budget to actually go investigate what was causing that until it started happening more frequently. And they started burning error budget more consistently.

Well, it turned out there was obviously a root cause, something was wrong that was causing these pods to restart more frequently. But it's, I think, a good example of understanding, when is an infrastructure problem or a bug really rising to the level of spending engineering time to go investigate it? I don't think that we have a better tool for that.

And there's one other point that I wanted to make here, which is that when you need to communicate the value of SRE work and the value of reliability work, it's really hard to prove a negative. And that's what I think a lot of people try to do today with using incident counts or MTTR to say, oh, we're doing a great job improving reliability because we've reduced the number of incidents or because MTTR has gone down, but that, I think we all know, is a pretty poor proxy for the actual customer experience.

So the companies that have successfully adopted SLOs, now they've adopted the vernacular of error budgets, and they're reporting on that to leadership. They tend to be able to defend the work that SRE is doing in the investment in SRE, and draw a much clearer line between what they're doing and ROI and revenue, and things like that.

MATT SIEGLER: I think you've made a good case for why and how you build SLOs and what they're for, and the process for creating them and crafting them and engaging with them and a bit of their life cycle. That was a good overview. I'm curious to hear a bit about your experience with organizations that were perhaps resistant, maybe culturally or even technically, and how you push through them, or even maybe the disengagements from them if they said no and never mind, or if they just abandoned them.

I'd love to hear about why, what was their reasons or what happened there because sometimes it just doesn't work out. And those are, I assume, informative experiences. Or maybe it survived, and SLOs reigned and we all got to have cake. What happened?

STEVE MCGHEE: Yeah, what went poorly?

MATT SIEGLER: What went poorly? [LAUGHS]

BRIAN SINGER: I think, believe it or not, as much as I think we want to believe observability is a universally understood concept and everybody can craft great Prometheus queries, it's actually not. And I think one of the biggest challenges is that for a lot of organizations, the engineers expect other people to just come in and build their observability dashboards and tell me what I need to know and just don't want to take ownership of this stuff for whatever reason.

And actually, I think this is what I've seen happen in the last 6 or 12 months is LLM-based AI has actually made the barrier to entry there a lot lower, because it's just so much easier to ask an LLM questions about observability or how to build the right query and get really good answers from Gemini or somewhere else. So that, I think, is an area where AI has actually really made it easier to adopt a concept like SLOs, because there's not as much learning that has to happen.

STEVE MCGHEE: So you're saying the world has changed a little bit, I imagine. One thing to point out, Alex, you wrote some books. Let me see.

ALEX HIDALGO: [LAUGHS]

STEVE MCGHEE: I have them in physical paper copies. Look at these. Look at these things. So you wrote a chapter in this one, and you wrote the whole-- look at that. That's crazy. Which parts of your book would you like to maybe not burn, but retract or change or adapt to the now future or the soon-to-be future? What has changed? What have you learned in the past 10 years, I guess, since then?

ALEX HIDALGO: So there's quite a bit I would actually change. I was asked to write a second edition, and I actually said no, because I don't have the time because I think I would actually change quite a bit. But the major things, like the big things for people who've read the book who are curious what I would do different today, is I would spend a lot more time talking about ways you can use your error budget.

So one of the biggest mistakes I think we made collectively as Google, as Googlers writing the first two Google SRE books, was that we spent so much time talking about how if you have error budget, ship features, if you don't have error budget, focus on reliability, right? That was the basic concept put out there. And that's not how things work in the real world.

One, reliability is a feature. It should not be separated from, quote, unquote, "feature work." And two, sometimes you don't own the code base. So how can you even stop shipping features and work on-- it's a very narrow view that worked very well for most teams at Google.

STEVE MCGHEE: Yeah, it worked for Google.

ALEX HIDALGO: Right. Right. And I've seen too many companies kind of latch on to that concept. I mean, many prospects and current customers that Nobl9 has engaged with, they come to us immediately with that as their only understanding of things. And I tried to deviate a bit in the book, but I don't think I did a good enough job.

BRIAN SINGER: Or Alex, they even say, we don't think we can implement this vision of error budgets. So we're just not going to do SLOs at all. We get that.

ALEX HIDALGO: Right. And so I would spend a lot more time in the book talking about the many different ways you can use error budget data to have better discussions, to have better decisions. How they're the best-- as Brian just mentioned, like a much better tool than MTTR or counting incidents for understanding how you've operated over time. They're essential to conduct proper chaos engineering, as far as I'm concerned. I don't really see how you can do chaos engineering without error budgets.

There's just so many different ways that you can use this data in a meaningful sense. And so that would be the biggest thing. We could have a whole separate podcast about what I would change in the book, but that would be the number one thing is just focusing more on how error budgets give you good data to make good decisions.

STEVE MCGHEE: Nice. So speaking of we went over a little bit of what went poorly, I want to poke at the bear a little bit longer. Where have you found customers, they drink the Kool-Aid or whatever, like they buy into it. They implement it. Have there been backfires or have there been cases where it's like these workloads just don't do well? They don't have an SLO or something like that. Are there places where people try it and they're like, no, bro, like it didn't for reasons?

BRIAN SINGER: I don't think it's quite so black and white like that. I think there's situations that can come up where I would say there's just a lack of executive buy-in more than anything else. And so the failure mode is just this team built some SLOs and this team didn't. And then when they went to root cause, why there was error budget burn, it was because of dependency from a team that doesn't have SLOs and has no interest in building them.

And it's like, what do you do with that at that point? Your error budgets aren't really telling you anything that you can act on. Maybe they are. Maybe it means you need to go, remove that dependency. But that's, I think, more of the frustration.

The other is, and I guess this is a little bit of a Nobl9 plug, but just the tooling. There's a lot of tooling out there that just is not prescriptive enough, or it just flat out does the math wrong and gives bad data. Or it's just really hard to implement, and then you spend a lot of time fighting with the tooling and essentially, building software to do SLOs, even though you think that you're not doing that.

And so that's something that we see folks struggle with. And we get there, and they say, well, we couldn't get it to work. And we've certainly created some converts there, Alex, right? But it's some combination of those. Or the one that I mentioned before, just somebody else imposed this on us. We built a few SLOs so that we could flip a checkbox to say that we had SLOs. Nobody ever looks at them again or updates them, and everybody feels like it was a wasted effort.

ALEX HIDALGO: Yeah, I mean, those are some of the most common failure modes we've seen is this being imposed on people. You really need to build the culture around things, right?

STEVE MCGHEE: Yeah.

MATT SIEGLER: Yeah.

ALEX HIDALGO: We churned one of our earliest customers. I'm happy to just admit that. And when I met with the leader who we kind of worked with to get Nobl9 working in their system, we had a chat.

We met up for beers, and we were just trying to figure out what went wrong because he really wanted it to work, right? He really wanted Nobl9 to take off. And what we realized on both sides it was a cultural problem. No one had spent enough time to get people on board, right?

It was suddenly just a new tool, a new concept. Oh, SLOs are hip, right? And suddenly, this was just kind of thrust upon this organization. And no one took enough time to teach people, here's why SLOs work. Here's why they're important. Here's why error budgets give you better data. And when no one takes the time to do, that cultural aspect of things, they're not going to see the use in the data. They're not going to understand the data.

So part of it is when it's imposed, like what we're kind of like talking about earlier, where every API endpoint has to respond within a certain latency amount. But it's also when just the concept is imposed on people. If somebody says, do SLOs, and there's no one around to teach them what they are, how to pick a good SLI, how to pick a good target, how to look at your data, when you don't have that kind of cultural aspect built in to adopting the process, then you're not going to succeed, and you're going to end up just losing the interest.

BRIAN SINGER: Yeah, so I'm thinking through some of the most successful deployments that have been out there. I'm thinking of one e-commerce company in particular, where what they did was they chose their two most critical customer interactions, their two most critical user journeys. And they said, we're going to go in there and consult with the teams that own those and help them build really, really good SLOs and make sure that they're using those SLOs.

And that's sort of a small enough bite of pizza that it's achievable. And then they took that and showed what they did to leadership and showed what they did to other directors in the organization. And those directors said, I want my teams to be doing that, because if this is working for these most critical user journeys, and by the way, we're supporting those user journeys in this other way, we should be doing that as well.

And that's then where the cultural adoption starts to happen. And folks want to spend the time on crafting those SLOs. And because there is a discussion that has to happen between product managers and developers and upper management that says, what is the definition of reliability for this particular service? Who are our users? And even those conversations are extraordinarily valuable.

STEVE MCGHEE: The lesson I'm drawing from this is like the goal is not to just click Enable SLOs on all my products in my observability suite and just be like, we did it, A+ guys. But like this actually involves thought and some real focus and some--

BRIAN SINGER: Yeah. Look. And I get it. Everybody wants an easy button, right?

STEVE MCGHEE: It would be nice.

BRIAN SINGER: I can't tell you how many customers have showed up and said, hey, can you just automatically create all my SLOs? And Alex has to hold his tongue sometimes because in our business, the customer's always right. But we help them understand how that might not actually be what they're looking for in terms of the outcome of an SLO initiative.

MATT SIEGLER: Speaking of automation and culture shifts, I would love your hot takes or deeply informed opinions on how AI is making all this possible these days. Any thoughts?

BRIAN SINGER: I can add what I'm seeing. I know Alex's opinions of AI have evolved over the last few years, but some of the prep work that needs to be done to create good SLOs, AI can really help with that. So we've seen success taking design documents, taking PRDs, taking obviously some of the code base, feeding that into an LLM with a large enough context window and basically saying, hey, help me brainstorm what some of the SLOs should be like.

You can read what the PRD is. What are some of the things that I should be thinking about in terms of customer expectations and risks? And it's a really good jumping off point for that. Further, I think if you're trying to figure out, hey, how do I come up with the right Prometheus query to get to the SLO that I'm looking for? Again, it's like creating a regex. It can be very helpful and be a force multiplier, and some scripting as well.

Just in terms of even onboarding to Nobl9, I've noticed customers need to create the scaffolding for importing all of their services into Nobl9, or coming up with some reporting or things like that. It can help make that process very productive. But I do think that because we're the users of these products, we still need a human in the loop to basically say, this is what actually makes sense for this particular service. That could change over time, but that's what I've seen so far.

ALEX HIDALGO: Yeah, I'm still a bit of a skeptic in terms of how useful LLMs actually are. I have, as Brian said, evolved my thinking a bit. I used to be very, very vocal about how much I hated the entire concept. I'm still a bit of a skeptic, but, yeah, I do understand that it's a way for people to perhaps build better context, but it has to be built for people. It has to still, at the end of the day, be a human that is figuring out, what should we be measuring? And is this the right thing?

And LLMs may give you a good starting point, but I do still worry that people believe in the technology too much so that when they get their LLM output to help them figure out what to measure or what their target should be, or any of that, that they might still overlook other things. Even if it's giving you good suggestions, you can't guarantee it's giving you every good suggestion.

STEVE MCGHEE: Yeah.

BRIAN SINGER: I couldn't agree more with Alex. I think there was an incredibly good take that-- I was talking with Amin Astaneh, who's an SRE. He's a consultant. And his take was basically, we see a lot of this AI SRE, a lot of companies that are pushing AI SRE, but really it's just sort of AI incident response. And a lot of it is just AI root cause analysis.

So I think just from a nomenclature standpoint, be very careful about what we call AI SRE, because I think at least in this group, my belief is so much of the work that SREs do is actually on the cultural and the human level. And I don't think that the AI can even approach the sort of outcomes that we get with really good human SREs today.

ALEX HIDALGO: And part of the problem that I see there as well is that perhaps some of these technologies can help you resolve incidents and point to what system broke and where. But as a member of the Learning From Incidents community, I do worry that at some point in time, we're going to lose the troubleshooting skills that might be required of humans at various points in time.

Like, at some point in time, the AI is going to fail because, well, I mean, what are we talking about? We're talking about the fact that nothing's ever perfect.

And are you then going to end up in a situation where you don't have the knowledge within your team to actually figure out a much more complex problem? And incidents really should be all about learning. It's not just about finding a root cause. I don't even believe root causes exist, right?

STEVE MCGHEE: Most folks have moved on to contributing factors.

ALEX HIDALGO: Right, exactly. But it's all about ensuring that the people walk away from those incidents, having a better understanding of what happened and how these complex systems are orchestrated and manufactured and how they talk to each other or how the components talk to each other, I should say. So I do worry a little bit that, are we going to lose some of that? In the same way, I've seen a lot of takes that I think I agree with using LLMs to do your coding. Are we going to run out of junior engineers? And then what do we do when we need more senior engineers? Right?

MATT SIEGLER: Seriously.

STEVE MCGHEE: Everyone should read the "Ironies of Automation" if you haven't already. We brought that up a few times in this podcast, for sure. But I'd like to close with one final take, which is maybe hope is not a strategy. If we're hoping for these machines to save us, maybe that's not the plan, or that shouldn't be the plan.

Thank you guys. We've learned a lot more. I didn't know there was more to learn about SLOs, but I learned more about SLOs. I don't know about you, Matt.

MATT SIEGLER: I did. I learned many things.

STEVE MCGHEE: Yeah, and how it works in the real world, not just in the books and the Googles, but out in the world itself. Thank you both for coming on. It's been great. Any final pitches or where can we find you on the internet? What would you like to say about, I don't know, cats or bicycles or anything like that?

ALEX HIDALGO: I'm very allergic to cats. I wish I could love them, but--

STEVE MCGHEE: Or dogs. Yeah.

ALEX HIDALGO: Right. So I'm a dog person instead. But a lot of that is due to the general allergy problems. So that's my opinion about cats. You can find me on LinkedIn. Just search my name, you'll find me. I have a very, very outdated website, but it does have a web form that does still work--

STEVE MCGHEE: Nice.

ALEX HIDALGO: --that you can use to--

STEVE MCGHEE: Is it CGI? Is it Perl underneath?

[LAUGHTER]

ALEX HIDALGO: No, it's a Squarespace site, because I used--

STEVE MCGHEE: What? Oh, OK.

ALEX HIDALGO: --to work at Squarespace.

STEVE MCGHEE: Right on, right on.

MATT SIEGLER: Appropriate.

ALEX HIDALGO: I just haven't updated it in a while, but you can use that web form to contact me if you'd like. I'm, unfortunately, basically not anywhere else on the internet anymore, but LinkedIn and my website.

STEVE MCGHEE: Yeah, we get it.

ALEX HIDALGO: Feel free to reach out about anything you want to chat about-- dogs, music, whatever it is.

STEVE MCGHEE: Cool. How about you, Brian?

BRIAN SINGER: I would say, feel free to reach out to me on LinkedIn as well. That's where I am. Just in terms of what we're doing these days at Nobl9, I think Alex actually framed it really well-- the first generation of Nobl9 was really about doing a really good job calculating error budgets and making it really easy to use that data in a variety of ways. Where we've focused a lot of our attention lately is actually exactly these topics.

As we've gotten more feedback and traction and more and more teams using it, how do we make it easier to adopt SLOs? And especially for those teams that might have some of the responsibility for SLOs but none of the actual authority, which shockingly happens very frequently, how do we make life a little bit easier for them as well? So we're--

STEVE MCGHEE: Right on.

BRIAN SINGER: --just thrilled to be on today. I love what you all do on the podcast.

STEVE MCGHEE: Thanks very much.

MATT SIEGLER: Thank you.

STEVE MCGHEE: All right. Thanks very much, guys. So long. See you next time, Matt.

MATT SIEGLER: Bye-bye.

STEVE MCGHEE: Bye.

BRIAN SINGER: Bye.

SPEAKER 1: You've been listening to The Prodcast, Google's podcast on site reliability engineering. Visit us on the web at sre.google, where you can find books, papers, workshops, videos, and more about SRE. This season is brought to you by our hosts Jordan Greenberg, Steve McGhee, Florian Rathgeber, and Matt Siegler, with contributions from many SREs behind the scenes. The podcast is produced by Paul Guglielmino and Salim Virji. The Prodcast theme is Telebot by Javi Beltran and Jordan Greenberg.

[DRUM ROLL]

SPEAKER 2: You missed a page from Telebot.