[THEME MUSIC "TELEBOT"]

JORDAN GREENBERG: Welcome to season 7 of the Prodcast, Google's podcast about site reliability engineering and production software. This season, we are exploring what it takes to ship software and what that even means to different folks. Join us as we discuss how to ship it as SREs. And, of course, the most important part is the friends we made along the way. Happy listening, and may all your incidents be novel.
JAVI BELTRAN: (SINGING) You missed a page from Telebot

DAVID BLANK-EDELMAN: Well hello, everybody. I'm David Blank-Edelman. I'm your guest host for The Prodcast today. This is episode 2 around the new edition of The SRE Book. The former one we're calling 1E, as in first edition, and 2E is the current one coming out secondly. And so we found that there was so much to talk about that we wanted to have a second episode. And in this episode, we are going to refer back.
Because of scheduling, we have some slightly new cast of characters here, and some old, some new friends to talk to you. And the second edition that we're going to talk about is nigh and coming. And so if you haven't heard the first episode, we recommend it because we might be referencing things we said there. And it was a banger, and I know this one will be too. So let's get back into this. What I'd like to do is go look backwards briefly, because we now have a couple of new people.
I would love to ask a couple of questions that we asked about last time, just to warm us into this. And the first question I have is for either Chris or Jennifer, which is, do you want to say anything about the reaction to the first edition, how it was received, whether you found it showed up in places you didn't expect? Tell me a little bit about how the first edition's impact was felt.
JENNIFER PETOFF: I think when we undertook this project back in-- I think we started around 2014. Niall can verify. But basically, I like to say this was Niall's brainchild. He had the idea. He had this subject matter expertise, but didn't necessarily have the skill set to make it real. So he basically brought me in. My job was to herd the cats and make one year a reality. Turned out with just a great and impactful partnership.
And I think when we started the project, we didn't really have an idea of how it was going to land. Would maybe 100 people read it? Would 1,000 people read it? Would it be popular? Would it not be popular? What would the reaction be? And so I think there was some trepidation when we published the first edition. And I would say, in the end, we were absolutely delighted and floored by the reception that the book got.
If you look at it today, it's been cited-- 10 years later, it's been cited over 1,400 times, according to Google Scholar, across a huge array of disciplines. So I think it's pretty-- I personally feel really proud of what we accomplished with the first edition. And I can only hope that the second edition is as well received. Although, as we know, hope is not a strategy. And therefore, we did our best to ensure the right outcome with the second edition.
DAVID BLANK-EDELMAN: Chris, do you have thoughts on this?
CHRIS JONES: I think I would totally agree. I sort of traced the Genesis back to a magical conversation in a micro kitchen in Dublin around 2009 or 2010. I forget exactly what year it was. There were a bunch of us chatting at coffee time about what the things we would want if we needed to recreate SRE somewhere

else would be. And that, I think, percolated eventually over the years. Certainly, what we were talking about at that point was sort of a short list of ideas, maybe 80-paragraph sort of thing or whatever that eventually grew into the first edition.

And I was certainly blown away by the reception. I certainly didn't expect it to be as well received as it was. So fingers crossed that this one will be as influential.

DAVID BLANK-EDELMAN: Have you encountered it in any interesting places that you didn't expect?

CHRIS JONES: So I was walking in Powell's bookshop in Portland. I think it was 20-- it was either late 2016 or early 2017. I was visiting for unrelated reasons, and I was just in the bookstore. I was just wandering randomly down the aisles. I wasn't even looking for it and came across, I think, three copies there. And I was sorely tempted to deface them by photographing them as a surprise. But I decided, no, probably better not to.

I've also come across it in a public library, not in San Francisco, which seemed a little surprising, but there you have it.

DAVID BLANK-EDELMAN: So I want to pull us or-- so, Jennifer, your hand's up. Go for it, please.

JENNIFER PETOFF: So I wanted to build on what Chris was saying, if I might. So number one, in terms of physical locations that I've seen the book that I was really excited about, I was on vacation in Beijing in China, and I saw the Chinese language edition, which was super cool. And the other thing beyond physically where we've seen the book and found the book, I think it's also interesting to see where it's been cited. I talked a moment ago about the 1,000-plus citations.

And I think everyone on this call has worked with me before and knows I'm easily nerd sniped when it comes to data. So I did an AI assisted analysis of about 700 of those citations. And just let me give you a few examples of where it popped up that I was actually a bit surprised about. So number one, in the field of chemistry, which is my background originally, there's a paper called "IT Support for the Optimization of the Epoxidation of Unsaturated Compounds. In electronics, reliability tests and data analyses of solder joints.

In terms of water and cities, civil engineering, the internet of things architecture for urban water management. Delay tolerance space networking, so we're getting into outer space now. Construction, green infrastructure, opinion mining, and construction material reuse, optimization. Life science, high throughput neuroeconomics, large scale experiments on risk preferences in rats and mice. And finally, library science, reducing system toil in a university library. And that one I was excited about because they used our own language in their citation.

So anyway, those are just a few of the places that I was really surprised to see it beyond just the traditional computer science, systems engineering sort of spaces.

DAVID BLANK-EDELMAN: That is super cool. And I think I've seen the movie made out of one of those. I don't remember the product placement there. Anyway, well, so what I want to do is I want to connect that into the current time, because we had talked a lot amongst us last time about what has changed since the first edition came out, and how we moved into the second edition, what was there that wasn't in the first edition, and that foci.

One of the questions that I had that came up over our previous conversation is when we were talking about how do you actually write, especially in this time of AI, there was some conversation about there being fundamentals that were still true in the current day since the first edition was first published. And my ears perked up, but we didn't get a chance to get into it. I'm really kind of curious whether any of you folks

have some sense as to what are examples of those fundamentals that are still true in this day, that feels like it's in constant flux. At least that's my feeling of the tech world at the moment. David.
DAVID HUSKA: How down to bedrock do you want to get? The fundamental fundamentals of fundamentals I think have not changed. And we made that pretty clear in the book. We are still engineers. We are still focused on reliability for large complex systems. Hasn't changed. Everything else, the tools we use, the org chart, the error budget policies, all that's negotiable. AI is a good example of that. The automation, the tools we didn't have 10 years ago, very much on the table now. And that will continue to evolve, but those fundamentals haven't changed.
DAVID BLANK-EDELMAN: Cool. Betsy and then Chris.
BETSY BEYER: I would say that runs the gamut from some fundamental practices like postmortems and retrospectives. Those are always going to be really important. Now, they might be easier to automate now with the help of AI, but I think that will always be a best practice for our discipline and many others. And then as far as conceptual fundamentals go, as we were talking about in the first episode, that we got feedback that made it obvious we didn't touch on culture, the culture that supports SRE, that is something that always needs to be there in order to have a successful SRE organization. And we really do spend a lot more time on that in this edition.
DAVID BLANK-EDELMAN: Cool. Chris?
CHRIS JONES: So I think there are two aspects to that. One is the technical fundamentals, which I think David talked about a bit. The other are the sociotechnical aspects. So for example, incentive alignment-- if you don't have aligned incentives throughout your system, whether that's a system of people, or computers, or people and computers, the system is not going to produce what you want, similarly, the importance of the feedback loop around learning from when things go wrong, which Betsy sort of alluded to.
So I think all of those things, those haven't changed. We didn't talk about them explicitly very much in the first edition, partly because it's sort of like fish in water. It was just in the culture, and Google is not-- a book from Google about Google is not institutionally well placed to talk about the things that are institutionally different in the Google culture of that time, from other places, with feedback from outside after the release of that book, a mix of new contributors, mix of old contributors elsewhere.
That gave us a more-- and also just broader exposure throughout the industry to these ideas. That made it much more possible to spend time focusing on these things that are actually fundamental that we really didn't acknowledge explicitly because they weren't obvious to us at the time.
DAVID BLANK-EDELMAN: Cool. Niall, and then Jennifer and then Betsy. Oh, your hands down again. OK, cool. So, Niall, please.
NIALL RICHARD MURPHY: Yeah, sure. I think the way I'll put this, which picks out some of the details in what some of the people have said previously, is that the mindset remains constant in the sense that you should care about reliability. And in fact, in the book, we go into great detail about ways in which you can communicate the value of reliability, why you might care, and in particular, how to communicate your caringness to other people, which turns out to be a key point of being necessary in an organization.
But I also think that one of the things which is going on in the kind of wider industry right now, which I think informs some of the use of AI is that there's a certain set of people who are hoping that adopting AI in the SRE context will relieve them of the responsibility of caring. And I don't think that's quite true, and I'm not sure it'll be quite true ever, though, obviously, important question marks remain under that

heading. But I think the thing that remains fundamental at the basis of the book is to argue for the importance of reliability, which I think is kind of self-evident in a way that 1E, possibly inappropriately, represented as self-evident and 2E represents as, oh, actually, we have to argue for this. And I think what's happening to software engineering, what's happening to the rate of delivery of new lines of code into production, the ensuing instability, all of these, to my mind, continued to make good arguments for caring about reliability and doing something about it in a systematic way.

DAVID BLANK-EDELMAN: Cool. Jennifer?

JENNIFER PETOFF: And just the final point to make around the mindset considerations, I agree with Niall. It's the cultural components. It's the mindset that remains the same 10 years on. And specifically, I led learning programs for a very long time, like SRE education programs for a very long time. And that culture of learning and continuous improvement, I think is really foundational to SRE. It was in 2016. It is today. And I just wanted to give that a shout out to learning.

DAVID BLANK-EDELMAN: Cool. David? And then I'm going to step in and ask the next question that will get us even deeper into the abstract.

DAVID HUSKA: Sure. I'm just noticing, listening to all these responses, that the advice we've given on how to get started on SRE has not changed in 10 years. And this comes up a lot working with the first edition. As people new to it read this book and are staring a little bit into the headlights of how to get started, they'd ask, what can I do? I don't have Google's budget. I don't have the support system. And the same fundamental advice is there. And I think that speaks to all the things that everyone just spoke about. Find the things that your users care about and measure that. Refuse to punish people when bad systems break. Learn from it, and find your biggest time sinks and automate them. And in 10 years, none of that has changed. Not one bit. So those underlying fundamentals are still there in what we communicate to people when we want them to be successful with this. So I think that speaks a lot to how much hasn't changed.

DAVID BLANK-EDELMAN: Cool. So I have two fairly abstract questions. If you need to perhaps adopt The Thinker pose from Rodin for either of these, it'll be totally fine. Yeah, there you go. You're all looking very, very thoughtful. But before I get there, I'm going to do a quick palate cleanser. One of the questions that I asked David, Niall, and Betsy last time that I haven't asked Jennifer and Chris is, if you had to recommend one chapter from the second edition that you did not write, what would that chapter be? Go ahead, Jennifer.

JENNIFER PETOFF: I would say the value of reliability. And the reason for that is it really helps frame the SRE conversation with leadership in their language. It gives the "what's in it for me" information that helps make it possible to get that buy-in that is really so necessary for a successful SRE practice. Shout out to David for really doing a great job with that chapter.

CHRIS JONES: So I'm going to hedge my bets here and give one technical and one sociotechnical answer, the technical answer being testing, which is comprehensive-- truly, truly comprehensive. If you need a guide on how to get started in your testing practices in SRE, this chapter is for you. If you need a guide on how to get deeper into testing, this one's for you. And then on the sociotechnical side, I would go with "SRE at the Tipping Point," which is a fascinating look at our possible futures and how AI is likely to affect the way that the software development lifecycle feeds into what SRE does.

We're making no claims in that chapter that it is the future, but it is a probable set of futures. And I had a lot of fun while I was reading it, so I hope other people do too when they read it.

DAVID BLANK-EDELMAN: So let's get a little fun and abstract. I am really curious, what did we learn about reliability itself since the first edition that is represented in the second edition, or even in the process of writing the first edition? What did we learn about reliability itself? Go ahead, Niall.
NIALL RICHARD MURPHY: So my tactical response is missing control planes, which is this idea in STAMP, STPA, and a bunch of other stuff that if you have an outage which is related to, ultimately, not just your lack of monitoring or observability on a subcomponent of the system, but also your ability to affect it, you can enumerate additional ways to control additional elements of the system or subcomponents of the system in quite a systematic, relatively straightforward way. It's a new lens through which to approach the idea of analyzing a system or subcomponents of a system, and there's an entire chapter on that.
DAVID BLANK-EDELMAN: Cool. Anybody else have a-- oh, Betsy. Yeah, please.
BETSY BEYER: Yeah, I will answer this in terms of what we learned while writing the book, which we also learned while writing the first book, but maybe should have remembered it better, is to, A, do not have a single point of failure for any particular bit of knowledge or process, but the corollary to that is have a very clear decision maker from the outset. There were some chapters where we had a wealth of authors because these people are really experts in their field, but it can be very hard to achieve consensus among really smart, opinionated people.
DAVID BLANK-EDELMAN: That seems like a shocker, David?
DAVID HUSKA: In our last conversation, we talked a lot about what is new in the book, and I think a lot of that covers your answer in terms of what we learned. Niall was talking about the evolution of what we learned, and we would expect after 10 years that Google has learned a lot about reliability. And we have. Maybe some exceptions that are like the sociotechnical and psychological safety stuff, which we already knew. We just didn't write about so much in the first book.
The things in the second book that do speak to that evolution are things like control planes, which Niall spoke about. Platform engineering makes a big appearance in this book and is something we really learned we had to change. We just couldn't keep working with that the way we were 10 years ago. Probably the safety science as well, learning from incidents as well, we already were doing that. Now we've just upped our game a lot. So these are all evolutions, not revolutions in terms of what we're doing. So I think if you look to what's largely new in the book, it'll answer that question in terms of, I guess, evolution.
DAVID BLANK-EDELMAN: Jennifer and then Niall?
JENNIFER PETOFF: Sounds good. I was just thinking about this sort of philosophical question. What is reliability? How has it changed? What is the nature? And I think reliability is subjective. It depends on the goals of your product. It depends on the goals of your organization. I think also, when adopting reliability practices like SRE, organizational context matters. That's become abundantly clear over the last decade. And I also think in terms of how reliability-- what is reliability today versus 10 years ago, it's a key to high-performing software and high-performing teams. That is what the DORA research shows. It's half of that equation. So I think there's a lot more data to support the importance of reliability in whatever philosophical form that it takes.
NIALL RICHARD MURPHY: Yeah, just super quickly, I am not speaking corporately for Google incorporated, 1600 Amphitheater Parkway in this way. But I do want to say that to a certain extent, this was one of the criticisms we received at the time. The first book was talking about questions of reliability,

and systems management, and design, and so on, as if it was kind of ab initio produced by Google, envisaged entirely by Google. And there was certainly a lot of novel Google material in that book.
But I think one of the things that 2E does, which we should say out loud here, is it acknowledges a lot more of the debt, and the precedence, and so on, and so forth to what we did, and also does its best to promulgate that in 2E, obviously, with a Google spin and so on. But when I say STAMP, STPA, and test and all those kind of things are represented in this book, we talk about things that other people have done. And it's important for us to acknowledge that.
DAVID BLANK-EDELMAN: Cool. Chris, Jennifer, and Betsy in that order, please?
CHRIS JONES: Yeah, one of the evolutionary ideas here is particularly drawn out by the platforms engineering chapter, which is that you actually have to be conscious about how you are creating, and designing, and changing, and evolving an ecosystem of software that depends on these systems that you're running and that you're providing. And while I think that was perhaps something that was known implicitly at the time we did the first edition, it certainly wasn't something we talked about explicitly.
And the growth of software-- I don't really want to use platforms again. That's not quite what I mean, but the idea of having golden paths of pre-vetted pieces, modules that fit together to accomplish whatever outcome you want is, I think, something that's much more prevalent at Google internally now than it was 10 years ago. We have the rise of a whole bunch of internal frameworks that were very nascent at the time.
And then the other aspect of that is, at the time that we were writing the book, 2014 to 2016, the first edition, Google was still very early in the cloud. And as a result, the way that we have to think about and operate systems with parties external to Google who don't have monitoring or debugging capability inside in the same way that an internal team has changed.
And that means you also have to be very thoughtful about how do you expose those sorts of interfaces to the outside. And David can talk about that for days. But that's my thought on what's evolved from 2E from 1E in some sense.
DAVID BLANK-EDELMAN: Cool.
JENNIFER PETOFF: So I just wanted to link back to what Niall was saying about some of the feedback on the first edition and a little bit of the Google centricity, so to speak. I think one of the ways we address that in the second edition was having more voices from our colleagues in cloud customer engineering. So I think that's been great. We worked with a lot of people on the different chapters, and they're the ones that have this firsthand experience with customers implementing SRE, and people who are not Google implementing SRE, and what the pain points are.
So getting those real life examples across a variety of industries, I think, makes those getting-started conversations and the getting-started material in the second edition hopefully more accessible. Yeah, so really excited that we cast a wider net for contributors this time.
DAVID BLANK-EDELMAN: Cool. Betsy?
BETSY BEYER: And I will further riff on Jen. In response to Niall's response about looking outside of just Google itself, we really made a conscious effort this time to also feature case studies from other companies, because what has become very apparent in the past 10 years is, yeah, Google might have invented SRE, but we are by no means the only people doing it these days. And we're not out there claiming we're doing it the best out of anybody. Every implementation of SRE is very much tailored to its particular environment and needs.

So to that end, people might be a little surprised that we look to Microsoft for a case study. We also have Spotify, Etsy, Disney. Waymo's a more internal case study. But we did do this to a pretty good degree in site reliability workbook that followed up the first book. But here, we really felt that showcasing other companies should take a more prominent place in the SRE Bible, if you will.

DAVID BLANK-EDELMAN: Cool. So now I have my second philosophical question. And my fond hope is I can get you to argue amongst yourselves with this question. How, if at all, do you believe that toil has changed and/or will change in the current day and the land of AI since it was first in the multiple chapters, both in the book and in the workbook, and then perhaps now in the second edition? I'm really curious how, if at all, toil has changed. David?

DAVID HUSKA: Yeah, them's fighting words. I think what people call toil has changed. Toil never was just stuff you don't like to do or stuff that's backlogged, although it feels that way sometimes. And AI has created a lot of that. So I'm noticing people referring to what AI has generated and placed on people as toil. I don't think that part's true. There's no question that the cheap and easy availability of software programming bits and bytes has shifted a tremendous amount of effort onto people.

But I think that collision is exactly what Niall mentioned earlier about people caring. Are we going to automate ourselves out of this completely? Zero chance for exactly the reason that we're seeing, that someone still has to care. Someone still has to be accountable and be responsible for what, ultimately, that software is doing. So yes, what toil looks like has changed, no question. It has shifted onto humans. We're no longer really automating keystrokes the way we were 10 years ago.

The fact that we're still having this conversation tells me it's not going anywhere 10 years on. Google has automated full swaths of the SRE profession out of existence. Manual machine provisioning, all that kind of stuff is gone. It's just been replaced, working as intended, with new things that have come in to replace that. And the practice of SRE has grown, expanded, not shrunk. So that tells me that we're never going to run out of problems that require that human judgment.

And as cheap automation continues to expand and make that software more available, the chaos that exists at that boundary of where it meets the road just grows, and grows, and grows. And that's the part that actually worries me. That's a separate conversation, but that's the part that worries me the most. And it's exactly because people are burdened, possibly burdened with too much stuff and have lost that context. They've lost that muscle memory from working with it, that it does become a reliability issue. And so definitely would not call it toil in the traditional sense, but it is absolutely a reliability risk and probably a separate conversation.

DAVID BLANK-EDELMAN: So I'm going to hope that somebody is going to argue that toil hasn't changed. And I'll give you extra points if you cite Lisanne Bainbridge in your answer. Chris?

CHRIS JONES: So I'm not necessarily going to cite Bainbridge, but I might cite the law of requisite variety. So one of the things that is perhaps Pollyanna-ish is that the possible future where AI is able to take on increased levels of judgment and handle all of the low-hanging fruit that we had in the form of toil, and suddenly, we no longer need to worry about, oh, can I approve this increase in resource quota for this particular user in this location? Because we can have something make an appropriate judgment. That would be great.

The problem is that as the software that we write becomes more complex and becomes delivered faster due to code-generating machines, the control surface for the number of possible states that that software can will also increase, which in turn means that the complexity required for the thing that manages that

has to increase. And if there's an analogy here, it's to the famous quote about writing software that's so simple because I can't it's harder to debug it, and I'm only so smart. That will also show up with AI, I expect. Now, the downside case of that is, well, if it becomes too complex for humans to debug, then we're really in trouble.

So I think toil will-- I don't see toil going away. I think that's just like housecleaning. No matter how much you automate your dishwasher, you still occasionally have to clean it. But what will happen is that it will change over time, and I think we've already seen that.

As David mentioned, we don't have to worry about individual machines anymore most of the time. But I worry about the case where the usual level of abstraction that people work on is so high that when there is a breach of the abstraction layers, and when the prerequisites to actually having a successful abstraction fail, that it will become difficult, A, to recognize that case, and B, to actually do anything about it. So that's the, I suppose, doom and gloom version of my previous Pollyanna-ish pitch.

So I don't know if this is the fighting you were looking for, David, but at least let's go at it.

DAVID BLANK-EDELMAN: Oh, I'm up for it. I like smart answers. Niall, you seem to get so excited about the topic that you pushed the red button a moment ago. Now that you returned, I'm curious what your answers to this question.

NIALL RICHARD MURPHY: Well, I was just going to say that software isn't very reliable, and that my web browser disappeared in the middle of the recording. So somebody should definitely get on that. I suspect Chris has made the point that I was going to make about meta toil. So instead, I will make a slightly different point, which is to say that, though competing definitions of toil exist, the one that I like is activities that you have to do in order to make the system continue to behave as it was going to behave as it should behave, which is to say, activities like in the old days when you used to do the failover between data centers or equivalent.

You would be performing actions in order to help the software continue to exist and serve users and so on, so forth. But it had no enduring value beyond that. And now in a universe where we click our fingers and get a script to do all of that, yes, we get meta toil with respect to the number of scripts and maintaining those kinds of things. But I think also, we will get activity, which is performed in order to help the software continue to exist, but is performed by the human working with the AI in order to do that.

But the proportion of it, I think, will go down, or possibly more accurately, would go down if the number of systems were held constant, and arguably, the number of lines of code within the system were held constant. But I don't think that's the world we're heading towards. We're heading towards the world with a violent proliferation of software. And I suspect that the "AI got us into this, AI will get us out of this" situation will continue, which is to say congratulations. You're now on call for the 10 additional ad serving systems that we created since yesterday. Good luck.

And these systems will require some management, and that management will be probably called toil, even if what it ends up being is a lot of English typed into a prompt. So I see a bifurcation where the bleeding edge and some proportion of the rest of the industry moves forward with proliferation of software, and a much smaller subset of the industry maybe does not do that. Yeah.

DAVID BLANK-EDELMAN: So, David, why don't you have the last word on this topic? And then I'll ask my closing question.

DAVID HUSKA: Sure, I know you're looking for a fight, and it sounds like we kind of mostly converged on more or less the same less optimistic future. Just to clarify that, because you did ask specifically about

Bainbridge, and it seems like we all described it in more or less the same way. I'm looking at, basically, so Jevons paradox. I'm paraphrasing here, but things get used more as they become less expensive.
So we know this is trajectory is going to continue up and to the right. At the same time, you have Bainbridge, who pointed out, what, 50-ish years ago that as things become more automatable and we don't really understand how they work, we then run-- we get bit by the just not understanding. We don't have the full context in our head. Again, this is paraphrasing.
These two are a perfect storm for SRE. And that's probably what the gloom is that everyone is feeling. As you have all this software thrown over the wall into production that really no one understands anymore the context of, where it's provenance or it's quality, you now have those ironies of automation from Bainbridge, having agents handle the 99% of small little outages that it can handle with a playbook
So it's very likely that we start to see-- even if we are still using SLO budgets and error budgets in five years, we see those small little outages start to drop to 0. But then the biggest loss of error budget being these big novel outages, which I think touches on something you brought up last time, David. I don't think we have time to get into it.
But what do you do when things are non-deterministic? And we started to talk about making them reliable in terms of small, bit-sized, deterministic things that we can roll back and get out of, we can get ourselves away from and roll back to a known good state. But that just gets harder, and harder and harder as the automation makes it easier, and easier and easier to get away from it. And the short-term financial gains of doing that are attractive until we get bit hard.
So maybe everything old is new again. These are the same things with reliability as Google and others started to grow really big. And then we realize, oh, beyond a certain point, we have to put in engineering fundamentals. Maybe we just have to go through another big growth spurt to get bit by that again, and then realize we need to do this with a whole new set of tools.
DAVID BLANK-EDELMAN: So I'm quite the softy, which is why I'm going to allow you, Chris, to have the last word on this particular topic.
CHRIS JONES: Thank you. And just to give an even doomier version of David's description there, or prognosis there, that's if the amount of software that the AIs are writing increases. But what if the complexity of the software that the AIs are writing is also increasing, such that the complexity of the outages in that software is also increasing exponentially?
DAVID BLANK-EDELMAN: So I am trying to decide between asking my last question and crouching under my desk after that answer. So I think I'm going to go with the last question, because I do have a role here to play, and I'm going to make it really hard for everybody. I appreciated Jennifer's citing of the "hope is not a strategy." And so to further that, not as a strategy, but to just pick up on that, I would love to hear a one to 1.75-sentence-long statement of what you hope for the people who are going to read the second edition.
CHRIS JONES: All right. So what I hope for the people who read the second edition is that you will find it a hopefully useful, and at least vaguely interesting guide how to solve your challenges in operating, and building, maintaining software systems at scale, and ideally, that it will give you some guideposts for what the future might look like and how you might prepare for it. Even if you're not currently at Google's scale, hopefully there are some lessons here that will be applicable to you in the future, whether that is the near future or the further future.

I think, as David was alluding to, software is just going to continue to grow, and the volume is not going to go down. So it may become useful to you sooner than you expect. I'll leave it to the readers and the audience to decide whether that's cheery or gloomy.
DAVID BLANK-EDELMAN: I forgot to leave out the run-on constraint, so let's allow you to continue to use that as an out. Betsy?
BETSY BEYER: I would say, and I'm thinking of people who are already SRE practitioners or advocates reading this book, just a fundamental understanding of why reliability is important and how to convince the people you need to that it is.
DAVID BLANK-EDELMAN: I did not track in my head which one of the top three people on The Brady Bunch set of pictures I see went first. Would one of you be honest and say that you raised your hand first and start?
NIALL RICHARD MURPHY: There's a lot of wisdom in that book, like codified wisdom. But in particular, the thing I'd call out is that there's a lot of new ideas. And some of these new ideas are super tactical, and ideas in observability, and testing, and so on and so forth. But the thing I want to encourage readers to do is to take away from the book the idea that old assumptions now require reexamination. And it is an excuse to rethink, as always, from first principles directly in your environment.
DAVID BLANK-EDELMAN: Fabulous. David and Jennifer, rock paper scissors. Or one of you can go first.
JENNIFER PETOFF: I can go first, and then David can have the last word.
DAVID BLANK-EDELMAN: Actually, I'll be the David that has the last word, but OK.
JENNIFER PETOFF: So my point that I wanted to raise was just, I think with the second edition, it's important to still remember that there is no-- there's still no easy button when it comes to doing SRE, being SRE. It's not a matter of changing the name on the door and, yay, it's all going to happen. I think it's important to recognize the importance of those sociotechnical aspects, the culture, investing in your people, blamelessness, the whole nine yards.
But I think people are hard. Organizations are hard. The tools are easy. And especially now in the generative AI era, the tools are, you magic up something every other second. So there is no easy button, but stick with it because the rewards are worth it.
DAVID BLANK-EDELMAN: Awesome. Mr. Huska, what do you hope?
DAVID HUSKA: Well, that's real tough to follow. I think everyone nailed the sentiment that I feel, Niall and Jen in particular. The first edition talked about the genesis of SRE, which was engineering applied to operations. That is still very much true. Everything we've talked about, the problems at the technical level are still engineering problems, and they will still yield to engineering solutions. And for exactly that reason, I think Niall is right. Think about these again from first principles like an engineer.
And 10 years from now, if there's a third edition, we will be speaking about those solutions that were derived from applying engineering to operations. And the second bit is the fact that, yes, computers are hard, but also the human element of that probably has become more important with this advent of AI than it was ever before, because that's not getting automated. That's not going to be made more cheap, and that is going to place more stress on organizations, which are made out of humans, obviously.
So understanding those components of it and making sure those are really working are the two things that I would have people take away from this, the human element and the fact that this is still engineering applied to operations.

DAVID BLANK-EDELMAN: It's a fabulous place to end. I just want to thank all of you for joining me and all the folks who joined us in the first episode of this. I really appreciate it. I want to thank all the folks listening to The Prodcast. I hope you'll continue to subscribe. Click like and all the other things you're supposed to do, not to a podcast. And thank you all for listening. Go enjoy the second edition as much as I have. Take care now.

SPEAKER 1: You've been listening to The Prodcast, Google's podcast on site reliability, engineering and production software. 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 hosts Jordan Greenberg, Steve McGhee, and Matt Siegler. The Prodcast is produced by Paul Guglielmino and Salim Virji, with contributions from many SREs behind the scenes. The Prodcast theme is "Telebot" by Javi Beltran and Jordan Greenberg.

[ROBOTIC VOICE]