Google Prodcast Season Four Episode 7 | The One with STPA, Jeffrey Snover, and Theo Klein

[JAVI BELTRAN, "TELEBOT"]

STEVE MCGHEE: Hi, everyone. Welcome to season four of The Prodcast, Google's podcast about site reliability engineering and production software. I'm your host, Steve McGhee. This season, our theme is 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 remember, hope is not a strategy.

—

STEVE MCGHEE: All right, Hey, everyone. Welcome back to the Google podcast. This is Google's podcast about SRE and production software. Today, we have two special guests. We have Theo and Jeffrey. Jeffrey and Theo, why don't you guys introduce yourselves?

THEO KLEIN: My name is Theo. I'm a site reliability engineer based out of New York City. And I've been working at Google for about 5 and 1/2 years now, specifically with Google Maps. And in the last two years, I've been working on this thing called STPA, which we'll talk quite a bit about in this episode.

I'm super passionate about it. I used to be an STPA hater, but now I'm a convert. And I've been shouting STPA from the rooftops.

JEFFREY SNOVER: That's awesome. And I'm Jeffrey Snover. I'm a distinguished engineer. I've been working at Google for 2 and a 1/2 years. Before that, I was 23 years at Microsoft. And indeed, one of the things that drew me to Google was the opportunity to work on STPA and risk management.

STEVE MCGHEE: Cool. And then we also have this other guy. Matt, you look-- I've met you before, I think. Have we met? Have we met?

MATT SIEGLER: Yes, we have, many times now.

STEVE MCGHEE: Cool.

MATT SIEGLER: I'm Matt Siegler. I'm your co-host.

STEVE MCGHEE: OK. So STPA, those are letters. Would someone like to un-letter them for us, de-acronym them? Or tell us what it really is, even if the letters don't matter. What's going on?

THEO KLEIN: I mean, STPA stands for Systems Theoretic Process Analysis. And what it really is is a novel way of analyzing complex systems. So traditionally, when we look at outages, we think about the five Whys. And we think about a time series of events that leads to some unacceptable outage.

And when we think about the five whys, we say, OK, go back through the timeline, and then we find the fifth why, and that's the root cause. But Systems Theoretic Process Analysis takes a completely different view on hazards and losses. And what it says, it claims that accidents and losses happen when we lose control in the system. Some part of the system is misbehaving, and it's a control problem, not a failure problem. So I don't know if, Jeffrey, you want to take it from there.

JEFFREY SNOVER: Yeah, that's exactly correct. So this is grounded in safety design, came out of MIT. Again, all kind of grounded in this control theory. So that's MIT, right? So it's super good and then lousy names. [LAUGHS] But that's exactly correct.

And so there are some circumstances where there is no acceptable loss. Data privacy, data integrity, things like those-- there, you really can't tolerate loss. If you think about it-- when I talked to Ben Treynor about coming to Google, he said, Jeffrey, the heart of SRE is to manage this tension between reliability and innovation.

And the way we did that was by coming up with this loss budget, the error budget. And what that says is, hey, we're going to define an error budget. And then you can go as fast as you want until you incur this amount of loss. And then you got to stop and then reset.

And that balances reliability, because he said, I know how to give you reliability. Get it working and cut off the check-ins. But then you don't get any innovation. Because I can give you all the innovation you want as long as you don't want to have any-- as long as you don't want to care whether it is up or not.

And so basically, the loss budget achieves that. He says, but the challenge is, how do we take SRE to the next level? How do we get this balance between reliability and innovation without the loss? And so that was the challenge he posed to me to come to Google and work on that. So risk management was a big part of that. And then STPA is just a fantastic tool in the risk management toolbox.

STEVE MCGHEE: Gotcha.

MATT SIEGLER: Yeah. I'd like to first talk a bit about what control means here. Tell me what you mean about control. That's a phrase that I think I know what that means. But I think you have a specific meaning in mind here.

THEO KLEIN: Yeah. "Control" is a really important term in STPA. When you think of a complex system, you have a bunch of different independent actors that are making actions in that system. So you can think of a human as a controller. Your computer is a controller in some sense. The environment is a controlled process.

So when we talk about control, really, we're talking about actions that constrain controlled processes. So an example of control might be something-- so in the system, let's say of a thermostat in a room. I'm a human in a room with a thermostat, and there's a boiler.

As a human, I can control the temperature of the room via the thermostat. And that goes through the thermostat, into the boiler, into the room. The room has no control over the temperature. It just is. I, as a human, have the most control in the system because I have the most agency.

The thermostat has some control over the boiler, but the thermostat doesn't have control over me. It just has control over the boiler and transitively to the room. And the boiler controls the temperature of the room itself. So control is really about authority and the actions that you can impose on the controlled processes.

JEFFREY SNOVER: Yeah. It is a different way to think about the problem. And it takes a little bit to get your head through the knothole. That's why I think Theo mentioned he was originally an STPA hater. It takes a little, like, what, what? But then you work through it a couple times. And then you're like, oh, I get it.

And so the heart of it is that this controller performs actions on a controlled process. Well, how does it do that? Well, first off, it has goals. I want to be comfortable. So you got to know what your goals are. And then it has a worldview. So where do you get a worldview? Well, you get a worldview from information, external information or information from the control process.

And then you take that worldview. You compare it against your goals. If everything's aligned, happy days. You do nothing. But when the worldview is in conflict with your goals, then you have to do something. And you do something, which is to say, you perform an action to make the world more aligned with your goals.

So I'm cold. I look at the thermostat. Oh, no wonder. My damn kid said it's too high or whatever, too low. So I'm going to change it. So then I change it.

STEVE MCGHEE: How different is this from declarative versus imperative systems? You're kind of setting a goal, and then you're letting the system do its work. We've talked publicly about an Annealing system that has been developed inside of Google and the production world.

And then also in Kubernetes, we have these operators, where we're setting some intent. And then we're letting the operator observe its system and collapse the wave function into something where your intent becomes true. It seems like these are cousins, at least, or similar to each other. Is that right?

THEO KLEIN: Yeah. I think the systems that you described can definitely be modeled in a control structure, where these agents are controllers, and they are very clearly attempting to control some underlying system. So it models control but also feedback. And Jeffrey was describing that feedback. And you were also describing that feedback, Steve, about observing. So you have an intent, and you observe the production system. And you're saying, ah, the production system is not in the state that I want it to be in.

So you're observing. That's feedback. And then you process that feedback, and then you execute an action. And so that's the feedback. There's a control and feedback loop. Now, the problem is that there are problems that can occur when some part of that loop breaks. Either I misinterpret the feedback, I believe-- so for example, the production environment is in a bad state.

Let's say we're having a complete outage because we have the wrong binary release. My system that processes the production environment, incorrectly observes my production environment. And it says, oh, I have the correct binary in production. And therefore, it does not issue an action.

The result of that is that my outage persists, even though I have told my production manager to remove any binary that is, quote unquote, "bad." So that's an example of it being unable to interpret the feedback correctly and as a result, letting an outage persist.

So these feedback loops are really important. And that's really at the core of STPA, is modeling these control and feedback loops and then identifying the conditions under which these control and feedback loops misbehave and, as a result, achieve an unacceptable loss.

And when I say unacceptable loss, what I mean here is we have a loss of revenue, or our lawyers get upset, or XYZ problem that maybe I can't talk about on this podcast, something that is truly unacceptable. And we try to figure out the ways in which this system might result in that unacceptable loss.

Now, Steve, I want to go back to your example because you actually produce a perfect example to highlight two of the critical aspects of STPA. So you said, hey, what's the difference between basically, a desired-state configuration-- desired-state controller, where you set things and it goes, versus an operations-oriented model?

Well, they're both controlled processes. And they're patterns for controlled processes. One is where the human gathers the information, compares it against things, and then it performs-- you perform the control actions. The other is where you say, hey, I express my intent. And then the system does that loop.

And then those are two different models. Both work with STPA. They have strengths and weaknesses. But one of the great benefits of STPA is that it is a system that allows you to analyze things that are sociotechnical, which is to say, some of those decisions are made by the system, and some of them are made by you.

So for instance, even the case of desired state configuration, you made the configuration. What was your process for doing that? And often, that has controls and is informed by systems, et cetera. And it is often the source of error, like, oh, the system worked perfectly. I just gave it the bad configuration.

STEVE MCGHEE: Are you suggesting humans are fallible? Jeffrey, how dare you, sir.

JEFFREY SNOVER: It turns out.

MATT SIEGLER: I'm following you here. It feels like this is a bit theoretical. But I'm getting-- I'm feeling us getting towards something practical pretty soon here. So we're going to do some modeling here where you're modeling the behavior of something that maybe traditionally, we had some implicit belief in how it behaves.

We're going to model it a little deeper, a little deeper than we used to. We're going to model these-- we're going to model these losses in a different way. What is this going to buy us? Why would we go to this trouble? Because it sounds like there's going to be some trouble. I'm going to get some payoff. Why am I headed in this direction? Presumably, because of a payoff. Tell us a little bit about that.

THEO KLEIN: Yeah. So that's-- I mean, that's a really great point. Ultimately, everything comes down to cost and benefit. And as site reliability engineers, my benefit is making sure that our systems are safer. Now, what tools do I have in my toolbox to figure out whether or not a system is safe?

Well, I can do design reviews. I can do code reviews. I can implement n plus 2 redundancies. I can implement SLOs. But none of these really get at the core benefit of understanding whether my system is safe. I don't really-- unless I-- in my design review, I try and really go down this random walk of all these possible edge cases.

I don't have the framework through which to systematically review the safety of my system. And STPA-- and this was a process that took us a long time to get right with software. But STPA allows us to very efficiently identify key design flaws before we've even written our code.

So I'll give you an example. I ran an STPA on a system that had not yet been built. All right. It processes road disruptions, takes as input, information from the world, and it figures out which roads should be closed on Google Maps. And this design seemed really simple, read a file, figure out what's changed, write it to a database. Couldn't get simpler than that.

JEFFREY SNOVER: How hard could it be?

THEO KLEIN: How hard could it be?

STEVE MCGHEE: I can do that. Come on.

JEFFREY SNOVER: What could go wrong?

THEO KLEIN: What could go wrong? Now, we look at this design, and everyone looks at this design and gives us a thumbs up. Let's launch. And then I said, hold on. There's this new thing called STPA. And it allows us to analyze designs that are relatively-- well, we think this might be simple, but there are a few steps in here.

Let's analyze this. And give me-- and I tell this to my developers. And I say, give me an hour and a half of your time, and let's run STPA on it. And let's see if we can catch any design flaws. And if we catch a few, maybe you'll want more of this STPA. And I say--

STEVE MCGHEE: This feels a little bit like formal methods. Is this where you look back at your system, and you like to find the algorithm in TLA or something, and you run a checker against it, and you're like-- is it that level of system, or is it totally different? Am I getting that wrong?

THEO KLEIN: Well, that's a great question, and it actually comes up a lot. People ask, oh, I've used TLA+. And this is not-- this is not like TLA+.

STEVE MCGHEE: This is not that. OK.

THEO KLEIN: This is not that.

STEVE MCGHEE: TLA is hard. I don't want to do TLA. [CHUCKLES]

THEO KLEIN: And I think STPA is hard to get right, but it's hard for a different reason. It's not tedious. And you aren't trying to exhaustively define the behavior. Actually, you look at the abstract responsibilities of all of the different parts of your system. And so what we did is we spent an hour and a half with these developers of this road disruptions pipeline. And we found something like three system gaps in an hour and a half.

STEVE MCGHEE: Can you describe any of those in particular? Like, what's the detail?

THEO KLEIN: Yeah.

STEVE MCGHEE: Is it like they forgot a semicolon, or is it something more fun than that?

THEO KLEIN: No, it's--

JEFFREY SNOVER: Yeah, great question.

THEO KLEIN: It's like a fundamental requirements gap.

STEVE MCGHEE: OK.

THEO KLEIN: So the one that I give in my talk is the one where essentially, we-- the impact of this gap is that we have a logical flaw. The result of it is that we forget to add road closures onto the map. All right. So imagine this. You have a parade in New York City.

And we've known about this parade for months. And we could have added the road closure across Brooklyn. And yet, for some reason, we don't add the road closure onto Google Maps. And we navigate users across that parade. So now you have a backup of cars trying to cross a parade.

JEFFREY SNOVER: No bueno.

THEO KLEIN: And then we have this mega PR event. So that's the loss. All right. So that's-- so then the question is--

STEVE MCGHEE: Seems bad, got it, yeah. For sure.

THEO KLEIN: We don't want that. So how can this system result in that loss? That is the question of STPA. We try to answer that question.

MATT SIEGLER: Presumably, you did not a priori say, let's not have active roads, cross parades somewhere. That didn't go into the analysis ahead of time.

STEVE MCGHEE: No.

THEO KLEIN: So no.

MATT SIEGLER: So how did you discover losses without being so specific ahead of time?

THEO KLEIN: The losses – we send users down a street that has a parade. All right. Concretely, what that means is, we don't have a road closure in our database on Google Maps, where there should be a road closure. All right. So there's a missing road closure in our database.

So you ask yourself-- and now, I'm going to be paraphrasing a lot of the mechanics of STPA here. But essentially, we say, well, how on earth could that happen? How on earth could it be that we don't add a road closure when we should be? So we have, in our control structure-- the bottom part of our control structure is our database. That is the system that we're controlling.

And we have a controller, which is another piece of software, whose goal is to ensure that all active road closures are on Google Maps. And so it has a control action of, add a road closure or remove a road closure. And so then the question is, how could it be that we don't add a road closure when we should have?

Now, that's the question. And then we look at the feedback. We look at the control actions. We look at the logic inside of that controller. And in this particular case, we discovered that the logic inside of this piece of software was flawed.

So how on earth did this piece of software figure out when to add a road closure? Well, let's look into it.

What it did, gets a file. All right. And the file contains all active road closures. And then it gets another file, and that's version 2. And what it does is very simple. It diffs the two files, and it says, aha, here is a new road closure in version 2. Let me add it into Google Maps. But what happens if, in the process of adding this road closure, it fails to do so?

STEVE MCGHEE: Oh, like the diff just doesn't diff or something like that.

THEO KLEIN: The diff doesn't diff, or maybe the--

STEVE MCGHEE: Or it's a crash or--

THEO KLEIN: --HTTP request.

STEVE MCGHEE: Yeah.

THEO KLEIN: Exactly. The HTTP request fails, no retry. Or maybe there's a maximum set of retries, and it fails all three retries.

STEVE MCGHEE: Yeah.

THEO KLEIN: All right. You see where this is going?

STEVE MCGHEE: I have a guess. Yeah.

THEO KLEIN: Version 3 is compared against version 2, no diff.

STEVE MCGHEE: Yep. Should be fine.

THEO KLEIN: Should be fine.

STEVE MCGHEE: Parade away. Have a parade.

THEO KLEIN: Parade away, exactly. Well, according to Google, there is no parade. And so the flaw here is in assuming that the previous version of this file is equivalent to the state of Google Maps. Now, if you were to look at this from a data flow diagram, you would say, oh, of course, you just diff, and then you apply. And if you fail, you reapply.

Or even if you don't think about failure-- I mean, that's sometimes the case, where we don't think about this edge case. And this is not-- I'm not trying to disparage anyone. Really, this is the state-- the number of states that you have to think through in order to exhaustively identify all of the failure modes is enormous.

I mean, it's 2 to the n, where n is the number of parts in your state. So with STPA, you start from a top-- you start from this top abstract view, OK? Here is the unacceptable loss. And then you dive a little bit deeper. And you say, how could this happen? And then you dive a little bit deeper. And you say, OK, it must be this control action that could possibly lead to this loss.

And then you ask another question. OK, so under which conditions could this control action be unsafe? And then you might even dive a little bit deeper if you'd like. And then see that your diffing logic is flawed. And that is an example of one case where you lead to the loss.

STEVE MCGHEE: This is a great-- I'm really happy. This is a concrete example because I've never heard one before. I've heard STPA so many times, and they've been like, it's clearly a matter of risks and rewards and Ps and As and something like that. I'm like, yeah, but what is it really? So this is great.

So my first question or my-- the thing in the back of my head is like, when you went to go to analyze this, you had to use your smart human brain to come up with that high-level abstract risk of, I don't want cars to cross road closures. That was the thing that drove the rest of this analysis. Is that right?

THEO KLEIN: Yes.

STEVE MCGHEE: OK. So the step one is thinking about the abstract unsafe behavior--

THEO KLEIN: --the losses.

STEVE MCGHEE: --of a system. The losses, thank you, that's the term. And then you go through this process of figuring out what could then drive those losses to occur in reality. Is that--

THEO KLEIN: Exactly.

STEVE MCGHEE: Sound about right?

THEO KLEIN: That's exactly right.

STEVE MCGHEE: I think we got it. Matt, I think we're done.

THEO KLEIN: You got it.

MATT SIEGLER: Well--

STEVE MCGHEE: Let's call it.

MATT SIEGLER: --can we talk a bit about the relative concern of different kinds of losses?

STEVE MCGHEE: Yeah.

MATT SIEGLER: You can have very unusual losses, like rock falls out of the sky and causes a damage to the road. And that's a surprise. But that probably happens almost never. But it does happen. Then you have a parade, which is rare, but does happen once in a while.

And then you have a traffic jam, which happens everyday. I'm going to guess there's some probability entering into this. Do we have some of those things happening in there?

THEO KLEIN: That's an interesting question. And in the world of systems safety, probability is a hot topic. And depending on who you ask, someone might get upset that we brought up probability, in part, because it is actually very hard unless you have many, many samples.

It is actually very hard to provide an accurate probability. And so what we've been trying to do, at least within GEO Google Maps SRE, is by using a different metric to prioritize losses and mitigations. So really, across all losses, there is a-- every loss is unacceptable.

So some may be more unacceptable than others. But truly, we don't want any of them to happen, by definition. Then when we look at ranking loss scenarios, we think about the number of ways that a loss scenario can occur. And then, actually, what is more important, I think, is the cost of the mitigation.

So how costly will it be for us to prevent this loss? Maybe we can-- for 20% of the cost, we can get rid of 80% of the problems. And that's really just the cost-benefit where we start horse trading.

JEFFREY SNOVER: And it's important at this point to draw the distinction here. And now, we're talking about a set of things that deviates from the MIT STPA approach, right? So from them, they're sort of absolutists. Hey, this is used for the design of a nuclear plant. This is used for an aircraft carrier. And for that, you got to solve all of them.

And it doesn't really work for our world. [CHUCKLES] So we have had to take STPA in its pure form and then say, hey, in a world of commercial software development, you need to make some changes. And so Google's been leading that way. We've been documenting those things. Theo and others have been giving talks about how you take this great technology. But then you apply it in the real world for software systems and to be effective.

MATT SIEGLER: Yeah, I was going to say that, in the software world everything is possible.

JEFFREY SNOVER: Everything is possible, baby.

THEO KLEIN: Yeah. Now, to go back to this Pareto point, what we end up doing within Google is actually performing a 20% of an STPA. We rarely have the time to complete an STPA, and usually it's because we find so much at the 20% mark.

So to complete this road disruption story, we found three problems within an hour and a half. And the developer said, more, please. Let's do another session. And so we did another session. And overall, we spent about 27 hours across five engineers. And we found seven design flaws, seven design flaws.

And this is 27 SWE hours. Now, these 27 hours included the cost of fixing these problems, which was so cheap because we hadn't built the system yet. It's a super efficient design review. And we were able to solve these design flaws by rewriting the design document. And we only analyzed, I would say, 20% of the control structure in order to find these seven design flaws.

MATT SIEGLER: This sounds a little bit like test-driven development when you're at the engineering point but upstream of it at the design point. How thorough do I need to be in my design before it can be useful? Because lots of designs are not particularly thorough.

They're not deep into their trees of all the different states you could get at. When can you show up and actually apply this? I haven't figured out all my edge cases yet. I'm on iteration 2 of 5,000. When can I get to STPA? How mature do I need to be?

THEO KLEIN: So the good news is that you can use STPA before you've actually designed your system. We have started experimenting internally, and it has been done outside of Google already. But we've been using this internally where we use STPA to design our system.

We start with the control structures, and we define the goal of the system that we want to build. And we think, we think, in the abstract, OK, what controllers do we need to make sure that these losses cannot be achieved? And then we say, OK, we need this abstract controller to constrain this process.

We need some system to ensure that our database is never in this state. And then, once we've defined these safety requirements, we can then go to the paper and say, how do we build that? And here are the safety requirements.

JEFFREY SNOVER: What we found a lot is these feedback mechanisms are broken. Second biggest thing, go and say, hey, what's the goal of these system? Silence.

[LAUGHING]

It's like, wait, no, I'm just asking what's it supposed to do. Silence.

MATT SIEGLER: A lot of implicit goals.

STEVE MCGHEE: Yeah.

THEO KLEIN: Or sometimes, we don't even realize what we're trying to achieve.

JEFFREY SNOVER: Yeah, that's the point. And so, again, back to how STPA can be helpful. You got to start with, what are your goals? What does success look like? And then, from there, you say, OK, what does loss look like? And then that can guide you to designs and decisions.

STEVE MCGHEE: So it sounds to me like STPA is not a compiler. It is not a system that looks at a system and goes like, here's 16 errors. It's like it's a discussion technique. It's a human-to-human conversation over some hours on a whiteboard and a lot of coffee, probably, like Red Bull.

THEO KLEIN: That's entirely correct.

STEVE MCGHEE: Cool.

THEO KLEIN: It is a human-driven process. And in some ways, it's a question-prompting engine, is what I think of.

JEFFREY SNOVER: And since we're talking to an SRE audience, a lot of people will say, well, hey, how does this compare to reliability? So Theo, you've got a great example of the difference between reliability and safety. Have at it, my friend.

THEO KLEIN: Thanks for setting me up on this one. I have a prop. I do have a prop for this. So the idea, the difference between reliability and system safety can be conveyed with the butter knife. Now, the butter knife is a reliable component, right? I mean, I can't break it. And I'm really putting in all my energy.

And this is a good Google knife, not branded Google, but it came from it, property of Google. Thank goodness, I can't break it. And I can perform-- it can perform its responsibilities well. And I can cut butter with it, and I can do so safely. I won't get hurt by using it to cut butter.

But what if I took this butter knife and I shove it in the electrical outlet? OK. I'm going to be electrocuted.

STEVE MCGHEE: For the audio-only audience, Theo showed an outlet and put it-- that was uncomfortably close to the outlet, man.

THEO KLEIN: Too close, I know. I apologize.

MATT SIEGLER: You've practiced that move.

THEO KLEIN: I have, yeah, more than once, actually.

STEVE MCGHEE: So this system you're describing is unsafe, despite it having reliable components. Is that what you're saying?

THEO KLEIN: Exactly. So the outlet's goal is to provide electricity. It's a very nice goal. It's very good at that. It's very reliable. The knife is to remain a knife and to be metallic. And it just so happens to be conductive. All right. And I'm also reliable. I'm reliably curious.

And I do wonder what happens if I shove a knife in an outlet. And as a result, I achieve a loss because I am irreparably damaged.

STEVE MCGHEE: I love it.

JEFFREY SNOVER: Well, this highlights again, one of the great elements of STPA, the thing that differentiates it from a lot of other technologies and approaches. And that is STPA is great at finding flaws, losses for every component of the system works reliably and as intended. But the system produces devastating losses.

And STPA is the only mechanism that I know that provides the context for finding those systems. In fact, the 737 MAX system, that was an example. All the components of that system worked as designed, worked reliably. But there were unexpected interactions between those systems that then resulted in a terrible and devastating loss. So again, one of the things you get out of STPA that you don't get out of anything else.

STEVE MCGHEE: Can I show you a book? Do you guys like this book? So this is Thinking in Systems by Donella Meadows. Yeah. This is one I like. I think this is-- it's a good introduction to this type of thing. There's also Systematics, which is another good one. That's a little bit more technical-- or Systemantics, I think. It's kind of an older book.

But if people haven't looked at the books around systems thinking in general, I highly recommend it. Like, for whether you're going to STAMP or not STAMP, like, just SREs in general. I think, looking at this type of way of thinking about components versus systems is super, super imperative.

THEO KLEIN: 100%. And just one thing on that, I am so grateful that I learned about STPA, not only because it helps me in my work at Google but also because now I see the world in systems.

STEVE MCGHEE: Can't unsee. That's right.

THEO KLEIN: And it's completely changed my worldview. So I would highly recommend learning about systems theory. At MIT, they've used STPA to model all sorts of social systems. And they've found flaws. Right? So it doesn't just apply to--

JEFFREY SNOVER: Who could have seen that coming?

THEO KLEIN: Believe it or not, believe it or not. So it doesn't just apply to technical systems. It can apply to any system in the world. Yeah.

STEVE MCGHEE: I feel like I understand this way better than I did before.

THEO KLEIN: Hey.

STEVE MCGHEE: Amazing. Success. Matt, do you have any further questions?

MATT SIEGLER: Yeah. Where would you like to see the bigger picture go-- so with the SRE book, we got people to use these practices in their daily work. They actually build tools, use these things in production. Presumably, there's a corollary set of principles used in design.

People will actually have templates, off-the-shelf practices that are analogous to what we're just talking about here, that they'll be doing them in design reviews. And it'll be second nature and won't be some set of experts that come and descend and then find seven really terrible things about the thing they just designed like you just described.

But it'll be just two engineers, A, engineer, B, product manager, go look at a thing and go, oh, yep, won't do that. What would you like to see happen that's just normal business as usual by learning some of these things?

JEFFREY SNOVER: Yeah, I think that you've got it exactly correct, that there will be practices and techniques and, hey, when do you apply it, et cetera. Now again, people can go look up STPA or STAMP at MIT, and you'll get really thick books. And they're great. They're fantastic.

But when you go to try and apply it, it's going to be kind of tough. I really would encourage you-- Google has produced a set of work that people can-- I would encourage you to start with that, honestly. We're leading the way, bringing these practices to services, bringing them to software, producing the content to do that. Theo has a number of-- Ben Treynor, again, inventor of SRE. And Tim Falzone, one of our STPA facilitators, wrote a great blog talking about STPA as the future of SRE.

And then, Theo, you want to point to the number of the other--

THEO KLEIN: Yeah. So I gave a talk at SREcon this past spring, where I dove into more detail about this road disruptions system. And I used the knife there as well, so that's a great primer. And at MIT, there is bi-yearly conference where you can learn more about STPA and STAMP, and those are free. And stay tuned because I think we'll be releasing more content about STPA to the community.

JEFFREY SNOVER: And the MIT conference, a number of the videos from those are online. And again, I encourage you to watch the ones from Google because-- the others are great as well. But we go into the details of, how do you roll STPA out to a community? How do you know when to go this far but not that far, things like that.

STEVE MCGHEE: OK, well, thanks very much, guys. Like I said, I feel like I actually understand it this time, which is fantastic because it's not the first time I've heard about it. And it's like, I feel like it's working. This is great.

MATT SIEGLER: It's going to stick.

STEVE MCGHEE: It's going to stick this time, finally.

JEFFREY SNOVER: Definitely, it's one of those--

STEVE MCGHEE: This is one.

JEFFREY SNOVER: --new things that you got to get your head through the knothole.

STEVE MCGHEE: It's right, totally.

JEFFREY SNOVER: And once you're through the knothole-- because that could be a difficult transition. But then on the other side, you're like, oh, wow. I wish I had known about this before.

STEVE MCGHEE: Well, where can our listeners hear more from you, both of you, and about this program? Do you have any final send-offs, like follow-up notes? Where should people keep going for this journey?

JEFFREY SNOVER: Yeah. OK, so I'm on Twitter at @jsnover. I'm also on Bluesky at jsnover.com.

THEO KLEIN: And I don't have a Twitter or a Bluesky, but maybe I should. But you can find me on LinkedIn, Pierre Theo Klein. I have a second first name, Pierre, spelled P-I-E-R-R-E Theo Klein, and that's primarily where you can find me, on LinkedIn.

STEVE MCGHEE: Cool.

THEO KLEIN: And then otherwise, for STPA, check out our blog posts, and check out MIT STAMP.

STEVE MCGHEE: Awesome. Thanks very much, guys. This was awesome.

MATT SIEGLER: Super duper.

JEFFREY SNOVER: Yeah, thanks for having us.

STEVE MCGHEE: Until next time. So long, folks. Adios.

—

[JAVI BELTRAN, "TELEBOT"]

JORDAN GREENBERG: You've been listening to Prodcast, Google's podcast on site reliability engineering. Visit us on the web at SRE dot Google, where you can find papers, workshops, videos, and more about SRE.

This season's host is Steve McGhee, with contributions from Jordan Greenberg and Florian Rathgeber. The podcast is produced by Paul Guglielmino, Sunny Hsiao, and Salim Virji. The Prodcast theme is Telebot, by Javi Beltran. Special thanks to MP English and Jenn Petoff.