[MUSIC PLAYING]

SPEAKER: Welcome to season five 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.
(ELECTRONIC VOICE) [INAUDIBLE] Telebot.
MATT SIEGLER: Hey, everyone. Welcome back to *The Prodcast,* Google's podcast on SRE in production software. I'm Matt Siegler, and today we have a lot of folks with us. Let's start with my co-host, Steve. Who are you?
STEVE MCGHEE: I'm Steve still. I still remain Steve. How's it going, Matt?
MATT SIEGLER: That's good. Thank you. Great. And also, today we have two guests. Introduce yourselves, please.
FELIPE: Hi. Yes, my name is Felipe. I am a tech lead in the Gemini safety team. And I'm a jack of all trades, doing everything I can to help everyone, everywhere, all at once.
STEVE MCGHEE: Nice. I like that.
PARKER: And I'm Parker. I'm a product manager working with Felipe on model level safety. So we work on the Gemini family of models, but also on some of our generative media models, like Veo, and Nano Banana, or otherwise known as Gemini Flash Image Generation Version 2.0, or whatever it's called.
STEVE MCGHEE: Let's stick with the banana one. That's a much better name.
PARKER: Yeah, exactly.
MATT SIEGLER: Dang, that's a cool name.
PARKER: And we work on basically ensuring that our models output content that is hopefully helpful to people and safe. And we collaborate with a lot of teams across Google to bring the models to users in all kinds of different product experiences.
MATT SIEGLER: All right, let's get right out of the gates some terminology, rather. What do you mean by safe? I think I know what that word means, but let's have our audience some clarity. Exactly--
STEVE MCGHEE: It does not cause fires. Is that it?
MATT SIEGLER: Yep, that's right.
PARKER: That's one definition.
MATT SIEGLER: Easy.
PARKER: Yeah, I can take that. So safe breaks down in a bunch of different ways for us. Probably the simplest to start with is we do have product policies that have been defined by our trust and safety teams about all kinds of content that we do not want our models to generate. These are things like the worst of the worst, which could be even illegal content around child safety type of things, also dangerous content, so instructions for how to build bombs and do things like that. But also, depending on the modality, it can also be things like nudity or other kinds of image or videos that we--
STEVE MCGHEE: Like diagrams on how to build bombs, that kind of stuff too?
PARKER: These days, we're getting more into that territory, where the models now have a lot of more world knowledge than they once did.
STEVE MCGHEE: Crazy.
PARKER: But then also, it has to do with other dimensions of safety, which we call-- so one category of safety that is very important, even under certain regulations in Europe, are what we call frontier safety. And this has to do with the ways in which models could be used to harm

people in a more fundamental way. That could be around cybersecurity risks, but also things like CBRN, which is a long acronym for chemical, biological, radiological, and nuclear weapons, and related issues. So we have a whole framework around that, which is our internal policy that we also have shared with the world called the frontier safety framework that specs out that area of safety.
FELIPE: Yeah, I feel like safety in this new world has been quite changed from the place I was before. So I spent a long time in search ranking. And at that point it was much easier to define the boundary of what is working on safety versus what is working on ranking and quality. And it was somewhat deterministic, and the lines were well defined. You know what you want to show, what you can show, what you cannot show. And the interaction between the user and the machine and your product, it was well defined. But now I feel like with those models, the interaction is open-ended and unbounded. So then the safety definition becomes very squishy and a continuum.
So what do you call safety? What do you call quality of the model? Is a model that is sycophantic, is it a safety issue? Is it a quality issue? And I think right now-- that's why I said I'm a jack of all trades because I'm not working only on safety. When you work on safety, you work on everything. So it touches so many aspects of the model that it's hard for you to define with few words, this is what safety is. And it's also not something that you know when you see. It's not like that. You really have to have some experience and have other people show you. It's not something you just get it. It's quite squishy in some ways.
PARKER: It's a polarizing topic too. I'll just add, Steve, that a lot of people, some of the loudest voices on the internet, would prefer models that are maximally free to use, meaning they don't quote unquote "censor," or they follow instructions to the T. So there's a tension there between--
STEVE MCGHEE: Yeah, so I can see how this very quickly gets away from a research program, where we have-- how do we even stop it from doing x or y? And it gets pretty quickly into product decisions, which is, I'm guessing, why Parker's here. And then my other question about how this works in terms of safety for consumers is there's billions of them. And we're in production, and it's no longer research. It's out in the world. We've moved from research into production. So just to remind ourselves, this is an SRE-adjacent podcast.
And so this is, I think, where it gets real funny or really tricky is-- yeah, so in the research world, in the lab we can say we can hamper down certain types of responses. But how does it change when it gets into production? How do we know it's actually working? How do we know that the quality is actually at whatever bar it is that we want, whether it's safety or other measures of quality, for the end users every day and next week and the week after that and the week after that? What's the work that goes into making sure that the production user-facing system actually continues to hit these-- once we've drawn the line for safety, how do we know we're actually hitting it over time? That seems really, really hard to me.
FELIPE: Yeah, I feel like this is the classic changing the wheel while the bus is running because you really cannot predict all possible failure modes because of this factor I said before, where everything is-- so much of it is subjective and squishy, where the lines have to be drawn as you see new things happening. So you might decide that the model is not behaving the way that you wanted, and you decide that this is unaligned with what you want. And then you have to fix the model on the fly. And can you then predict and check for all possible dimensions and all possibilities of how the model will fail beforehand?
We try, right? We try to have as much benchmarks and as much evals as we can, but there's always this constant watching after the fact as well if the model is behaving how we want. And that's where I think the research goes because it puts the researchers right at the line of duty where you have to be there at the frontiers, watching the model behave and fixing the model as

it goes as fast as possible; as opposed to the way that research used to be, where you have plenty of time to test things out and then run experiments on your back end and then wait.
And then a week later, you got some interesting paper out of it. But this brings also this problem of the loop. As you see new things happening, you try to fix them. And then something else happened because you try to fix, and so on and so forth. So being able to do all of this, the entire loop very, very quickly, it's part of my job. It's part of what me and Parker does is designing and putting in place the systems that enables us to react so quickly and fix things on the fly like this.
PARKER: I'll just add a little bit. To some extent, there's all of the classic approaches to trying to keep a pulse on how users are experiencing the product. So there's in-product user feedback. So people can do the thumbs down thing and give us feedback via comments. Obviously, we also keep a keen eye on social media, and what people are sharing about the models, and their experiences with products that use the models. But those are really just very partial glimpse into what people are actually experiencing. And what Felipe is saying is absolutely right.
Think about the Gemini app or a ChatGPT-type of experience, where people are doing all sorts of different things. It's a very general purpose tool. And they can have very, very long conversations, which even if you were to look into the logs and try to understand them, there's a lot of additional context there across multiple sessions with the user. So it's a really tricky problem of how to monitor these systems effectively, even for known risks, much less new risks that might be coming up that we're not as aware of out of the box.
I do want to say just plus one to what Felipe was saying. Before we launch a new model, we do as much offline testing as we can, including what we affectionately call ART, so Automated Red Teaming, where you're looking in a more exploratory way at all the different ways in which the model might react to different kinds of inputs, some of it adversarial, intentionally trying to jailbreak the model in different ways, others just more benign, just trying to mimic real good-faith users, and just seeing how the model responds because a lot of this is just, like Felipe said, the fact that the output space is infinite here and input space as well. How do you explore that in a way that hopefully gets to the most egregious stuff before you even put these things out? So there's a lot of work that we've tried to do in that.
MATT SIEGLER: Well, hope is not a strategy, I think, we keep saying here. And I want to go a little bit back to something Steve called out, which is-- and then you guys have echoed, which is researchers and SREs are getting ever closer to working in concert in ways here that they've never before, which is, to me, a very personally exciting thing. And I think you and I, Parker, have personally worked together in this for many years. And I think this is now becoming fruitful. And I'd like to ask you to reflect upon what is novel tooling that you can speak about that's happening, that is noteworthy, that is not just shopping carts don't work.
That is happening here. We have something going on where we're deploying a suite of tools that are being used by people. And they're used by people, and we want them to work well. And we need feedback loops for when they misbehave or do things we don't want them to do that we can feedback and go, alert and do, nope, we don't like the quality we're showing to these people. We'd like to make quick changes. We have something go red, and then someone wakes up and goes, oh, yep, we need to do something.
Or something automatically happens, and we make a quick intervention. And we quickly change something rather than going back to a workbench on a researcher's desk. And they come in on a Monday and go, oh yeah, we need to quickly tune in and deploy a new version. So something new is happening. Talk about how that something new is actually doing something faster than over iterations of weeks or months or quarters or cycles of journals.
FELIPE: Yeah, I think we can speak about the whole spectrum of fixing a model on the fly like

this. And sometimes it's not on the fly because it's a model. It's really hard to train. It takes a long time to train. Even after it's fully trained, you can do certain kinds of tail patch post training and get the model fixed in that particular issue that you have. Even just the tail patch post training, which is a very small iteration on top of the post training that already happened, it still takes like weeks, right?
So it's not something-- not because of the time of the training, but just the time of designing the steps you have to do for training-- creating data sets, and then running the GPUs, and then deploying the model. Even deploying the model, releasing the binary-- it takes several hours or days, depending on which model. So it's not feasible to do that very quickly, but that's one side of the spectrum. And on the other side is a super dumb strategy where you just match the exact query of the user.
So if the user has found a jailbreak and you just match the exact jailbreak, the classic then kind of jailbreak. And you can do that. That's very brittle, kind of dumb. It will work only for that exact phrasing and that exact-- but it's very quick. It's very quick for us to deploy and just block it. But what it goes in the middle is where it gets interesting. And that's where a lot of the innovation has been happening in the past couple of years. And those tools that we have right now to try to fix the model on the fly without having to do the full retraining and to only train on very specific slices of the model, that's where most of the innovation happens.
Of course, I'm not going to be able to speak of all the details because a lot of it is IP from GDM. But there is very interesting things that we've been developing so far that can be done very surgically-- changes to either knowledge of the model or understanding of larger issues that the model has. And the model can self-monitor. So the model will know that this is-- I now understand, I've been told that this is dangerous, and I have to steer away. Or I have to behave in a different way for this particular kind of query, or for this particular topic, or this particular response that I generate.
And you can do that in a shorter time frame. It's not like weeks. But it's also not a couple of minutes, like you do in a regular expression, like matching the query. It takes a couple of hours for us to do this type of training, and deploy takes another couple of hours. But it's at least a middle ground where you can react quickly and then change that type of behavior. And it's also a little bit more, way more flexible and non-brittle, where the model has the understanding of its own thoughts and can react for different types of inputs. It's not just matching a single input.
STEVE MCGHEE: It kind of reminds me-- tell me if I'm getting this wrong, but it reminds me of early DOS protection. It was like, let's just look for this exact string. And that worked for some things. And then later we developed these better algorithms where it was a lot more about more subtle signals and looking at broader concepts. It's not the same actual system, obviously, but we started with the very direct things, and then we got to that chewy middle. And that's where the real fun is.
PARKER: I'll just add two minor things on top of what Felipe said. But really, practically speaking, for any company that is deploying these models and wants to have an initial toolkit of what to do about safety, I think you need to think about system instructions or developer instructions, which is basically instructions that you give the model about what your definition of safe means. And minimally, that will help the model understand this is one of-- in addition to giving the model a persona and explaining what is the model actually doing for the user in whatever context it's being deployed.
You may have some safety guidelines, which is to say, look at the query, think about these things. And if it goes against these rules, either respond in this way or don't respond at all, however you want them to do that. Think about what those should be, and those can be updated at any time really cheaply, right? Now, of course, not every model follows every

instruction to the T so it's not 100% guaranteed as an enforcement technique.
The other thing is just classic filters, things that you can either-- a lot of companies obviously already have content moderation filters that they can repurpose for Gen AI products. The other approach is just to take other LLMs and use them as a prompted classifier, which is very, very common these days, which is just to say, ask Gemini or your model of choice, is this piece of content that I'm showing you right now, does it violate one of these rules?
If so, we will not show it to the user, that kind of thing. Those are two very, very practical, and frankly, easy things to do that can be updated more or less on the fly, adjusting what the filters can do. And that's leveraging LLMs to do the work. Now, of course, second order and third-order questions about can these things be jailbroken? Yes, these are all risks. And this is where we're dealing with, it's going to be an arms race, effectively, trying to get to better and better protections.
STEVE MCGHEE: So while your teams are doing things like this, where you're setting up defensive structures and filtering and retraining, and then also having other LLMs come in and, hey, was that good in this way, in this particular definition of good before it makes its way out? So it feels like a multi-layered defense scenario, which is great. This tends to work when we have not just a hard shell but many layers of things when it comes to defensive systems. I have a question about that.
If you're all doing all that, and that's great, should users of these systems also be adding more layers? Should they be, on top of that, saying, yo, LLM, I want to build this app which does this creative thing and then comes up with something else? By the way, layer, layer, layer. So my question is if an SRE out in the world who's listening to this wants to use an LLM for something, and they don't want to delete the root file system of a production cluster, for example, I'm not sure that's going to be in your training data. Hey, remember, never suggest that we delete the root file system off of a production system.
What's some advice to someone who's out there who is trying to use an LLM to generate code or systems or something on their system? It's a weird circular question, I guess. But I think this is on the topic of a lot of SREs' minds these days of I want to generate a bunch of code to help me do my job, but oh, god, it's a little scary. What do we do to make sure the system itself is-- the LLM itself is safe, but also the thing that it's guarding or building or fixing, like AcmeCorpService.com, that thing stays safe also? Is that something that you guys think about or that you give advice about?
FELIPE: Yeah, I think every once in a while I speak to the folks in cloud that are dealing with the clients, and this touches the problem of alignment. Alignment is not a constant. It is different between person to person, between company to company, between industries, governments, and so on. So if you have a different client, if your client is, for example, a hospital and you're doing certain kinds of queries that are deep questions about medicine and how to do certain kinds of procedures that you might not want to show that to a regular user or an underage user you don't know, but for that client you should.
So you can have different kinds of policies. And some other clients might have even higher restrictions. If you're a game developer and you want to show this to underage users, then you have to have different guidelines there. So far, the types of systems that are available for the clients, they're still maturing. The kinds of systems are the ones that Parker already enumerated, which are somewhat either brittle or not 100% reliable, which is you write a system instruction and just pray for the model, please, please, please, don't do something stupid here, which works most of the time, not 100% of the time. And that's the problem.
So as things mature, I feel like we will see more and more those other kinds of systems where even the clients have access to and do a sliver of training on the model just to show this is the

stuff that you need to pay attention to. And if you are deep into the weeds there, if you're good as a developer and you have AI engineers working for your company there, you can get that done by using open source models and then doing some sort of post training on top of it, or using LoRAs or other techniques.

But it gets complicated and hard to maintain very quickly. Every time, for a new model, you're going to have to do the whole pipeline again. So maintenance and long-term updateability is a big concern. And I think that with those other kinds of systems that you're training the model, it gets expensive and high maintenance very quickly. So that's why I feel like it's still in its infancy. I think there is a lot for us to do there, where improving long-term maintainability and cost of maintenance there.

PARKER: I'm going to add a couple of things. One thing is for SREs, because I think this community is amazing at this, is thinking about all the ways things might go wrong and designing smoke tests, at both the model level and at the product end-to-end level, of even if few queries that are really targeted at things that if they go wrong, we're in bad place. And run those every time. You know what I mean? Have your own-- these things are often affectionately called vibe tests. People are checking for all kinds of vibes-- security vibes, safety vibes, quality vibes, good feels, humor, all these things. And I think SRE just has people who are just amazing at imagining all of the different ways in which this could go sideways.

The other thing that I think is really important is I think SREs need to have visibility into the, for lack of a better term, all the traces. What is actually being shown to the model? Because a lot of the times people don't realize how complicated that gets in a production system, that we're layering on lots of different instructions, lots of different contexts being pulled in from all these different data sources, which could confuse and distract the model, lead it to weird edge cases. And so having that observability, which I think is, again, a very deep SRE principle of can we actually see from the model's point of view what it is being instructed to do and help potentially root cause issues that might be caused by this weird composition of the context window, which is essentially, that is the model's entire world. It knows nothing that is not in that context window because these are not continuously trained systems.

So I think having that observability, having the SREs be a participant in thinking through what is a rational context to give the model for the purposes and to avoid a situation where the model is being asked to do two opposite things, which I think, in certain situations, that happens unintentionally, where you have multiple people contributing parts of the instructions or the SIs. Yeah.

STEVE MCGHEE: Can I throw out a phrase I learned the other day, which is drift detection? I think I understand this a little bit, maybe. I don't know. But basically, we think it's doing the right thing for a while and then later we're like, oh, god, it's not doing the thing anymore. What's happening? And this isn't necessarily about safety. Sometimes it's just alignment with your company's economic desires or something like that. Or just like, are we actually helping the customer with [? foo? ?] Is this a term that people should be aware of, and do they need to be defensive about it? Or do they just wait for the foundational models to become perfect? What do you think?

PARKER: Drift detection, that concept, MLOps-- I'm laughing because Matt and I were talking about this like 10 years ago.

MATT SIEGLER: Yeah, it's not new.

PARKER: It's not a new concept, but no, no, no, no, but in a good way in the sense that, yes, you should be monitoring as much as you can. It's challenging to do so. But for example, if you have filters in place, you should be monitoring the rate of those filters. And you should probably be sampling some of the examples. And looking at-- the filter is just a classifier, so just do your

confusion matrix.
What is the false positive rate on this classifier? What is the true positive rate? How can we continuously improve that part of the system? And that's one fuzzy drift detection mechanism, Steve, right? If your filter rate all of a sudden pops from 5% to 10%, or whatever it is, probably something has changed. So those kinds of principles absolutely apply.
MATT SIEGLER: Thank you for calling out the character traits that we think are particularly well suited to being scrutinous of quality. It's only recently that we now have something rapidly being used in production by billions of people at scale, which is suitable for this work. Until recently, it was novelty. It didn't have outcomes that were valuable enough to bother pointing a series at. But here we are. And I think [? ML, ?] it's about to have its kind of golden era. But I'd like to see the transferable skills actually transfer, but we're just inventing something new, and it's ill-defined.
But we just called a second ago about what is DevOps when ML code from the brain of the organism is generating? And you're like, well, how are we going to review this stuff? And how are we going to launch this stuff? And how are we going to iterate on this stuff? And then how are we going to review its proposed propositions? And it's weird and wild and new. But I do think the same principles apply, which is alignment with the objectives of our users, having really strong values.
I don't think-- that's not new at all. That's the stuff that we've been doing for 20 years, so awesome. But I think a lot of our audience is curious about how they're going to do what they're doing right now, which is keeping shopping carts working to doing what you just described. So do you have any ideas how they can even get going? They're out there doing that. They want to be doing what you're talking about, and they've got to get there from here. Thoughts?
FELIPE: Yeah, I there's what's happening right now in the present and what I think is going to happen in the future. I was just reading this blog post from LessWrong recently, and they were talking about how improvements on everything, including AI, usually happens in step-by-step mode in somewhat, not linear, but somewhat incrementally over the time. But the perceived quality, the perceived usefulness is a step function, where suddenly becomes useful. It's because when you get to that threshold of I cannot trust my AI to write the code for me because I have to check every line of my AI agent.
That's because there is a 20% chance that it's going to use a horrible bug in one of the lines, so I have to check every line. So that's not useful. So I'm going to have to spend as much time checking as I would have spent writing the code myself. But then suddenly, if that becomes a much lower probability that a horrible bug is going to be introduced in any given line, then I'm going to maybe not check every line anymore. And it's going to become useful.
So that step happens all of a sudden, while the AI has been improving over time. And in the present, we're still in the middle of this. I think we're close to that step function, but we're still in the middle of this, where I have to check the code, right? And the problem that is happening right now is that, well, it is really easy to write a lot of code, way more code than I ever wrote my whole life.
PARKER: Yep.
FELIPE: And that's true for all of the engineers, and it's true for people who are non-engineers, and it's true for people that never touched an IDE before. So what's happening right now in the present is we have a lot more code being produced, and still the same amount of people who are experienced as code reviewers. So what this means is that we are probably introducing a lot more bugs everywhere in all of our systems if you're not reviewing them.
So what do you do? You just hire a bunch of code reviewers, more than the [? suites ?] that you're firing because the AI is replacing them? I don't think that's necessarily what should

happen, right? But I do agree with that point on the blog post. I don't agree with the whole post, but I agree in that point of I think in the future we will cross the line where it becomes reliable to the point that you will not necessarily have to review all of the code.
But meanwhile, it should work like a factory, where every step of the factory, you have to have a quality control. You don't have a quality control only on the end. You do at every step that is important. And this could be code reviews. This could be monitoring. This could be automated monitoring. This could be automated monitoring on your network to see if there's a new unforeseen activity that shouldn't be there. And that might mean that a new bug was introduced or something. So I think right now at the present, we still need this type of checks that you guys were mentioning and Parker was mentioning before as well.
PARKER: I mean it. The only thing I can say is I always love working with SREs because I think the pressure to ship fast is everywhere right now, for all the reasons Felipe just said too. It's easier than ever to just throw together a new product. Look at how quickly OpenAI built, which was it? They were like six weeks to build this massive new feature and shipped it. And with that comes an enormous amount of pressure on the teams who are actually responsible for maintaining it. And so I think, to some extent SREs' time to shine is here, both in terms of just keeping these new products up, but also managing these risks and not burying our heads in the sand until something explodes.
So I feel like there's also the question of how do you use these models as part of SRE to do your jobs better or more easily, or faster, or at more scale. I have no special knowledge about that one. But I think every single person right now is having to think through that question of how do I use these models? How much can I trust them to be a part of my core toolkit? And so I think it's a pretty exciting time to be doing this work.
MATT SIEGLER: Wow.
FELIPE: I also feel like we are the wrong kind of person to ask that kind of question because we're always so-- or maybe we are the right people because we're always thinking about what can possibly go wrong.
STEVE MCGHEE: That's true.
FELIPE: I think the models are doing stupid things right now. We have to fix them, and they're not reliable. But if you ask other people, they're going to say, well, have you seen what happened the last two years? The models are really reliable right now if you compare to two years ago. Yeah, OK. But there's still room for improvement.
MATT SIEGLER: Well, great. We're just about running out of time. I'd like to ask an open-ended question we like to do with our guests at the end of the episode. What's something that concerns you about where we're going right now, big picture, either culturally within or within the industry, that we're not keeping our eyes on the ball about, that we need to really care about the value to make sure we work hard to-- but that we are using our forces to keep from doing so, that we do?
STEVE MCGHEE: What's in the shadows? What's the scary part right now?
FELIPE: Yeah.
[LAUGHTER]

PARKER: I think that the speed makes it very challenging to exercise foresight across the industry about where this is all going. And that is the thing that I think keeps me up at night is, how do we deal with the problems we have today and that have been created by things that have been shipped yesterday, while also preferably skating to where the puck is likely to be and where we can probably tell it's going to be in six months? But there are not enough hours in the day to be trying to solve the future problems and today's problems.

I guess at Google and at DeepMind were spoiled in that we have research teams whose job it is to think ahead. But I think about that as an industry-wide challenge. I can imagine me doing a similar job at a company that doesn't have the kind of research bench that the labs do, trying to think about how to get ahead of these things. Whereas, when we're shipping at a slower pace, there was more opportunity to think about how do you automate things that should be automated, and how do you do more thorough checks before you launch?
I also think that there's an element of education, for lack of a better word, around what risks are out there. The Twittersphere, you have the AI safety Twitter. If you tune in, they're talking about all kinds of things that I think it would be hard to follow for somebody who's just trying to ship a great product. There's just such a long list, depending on what kinds of models you're using, what modalities, like Agentic and not Agentic, and just the permutations of all the things that could, in theory, go wrong. There's no way to tune in and have a full sense of them.
And so I'm curious-- maybe it's less of a concern and more of a hope-- that product people, SRE, all the engineering leadership will take time to also learn about these things, and that we'll have a better shared vocabulary for how to communicate about these things. And just as an industry realize that, yeah, we are in it together. That's one of the wonderful things about SRE and safety. It's like, we have to get this right. Sorry, that's a little bit of a rah rah.
MATT SIEGLER: Awesome. I knew you'd be optimistic, Parker. It's wonderful.
FELIPE: Yeah, I wanted to plus one on the pace of change, the velocity that has been happening over the last couple of years. If you think about when I joined GM, and before that, when I was working on Bart and all this stuff, we were quite separate between the team doing today's safety, what the model is doing something stupid today, versus the team that is doing the next level safety, the next model, the frontier safety. Those were two separate teams. One is doing research long-term, the story about what's going to happen in the future. The other is what we have to do right now to fix the stupid model that is doing something stupid today.
And I feel like today, since two years ago, has changed so much that right now, the two teams are working together because we are right there, very, very close to each other. And this is what actually worries me is that the pace of the evolution was so fast that the long-term research that we had to do, now, we don't have enough time to do long-term research because we have to do that long-term research for the model is going to come up in six months. So you don't have two years to do research. You have a few months only. And by the time that that model comes out, you have to be ready. You have to deploy all the ideas that you had.
So that's why we are now living in a world where engineering, deployment, and research is happening at the same time. And if we don't help each other-- I'm not a researcher from my background, but I have to do research together with the researchers because I have to deploy things that are needed right now. And it's kind of scary because we don't have enough time.
MATT SIEGLER: This has been great. Wow. Thank you so much. Really appreciate you giving your time today. It's been a wonderful episode. Thanks, Steve. Thank you, Felipe, Parker.
PARKER: Thank you, guys.
MATT SIEGLER: Have a good one.
PARKER: Great to talk.
FELIPE: Thank you.
[MUSIC PLAYING]
SPEAKER: 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 Prodcast* is produced by Paul Guglielmino and Salim Virji. *The Prodcast* theme

is *Telebot* by Javi Beltran and Jordan Greenberg.
(ELECTRONIC VOICE) [INAUDIBLE] Telebot.