# Google Prodcast Season Four Episode 9 | The One with AI Agents, Ramón Llamas, and Swapnil Haria

[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: Hey, everyone, and welcome back to the Prodcast. This is Google's podcast about SRE and production software. I'm back. I'm Steve. Matt's back. He's Matt.

MATT SIEGLER: Hi.

STEVE MCGHEE: We have more people, too, also today. I think we're going to be talking about friends and trends because that's the theme. But let's see, first of all, who our friends are today. We have two new friends today. Can you guys introduce yourselves.

RAMON MEDRANO LLAMAS: Hello. I'm Ramon, I'm a senior staff site reliability engineer. I've been at Google for, boy, a long time, 12 years. And I'm in a team that is called core. SRE. So we run infrastructure from authentication services to data stuff. And we have been working now for a year or so in building agents from production management.

STEVE MCGHEE: Nice.

SWAPNIL HARIA: Hi, I'm Swapnil. I am not an SRE, unlike most of us here. I am a software engineer, so I have some experience with production systems. I work in Core Labs, which is a part of Google that's building AI agents for different use cases. And I started at the bottom of the computing stack.

So my undergraduate degree was electrical engineering, so doing VLSI design. Then I moved to computer processors. Then I moved to operating systems for my graduate school, then databases at Google, and now finally, coming up full stack to agentic use cases.

STEVE MCGHEE: Are you going to do UX next and then psychology? Is that how--

SWAPNIL HARIA: If AI doesn't do it before me. Yes.

STEVE MCGHEE: Cool. So you guys both work in Core, which makes it sound like you work at the center of the planet. But I'm guessing that's not really the case. But you guys work on systems that are used by other systems. They're back end to the back end to the back end. Does that seem right?

RAMON MEDRANO LLAMAS: Yeah. How about that? It's infrastructure that is used by all the products. There is as well, another thing that is TI, we call it, which is the back end to the back end to the back end to all of the back end.

STEVE MCGHEE: TI is the technical infrastructure. It's another one of these sets of words which are pretty generic, but they make very specific sense inside of Google.

RAMON MEDRANO LLAMAS: People run into metal, basically.

STEVE MCGHEE: That's right. Cool. Well, so the other phrase, as you said, or the other word you used was "agents" or "agentic," and stuff. And this is where the trends come in. This is a pretty popular new term these days. I think a lot of people have heard it. But it comes with a question mark next to it a lot of the time, or they hear about it.

I have a feeling you guys are a little bit deeper than that when it comes to actually knowing what it is and then using it maybe and on building them and stuff. So can someone start with a high-level, like what are we even talking about here?

SWAPNIL HARIA: So I think the right way to talk about agents is to build incrementally. So we have our static deterministic algorithms, where it's a set of sequences. You have an input. I can work through with pen and paper-- how the operations are going to happen? How the input is going to be transformed into the final output? So that's a static algorithm.

Now slowly, we are seeing cases where some parts of this algorithm could be outsourced to an LLM. So for example, I could have a doorbell camera company, and I want to have a specific sound when the mailman comes just so that folks are aware that there is a package. So I could say "if mailman in frame." And this "if mailman in frame" is being executed by an LLM and giving us the response. So we replaced some parts with some amount of nondeterminism.

And then we go to the other end, which is the full agent, where there is no script anymore. The agent gets

an input. It has some high-level understanding of how to solve the problem, but it comes up with its own set of sequence, steps dynamically on the fly.

At each point, it thinks, OK, do I need more information? Do you know a function call? Should I invoke one of these good old-fashioned static algorithms instead? And over a series of steps, it comes to an answer. But there is no script. There is no input. So there is an input and an output, but we can't manually see how it flows from the input to the final output.

MATT SIEGLER: That sounds a little out of control. No script? How are we going to get to our goal here? I'm a little skeptical.

RAMON MEDRANO LLAMAS: You establish a goal and then you can see the trajectory. We call it that the agent is following. An agent could be just a singular call to a model. You just prompt an LLM with a goal, and that could be an agent. If your goal is like, calculate the inflation in the US for the last five years, that's something that you might have memorized or give you some code to run.

And then that's it. That's a very simple kind of way to see it. And then you can go to systems that are way, way more complex. They are super untraceable. Well, not untraceable, but they are difficult to debug and trace what they are going through to achieve the goal.

MATT SIEGLER: So I'm getting a read here, the agents are goal-directed, perhaps have an unexplainable set of behaviors that they have to get to that goal. And you're going to give them a lot of capabilities. And I'm hoping you're going to tell me what these capabilities can be. How do we define these capabilities? Are they totally open-ended?

RAMON MEDRANO LLAMAS: It depends. So there are tools that you can-- Swapnil mentioned some of them that you can call a tool that is built in your agent. For example, if you open gemini.google.com, there are tools that are already transparent for you.

Google search is one. It calls the search as a tool for continuing the trajectory to a goal that is the problem that you put in there. But then you can have the agent generating code on the fly and executing it in a sandbox.

SWAPNIL HARIA: So when you give a prompt to the LLM, what information does it have? It has a prompt. And then it has been trained to be a good large language model. So there was some amount of training data that was fed to it, and this data has been getting bigger and bigger over time.

But let's say it was trained on March 25. So at that point, its knowledge of the world is limited to what was in

its training data on March 25. And if I ask it things like, OK, should I go out today for a walk, it has to figure out some dynamic information. So that's where tool calls come in, to interact with the external system, try to get more up-to-date information and be able to generate additional observations to answer the question.

And if you just have a few different tools, maybe it knows where my location is, maybe it can figure out the weather, maybe it can see, do I like walking? Is there a good park near me? So slowly, you can see how we can add capabilities over time to give a more refined version of the answer.

And in some cases, we might not even realize what the capabilities that are going to emerge are. So we'll give it a set of tools, but it is mostly allowed to do whatever it wants, until a set number of steps, to give us the final answer.

And that's where some of these emergent capabilities that people talk about, where things just emerge, where it realizes that with these combinations of tool calls, it can actually give us a much better version of the answer to a question we didn't even ask. That's where it gets surprising, but also more challenging to evaluate.

STEVE MCGHEE: So it sounds like the two categories of these tools or agent capabilities are getting at some context that wasn't in my training data. And the other one is doing stuff besides just saying words, like going out and articulating something or calling another agent, for example, would be one of those, or actually executing some code or something like that. Does that sound right?

RAMON MEDRANO LLAMAS: You can see how there is some capabilities that are very evidently simple, like calling the weather service, or if we are in production and you have an agent that is managing production, it's reading some time series data from your Prometheus or whatever are you using, but there are others, like a world modification capabilities or capabilities with side effects, like, I don't know, if we are in production, trigger a binary rollout,

So the tools and the capabilities that you give to the agent, you need to know what you are giving it because knowing or predicting how and when they are going to be called is not that trivial.

STEVE MCGHEE: With great power comes great responsibility.

MATT SIEGLER: Yeah. Is there a notion of a read-only versus a write-only agent, or how do you put boundaries or envelopes around your agents? Talk about the layers of responsibility and capability, because I presume I don't want to set it free in my organization without giving it some ACLs of some sort.

STEVE MCGHEE: Yeah.

SWAPNIL HARIA: So there are a few ways. So typically, at least in the agents that we build today, we don't allow them to make any kind of world modification erm, any ways to mutate the state of the world. So right actions are typically very restricted.

You see a lot of examples of coding agents. And the way these work is it's happening inside a sandbox. So you can edit any code you want inside the sandbox. You can run any tests. So you're modifying the state of the world, but only inside a sandbox.

And anytime you have to break the sandbox solution, if I want to make, let's say, write a request to a spanner database or to a database, you have to enforce some additional checks. In our case, we try to get human permission before it does anything. So it will recommend that you do this, or it will say, can I go ahead and make this call? And there'll be a yes or no option.

In some cases, you'll say, can I see some additional rationale for why you want to make this action? But you need to have additional guardrails in place for some of the write actions, as we call it.

RAMON MEDRANO LLAMAS: Now, this is a common pattern that you see it in tools that are, for example in Claude code, that is a CLI to do coding in your computer and your workstation. It has some safety parameters as well.

So I think the agents that we are seeing both in large production deployments, like in your workstation or in things like the Gemini app or in ChatGPT and these things, do you see how these differentiation between that we are only accessing data for reading and trying to execute an action, like booking you a restaurant or whatever? And there are still a lot of human approvals for all of them.

STEVE MCGHEE: Well, let's move into production in SRE spaces. So now that we have a basic idea of this model and this way of working, which I appreciate, I think that was really well done, what do we do with it? As SREs and people tending production, how do we take these generic ideas and capabilities and apply them to making a more robust and reliable system? What have you seen work so far?

RAMON MEDRANO LLAMAS: One thing that we have seen that works really, really well across the board is what is called one shot summaries, or one shot responses. So you, in production, give the agent access to many data sources, which is nontrivial because in an enterprise, integration of data access is not an easy feat because of permissions or different schemas, different things and so on.

But let's say that you give the agent, like a bunch of access to your Prometheus data. You have access to

your logs. You have access to your things. It can summarize the situation really well for you, for example. We have seen that working really, really, really easy. So when you get an alert, for example, it's going to-- you have an attachment that is “A”. So the situation of the world is like that. So you have this correlation or that.

So for example, in the case of troubleshooting, it saves you all the 5/10 minutes at the beginning to get the situational awareness of, OK, what's this alert and what is happening, kind of thing. You get that done. You need to do the troubleshooting yourself. But I think that works really well. All the things that we have seen that were good, and they are starting to show a lot of promise and reasoning on top of that.

SWAPNIL HARIA: So it's helpful to think of different LLM capabilities and see how we can apply them to the SRE use case. So the easiest Ramon mentioned was summarization. When the alert happens, there is so much information that, at least at Google, an engineer has access to, so things like, how many incoming requests were coming into the system, how many outgoing requests were going out, are there any outages above my service or downstream, below my service?

There are thousands of lines of server logs that you have to go through, thousands of lines per minute or per second in some cases. So there's a whole lot of information surrounding an alert. And just processing them is quite difficult, usually as a junior on-caller at least. You have to quickly parse out what is useful, what's not useful.

So this is one way in which an LLM can help, where it's summarizing all of it and weeding out the wheat from the chaff in some ways, keeping the important bits, removing all of the extraneous information. So that's one. Then slowly, you can get into more advanced things.

If you have a lot of server log lines, there might be an error hidden inside somewhere that looks a little bit like your outage error message. How can it find that quickly? So Gemini has or chat models now have access to large context windows. So you can dump a lot of information in there and say, OK, do you see anything that resembles the error message I'm seeing? So you can use it to find a needle in a haystack if you wanted to.

And then lastly, Ramon said, reasoning. That's where we are trying to build more advanced intelligence in some ways, or at least leverage that. So if you look at the SRE, they are doing two things. They are trying to gather clues from all of these diagnostic systems, and they are coming up with hypotheses.

The LLM can do that for you. It can look at these systems. It can figure out, here's a playbook of how SRE would have diagnosed this issue. I can repeat a lot of the steps. If you're looking at, what is the upstream error message looking like? Is there a spike in QPS somewhere in the system? Am I seeing that there was a

binary released in one of my dependencies?

It can take all of this information, do a little bit of debugging, surface them as higher-level insights, and then the human can quickly make sense of them. And in this case, we saved a bunch of time because the human didn't have to click through 15 screens trying to find the right log messages. They didn't have to add 16 filters to narrow down to the right cell in the right region, for the right service, in the right environment and so on. So it does a lot of the legwork of debugging for you.

MATT SIEGLER: It saved you a bunch of clicks, I get this. Do you have a particular implementation or launch that you'd like to talk about that you feel really great about sharing, that you can share that uses an agent that does save a lot of production time, a lot of toil, that you think is specifically demonstrative of how agents can save SRE effort or automate it out or amplify-- I think "amplify" is the right word here, what you're saying-- using the intelligence of humans to greater effort? I think that would really illustrate what you're talking about well, as well as in a risk-averse way.

SWAPNIL HARIA: So we have alerts firing all the time at Google. An alert, it looks at an error condition. So if your QPS is above a certain threshold, maybe you're returning too many errors to your clients, those are error conditions. And if they stay for some amount of time, an alert gets triggered automatically at Google.

So when the alert gets triggered, we have some use cases where the agent steps in first. By the time the human gets to their desk, which is typically three or four minutes, or they are context-switching away from working on something else, the agent has already done a lot of the common steps that the on-caller would have done.

So for example, was there a release recently? Is the new release showing a higher error rate than the old release? So things like that, it can quickly weed out a lot of the common mitigations. And it can either say, hey, look I found, the underlying root cause. Or it can say, OK, I didn't find the root cause, but I have ruled out these 16 things. So now you can actually go and see what is really the issue there.

And in doing this, we have seen that it can save a lot of human time, when they get to that incident of trying to figure out, OK, how do I find the right context for this alert? How do I find the right mitigation or root cause? And it's still giving them a lot of agency because we are showing them what the agent has done.

And they have to make the call, OK, this looks right. Here's the agent supporting evidence for that. And if it looks right, here's how I can apply the changes directly. So that's the process in which we are using agents right now.

RAMON MEDRANO LLAMAS: We are speaking about saving a few minutes of time and so on for the responder and so on. But what matters is that when you have an alert, you have an SLO or an SLA that is under the threshold or whatever it is. That typically means that some service is bleeding out some reliability. And some users are affected, whatever that is.

Like, in a company, there is a revenue impact. There is some whatever that is, financial or a reputational or whatever, it's shortening these minutes. But what matters is that this impact goes away in the sense of, yes, you are going to have to have humans at the end of the day looking at what happened and so on, probably.

The agent is going to give you just the mitigation action that you can take, and then the root cause is something that you need to analyze and fix later. But this reduces the amount of time that your service is not available for your users, which is really, really what it's about, especially when you are running a very large service like search or Gmail or whatever it is.

MATT SIEGLER: We do talk a lot about anomaly detection and the improvements that AI has been giving our responders. They're real. And the interventions are still at the human level. But that the feedback that we're getting is faster and faster in that the dashboards that we're using are more informative and less toil-y.

And I think what you're saying is the agents can take more action to go retrieve this information, consolidate it, make it more concise, and make it more actionable to me as the on-caller. And that's a wonderful thing that I want to have every day.

Fine. We have these agents. How are you actually deploying, testing, and finding out that they're doing those things correctly? Because I think that sounds like a dream. Does it work? And how do we test that? And what are we testing it against? Because this is against something else. This isn't just something you make, and then we go, that's great. Thanks. Good day. What's our A/B here?

STEVE MCGHEE: Works on my machine, right.

RAMON MEDRANO LLAMAS: That's the golden question, literally, right, how to evaluate these things? I think the evaluation of these new paradigms like services that they are not deterministic in a way. It's very hard for two reasons. So the first thing I want to say is something that Swapnil said at the beginning, that is, when you are doing a development agent, like for coding, for Cursor this kind of stuff, you have a sandbox.

You don't need-- you need to evaluate, but you have the luxury of being at the, I call it, the left-hand side of the development cycle because you spin a sandbox, worst-case scenario is like, just compile your application with the CL or the patch that is producing. And that's the test. If it compiles, passes the unit

test, you have an evaluation there.

When you are on the right-hand side in production and you have agents that they might take actions in production, there is no-- production is not a sandbox. And if you decide like, hey, this agent is going to drain every single cluster or every single data center or whatever you are deploying right in one go, because this is what it is, it's gonna go for it.

So I think the evaluation there is something that we are being very, very conservative in the sense of there is all the safety. But there is as well how we measure that the agent is producing like recommendations or insights that they are correct. So there, we have techniques. There are techniques like sandboxing. There are techniques like having golden data. And there are techniques like using human feedback for this. Swapnil knows this very much, better than me.

SWAPNIL HARIA: So there was a paper in 2021 with a very provocative title. It was called "Everyone Wants to Build Models. No One Wants to Clean Data." And I think that's very much applicable today because, like you said, writing some prompts and having something that kind of works for a demo is very easy. But then showing it that it works reliably 90% of the time is very hard.

And the problem with, at least, internally at Google is for production use cases or outages, we don't have a lot of good-quality data of what actually happened. So we might have 1,000 incidents per day. But in 99% of these cases, we don't have any idea of what the on-call actually did to fix the issue.

We don't have good documentation. We don't have good workflows. And what we want is we want to figure out what the human did because that's the right benchmark for the agent. But there is a lot of work that happens at the beginning of these projects where we figure out what is the right evaluation set.

So in our case, we had a bunch of alerts, and we had the right labels for each of them. So the label was something like, we rolled back the binary. We upsized the cell. We took some emergency quotas. We throttled the user, things like that. These are golden labels that you can have. And for each incident, we need to find out what was the actual action that fixed the issue.

We get a set of that. And then we make sure that the agent is evaluated against that. So the agent's output is checked with the actual labels. Now, this is easier said than done because the real-world data that the agent is looking at, dashboards and things like that, often gets lost over time. It might be retained for, let's say, 20 days, 30 days. But beyond a certain point, you lose access to that old data.

I no longer know what the log said 30 days ago. How do you keep evaluating in that sense? So you need to

keep generating a large quantity of golden data to make sure that the agent is continuously being tested. And you have ways to close the loop here.

So we mentioned that the agent gives you some preliminary recommendations, and the on-caller then makes the final call saying, OK, this is what I need to do. We could easily add a step where they record that final step so that the agent knows it, too. And we can have a feedback loop so that the agent can improve over time.

But that second part, of getting the golden data, what did the human actually do for an outage, or for whatever your use case was, what's the perfect response for a given question? Getting that is very difficult. And in often cases, it has other problems like retention of data and other stuff associated with it.

STEVE MCGHEE: So we went over how these agentic systems can be useful for production in general. And before that, we talked about what are these capabilities that an agent might do. We talked about tools and reaching out and doing things and stuff like that.

But now I want to go into a little bit deeper as to how to combine these things. So my guess is that you don't just write a prompt that's like, you are an SRE. What would you do in this situation? Context dump. Here's 20 gigabytes of logs. What is the process that you guys have actually gone through or you've actually--

And the other half of this, which, I think, this is a joke to tell a story is like, I'm guessing you're not training new models. You're not starting from scratch. So what's that thing in the middle? What is it that you're doing where it's not just like a paragraph of simple prompt, and it's not, like, 20 years of developing models?

What's the thing in the middle that you're doing? What's the complexity that you guys are actually building out? Just walk us through some of that. How would you actually go about doing it?

SWAPNIL HARIA: So it's important to start with the end in mind. So for example, the first step in any of these agent-building workflows is to have something called a golden data set. It has the input. So it might be things, what could users ask the agent, and it has a perfect output or a good enough output for that use case.

So let's assume we are building an agent that can answer health insurance questions. So you would have some common things about, is my health insurance valid if I go outside the country? So that would be one of the questions. And then the answer would be whatever the appropriate response was for that. So you start with that, even before you have any agents of any kind.

STEVE MCGHEE: So this is labeled reinforcement learning-type stuff where you have--

SWAPNIL HARIA: Similar. labels.

STEVE MCGHEE: "This is a duck. This is not a duck" kind of thing.

SWAPNIL HARIA: Exactly. But it's slightly more nuanced because it's not a yes or no thing. The output is actually a large paragraph in many cases. So evaluating whether your large paragraph is equal to my large paragraph of some other agent-generated stuff is not an easy problem. But let's not solve that right now.

So then the first thing to do is, as a human, how would I go about answering this question? In the production use case, if I get an alert, what are the tools that I use? What are the dashboards I look at to try to debug this? Because the LLM is not magic. If we don't give it the right context, it's not going to give us anything useful.

So we start from there. When we have the evaluation set, we figure out, OK, these are the three or four tools I need to get a rough understanding of the system. Let me teach the LLM how to use that. Let's see how the answers come back. Now I compare this with my ideal response, and I slowly try to remove the variance over time.

STEVE MCGHEE: And is there a name for that process that you just described? Is that HLF or RLHF or one of those letters? Is that the same thing, or is that something else?

SWAPNIL HARIA: So what you said is one of the techniques. The overall objective is called hill climbing, because you can think of it as trying to get to a local maxima. So iterative hill climbing is an umbrella term for this. And RLHF, Reinforcement Learning through Human Feedback, is a technique that you could use.

STEVE MCGHEE: So this is not training a new model. This is taking an existing LLM, and then you're refining it or distilling it-- is that another word that you guys use? That's a different word?

RAMON MEDRANO LLAMAS: That's a different word. So distilling s getting a model that is very large like Gemini Pro, blah, blah, blah--

STEVE MCGHEE: Oh, I see, small.

RAMON MEDRANO LLAMAS: --and then making something smaller. So it's not going to have the same quality but might retain 90% of the quality in half the size kind of stuff. So we just use the context. What we do is in context learning. So when we are creating is the context for the prompt, it might be a loop of multiple prompts to the same model with different information.

But the models that we use is Gemini vanilla. You would be laughing at the beginning, but the first prompt

that we had, it was like, you are an experienced Google SRE that is doing blah, blah, blah.

STEVE MCGHEE: You've got to start somewhere.

MATT SIEGLER: Wow. Yes, it's like an adventure game.

STEVE MCGHEE: Yeah. It's fun. Go north. Yeah.

RAMON MEDRANO LLAMAS: That's what our tool is called. And then another resource that is very interesting because the things that we learn on the way is that getting the golden data for starting is relatively-- it's not hard. It's just tedious because you need people to help you.

So we were asking colleagues across the company. It's like, hey, so we need these. For this other time that we are working on, we need a data set to test. So can you fill in this spreadsheet? And people go like that. But I wanted to do AI.

MATT SIEGLER: People love spreadsheets.

RAMON MEDRANO LLAMAS: Yes. No, I'm not. So that's for starting up. Then I think once you have some corpus and you have an idea that is running, I think there are ways to get online feedback. So you get feedback from the users and then from the alerts themselves. You can see-- one comparison is easy.

It's like the agent recommended you to do whatever training your data center or whatever, I don't know, some mitigation. And then the actual response that was taken after the fact, it was the same. So we know, for example, this is a good match there. So when there's mismatches, you can compare to say like, oh, the agent said to do some work or something and then the person that was leading the response did something different.

STEVE MCGHEE: So let me see if I get this right. You have the idea of the training-- not the training of the model but the reinforcement feedback, learning some of those letters around a historical knowledge base of people who encountered these problems. They took these steps. So we're building up some context here.

And then I think what you're saying also is that once the first version of this is running, or once an early version of this is running, it can then also see its own actions, and that's impact as well. So like, I chose to drain the offending or rollback the offending binary. That was bad. And lo and behold, the error rates dropped back down to normal. Does that go into another feedback loop of some kind, or is that beyond what we're talking about?

SWAPNIL HARIA: So this is related to what Ramon was saying earlier about. You could do this for sandboxed agents, so things like coding agents where you can actually test it out. But in production, I'm not sure if you want to let your rookie agent train a cell to see if that was the right hypothesis. So that loop breaks down. And that's why we have to do it artificially after the fact.

So what we do is we get the responses at the time of the incident from the agent. And at Google, we have this culture of writing postmortems for large incidents. Now, they are written in a specific format. We have to do the hard work of converting it from this human readable format, which is very verbose, to a format that we can easily compare the agent's output, which would be, this was the exact mitigation.

So you have to take those two pieces of information and see, did the agent get it right, or did it miss some step? If it missed a step, did it call a tool wrong? Did it interpret the output of a tool wrongly? Or in fact, in some cases, did it not have the right tool to call? And there are a lot of different error scenarios here.

So in one example that we often quote is there were a bunch of tools that our production agent had access to. And all of them were returning dates in different time zones. So one of them would say the alert happened in UTC time. The other would say in Mountain time.

And this was well known for humans who are used to these things, but it was not obvious to the agent. So we have a specific line in there where we say, OK, all the time zones are in Mountain View time zone, which is the headquarters for Google, and we made sure we had wrappers around each of our systems so that they would always return things in Mountain View time zone.

MATT SIEGLER: So there's a working theory that postmortems are the most nutritional ingredient to a healthy SRE culture. And I think you've just proven that to be the case through technology. And I really appreciate that.

SWAPNIL HARIA: I'm sure others have proven it before me.

MATT SIEGLER: But you did it through an actual experiment. And I really think that's a wonderful outcome here.

RAMON MEDRANO LLAMAS: It's this super great training data because adjusting or golden data for two reasons. It's not only that you have the response of, this is the mitigation that we had. You have the trajectory of the person, typically in postmortems. And you can see them in the SRE book, the people that are listening. We have example postmortems. And you have what is called the timeline.

So you have all the steps that the person took. So when we are inspecting the agent output, what we're

seeing is exactly that timeline but produced by one of the agents. So you can see, I'm calling this tool, and I got this data. This is a red herring. And calling this other tool and running this other data. So this is promising. Let me follow that.

Which is what you see when we are responding to an incident. One of these larger incidents is where you have a chat for the response and all these things. You see in a trajectory that is like a human trajectory for doing what the agent does that we reflect in a postmortem.

But this is fantastic because this is-- one piece of data that we need is not only the action that was taken at the end. So the correct mitigation action is X, like it's raining or whatever. But the trajectory of steps, that's something very important. I think that something is containing the postmortems all the way.

MATT SIEGLER: Have you ever proposed improvements to postmortems from ad hoc? Was it post hoc analysis of work that you can derive from things you've seen?

RAMON MEDRANO LLAMAS: Format. For example, the format. The format of the postmortem is something that we can do. The open formats, we have many postmortems in Google Docs and so on. It's difficult to mine for reducing agents.

And then there are all sorts of-- one thing that we discovered working on this is that the majority of the work that you are putting together for working in AI is not in AI. It's integrating different tools, different data access, accessing different postmortems in different formats for different teams that there might be old, for example, postmortems from last year.

It takes a while to curate and process all of these. So for example, just formatting the postmortem in human and machine readable format. So they are all the same. That gets you like a head.

MATT SIEGLER: Thank you.

SWAPNIL HARIA: We are trying to build a common language of sorts for some of these mitigations. So we have a taxonomy. So for example, if I ask you, what are some common production mitigations that you would take? You would roll back a binary. You would add resources and so on, run rollback experiments of some kinds.

Now, the thing is, we want to build a common taxonomy of mitigation so that when the agent says, do this, all of us, regardless of what product we are working on, we know what that action is. And it turns out one of the problems that we faced is that a lot of different teams have different words for the same thing.

So in our initial version of our taxonomy, we had "escalate," which would mean that you go to a different service because some other service is returning you an error, so you escalate it to their on-caller. But the incident reporting system we had at Google used "escalate" for a very specific reason.

So when we actually tried to expand the usage of this taxonomy, they said, OK, hey, no, this doesn't work at all, because some people thought that they need to increase the severity of this incident. So we had to move to "delegate."

Same with "experiments," different teams consider "experiments" to be different things. So when we say "roll back an experiment," they might understand it differently. But we need this common language to be there in all of the postmortem so that the agent, which is the same across these teams, can improve consistently.

RAMON MEDRANO LLAMAS: There is a nice publication that is called Generic Mitigations that we made public. I think it was like a few years ago. It already has 20 of them. So I recommend people to read that because this is what we are referring to.

STEVE MCGHEE: Yeah, actually one of the first episodes of the Prodcast was with Jennifer Macy, and she talked about exactly this. So we can definitely cite that one in the show notes as well. I have a side question. We've been talking about the bad day, the incident day, and response, and all this kind of stuff.

What about the other days? Can we do something with agents that helps my daily work when I'm not panicking or when I'm not helping people who might be panicking, something like that? What do you think?

RAMON MEDRANO LLAMAS: So we are looking into two places that the agents help, because luckily, the incident day is not every day. So it's actually the minority of the days. When it's stressful and you need to be prepared for that, and it's as well the evident-- it was for us, at least, it was an evident case at the beginning. It was like, OK, let's solve this one.

But there are two more things that are very interesting that we are looking at that day in general. One is your companion for doing things. In 98% of the time that you don't have an incident, you still have to touch production because you need to do your rollout. You need to observe performance regression. You need to do whatever. You need to move one service from one cluster that you have in some region to another.

Agents can help with that, because at the end of the day, the tools that we use when we are responding to an incident and the tools that we would use in this case are the same. But it's like, you need to apply a drain. You just call the capacity tool or whatever, kubectl, that you are using for the day. And it's essentially the same thing.

And the other is, the best time to mitigate an incident is 0. So it's preventing the incidents from happening. And there is also-- it's more complicated because there is all sorts of risk prevention that we can have. So there are things that we know that are bad.

For example, one thing that is very common, and we have spoken about them in the books and so on, is that many outages are driven by change. So they are policies and they are best practices around many different places, not only Google. But you do changes progressively, do changes incrementally by your different geographical locations or these kinds of policies.

So many times these policies are not enforced because you need to remember to put it in your configuration or whatever or your service configuration drifts over time. So agents might help us enforcing and discovering these spots where there are risks that we don't know about, dependencies, a service that has five 9s of SLO depending on one service that has two, because someone started sending some RPCs, or whatever it is. So discovering these kinds of things can help in detecting things before they manifest into an incident.

SWAPNIL HARIA: Let me give you an example of this. So at Google, we build planet-scale systems. That means we have access to a lot of sophisticated techniques for caching, admission control. And we have all kinds of libraries and utilities that we use internally.

They offer a lot of different control knobs for different things because they have to handle all kinds of cases. And there might be cases-- let's say you have a service that you are trying to add a caching layer on top of or admission control. Now, with admission control, there are many ways to do it right. There is FIFO. There is first in first out, last in first out, and other variants thereof.

If you decide on one of these strategies and if at this point, the QPS that you are getting from your customers is something that would break your SLOs because of this service, that is something that the agent could flag because it has access to the dashboards. It knows what QPS you are seeing. It has access to other dashboards of the other teams around you, and it knows what their admission control looks like.

So even before you make that change-- so when you send that change out for review, it can step in and say, hey, look, you've configured the admission control in this way, but I don't think that's the right idea, based on what I see. So that's one example of how it can step in before you do something that will cause an outage.

STEVE MCGHEE: Cool. Well, I'm afraid we're running out of time. So I have one-- this is called the lightning round. So each of you gets to answer two questions, but you have to just answer just those two questions.

So we're going to say like, let's see, number one is going to be, what do you think we should not be using LLMs and AI for in production today? What's a bad idea?

And then number two is, what do you want to work on next? What's the next big thing that-- let's say you're just done. It's a year from now or something like that. What's the next big thing you want to take a bite out of?

RAMON MEDRANO LLAMAS: I'll start. One thing I would not use LLMs for is for when you can use a regular expression for things.

STEVE MCGHEE: Solve problems.

RAMON MEDRANO LLAMAS: I've seen people-- yes. So I have seen it-- I want to extract this integer from all these texts and so on. Let me prompt Gemini for getting that integer. It's like, no need for that. That's fair. This is just a joke, but there are some cases where the classic statistical methods or very small specialist models for doing certain things, like anomaly detection of a time series.

It's a much better-suited model. They're faster, cheaper, and more reliable. So I think we need to balance using the LLMs for what they are good for and other techniques for what they are better-suited.

MATT SIEGLER: So you're saying math still has value, even today?

RAMON MEDRANO LLAMAS: Yes, math is still not deprecated.

STEVE MCGHEE: Actual operators.

MATT SIEGLER: Really happy to hear that. Thank you.

RAMON MEDRANO LLAMAS: Yes. What I would like to work next I think is the-- so how is the autonomy of these agents is something that is incremental. So we are going to see levels of autonomy going on. So I think we are at the beginning of this because all the agents that we have and the agents that I have seen around, they always require a human verification before going and execute some actions.

So I think one thing that I want to explore is like, how could be the conditions or what we should be having in production for leaving the agents to do things automatically? So how we could, for example, recover from an agent breaking some stuff or the agent recover itself and these kind of things? I think that's going to be a very large challenge, particularly in the production space.

STEVE MCGHEE: Swapnil, what do you think?

SWAPNIL HARIA: So I think one thing I would not like to see LLMs used for is insulating humans from challenging situations. So I'll give you the example of production outages. We are building agents that help out, but we don't want it to work in a way where humans don't know what is actually going on.

Because as engineers, that's how we learn over time. Our designs get better because we've gone through these outages. So we don't want it to be the case that it's completely offloaded to an agent and we have no idea that an outage even happened because the agent took care of it for us, because in that way, we'll never learn. So that reinforcement loop is broken for the humans itself. So that's one thing I wouldn't like to see LLMs being used for.

STEVE MCGHEE: Well, I'll throw it out there. There's a famous paper called The Ironies of Automation, where the more you automate a thing, the worse you get at it. You have to depend on that automation. So when it doesn't work, then you're in trouble. Yeah, so we don't want to break that loop. I'm totally with you. What's next?

SWAPNIL HARIA: I think so we are still deep-- there's a lot of problems to solve in building production-scale agents out there. But one of the things I'm often interested in is interaction models. Right now, we are talking about embedding AI in all kinds of things. But if you look at the interaction models, they are still very limited.

It's mostly a chatbot which comes in many forms. Or you click somewhere and you get some output and you can't answer back to the agent. But I suspect as we get more familiar with these systems, there might be a lot of other potential use cases that emerge just because we've changed the interaction models.

So it's interesting to see where we could take this, because at this point, I'm not an LLM expert. So I have very little control of how the models emerge over time. But I think the models are good enough. And the thing we are lacking at is improving the communication between humans and LLMs, which is where the interaction model comes in.

STEVE MCGHEE: Nice. Well, that was a very great discussion. Thank you so much. We could definitely keep going, I have a feeling, on and on about this kind of stuff. Matt, any closing thoughts?

MATT SIEGLER: Yeah, agents are the new software. I had no idea. This is it. They're going to do it. They're like the composite thing. I had no idea this is what we were going to be doing, but we're going to be doing them.

RAMON MEDRANO LLAMAS: I was reading today one thing that is like an AI is something that computers

are not good at. But then when it gets good at, like we call it software. So I think in a few years, we'll call all of this software, and we are done.

STEVE MCGHEE: Just another flavor of software. Well, thanks very much, guys. It's been a blast. So long.

—

[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 and Matt Siegler, 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.