

Google Prodcast Season Four Episode 5 | The One with SLOs and Sal Furino

[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. Welcome back to the Prodcast. This is Google's podcast about SRE and production engineering. I'm Steve McGhee. And who else is here today?

MATT SIEGLER: I'm Matt Siegler.

STEVE MCGHEE: Hi, Matt. You're back, excellent. We have a guest today from the big city, right? The Big Apple, I think it's called these days. Hello, guest. Who are you exactly? And what are you doing here?

SAL FURINO: Hey, everybody, I'm Sal Furino. I'm a customer reliability engineer over at Bloomberg. And I love service level objectives. They're something I'm just really passionate about. And I'm really happy to be here on Google's Prodcast.

STEVE MCGHEE: Awesome. Welcome, welcome, welcome. Bloomberg is a company that handles money or something, or helps people handle money, I think. What do you guys do there?

SAL FURINO: So what we do is we provide a lot of financial information about the markets to the world. So if you like that stuff or you like news and media related to it, which is kind of expansive if you think about the various different commodities and different things that are out there, we cover a lot.

STEVE MCGHEE: Would you say that computers are involved?

SAL FURINO: Oh, yes.

STEVE MCGHEE: OK, cool.

SAL FURINO: Many, many computers are involved.

STEVE MCGHEE: All right. Cool.

MATT SIEGLER: Once in a while.

STEVE MCGHEE: Excellent, excellent. So I think we're talking to the right people here. And so, you mentioned service level objectives. We like to call those SLOs. It's mostly funny for me because I live in a town called SLO (San Luis Obispo, CA) also. You know that. It's the best. I hope to move to Oslo someday just so that I can live in the other city that has SLO in it. I think that would be pretty cool. But if people are listening/watching to this and they don't know what an SLO is, what's your initial explanation to folks like, welcome to the world of SLOs? And have you met your new best friend?

SAL FURINO: So that really depends on who the audience is, because I think I explain SLOs much differently to my family or other, let's call it, non-technical people than I do a technical person. So where would you like me to start first?

STEVE MCGHEE: Well, I like to think that--

MATT SIEGLER: What's our clique?

STEVE MCGHEE: Yeah, Matt, who's our audience here? You tell me.

MATT SIEGLER: You're an engineer. You work on the software system, you have customers, and you are concerned that they're happy, and you'd like to increase the happiness of your customers. And you want to do that in a structured way. And you've heard about this thing, and you think this might be a structured way to do that. Go ahead and lead them forward through this path.

SAL FURINO: Yeah. So if you're an engineer and your team or someone or a team you're adjacent with produces software that people use, which I think should be most software systems out there, you should be familiar with something called telemetry or observability, that should be in most people's wheelhouse.

So if you think about what an SLO is, it kind of comes together with four things. First, there's an SLI, which, there's little debate about what an SLI actually is or isn't and so forth, if it's something that's a valid SLI, but let's just handwave all that away for just a second. And if you think of an SLI, you can think of it usually as a metric that's generated by some type of query. You're producing.

STEVE MCGHEE: It's an indicator. The I is an indicator, for starters, right? So it's a metric. It's a thing that tells us something, right?

SAL FURINO: Yes, exactly.

STEVE MCGHEE: OK, go ahead.

SAL FURINO: So after you have this indicator, you then have an objective, which is generally a value-- in my mind, it's a value that relates to the underlying metric at hand. So if we're talking about durations or latencies, these are-- be things like, I don't know, 500 milliseconds, 1 second, 5 seconds, et cetera, some unit of time.

If we're talking about maybe some type of availability. It might relate to some type of status code, such as HTTP status codes, for example. Like, maybe 200s are good, and maybe 400 or 500 are bad or something like that. It's some indicator of some bit of thing. So that was the second point.

Then the third part is the target. The target is how often the SLI needs to meet its objective. This is usually expressed by some percentage value. And being that we're in Reliability, we love nines. I'm not sure why nines are like the thing for our industry, but they really are. Maybe we blame the telecoms for starting out with five nines or something. But some people generally say there's 1, 2, 3, 4, 5 nines of reliability, and so forth. And an example of one 9 would be 90% reliable, or two nines, 99, three nines, 99.9%, et cetera. Yeah.

And then, lastly, we have the time window, which is how long the objective needs to meet its target. So these are things that could be a bit variable. They could be 15 minutes. They could be an hour. They could be a day. They could be 30 days. They could be 90 days, or a quarter. And there's different ways in which you can play around with time windows there.

But what's really neat, and remember where I said there was four things to SLO, there's actually a secret fifth.

STEVE MCGHEE: Perfect.

SAL FURINO: So when we get to that secret fifth thing there, it's an output of those other four. And that's something we like to call an error budget. And this is something that can generally be expressed by a sentence. Can I give a shout-out, actually, to somebody?

STEVE MCGHEE: Sure, totally.

SAL FURINO: Yeah. So, like, Fred Moyer, like, popularized this idea of expressing error budgets as a sentence when you put it all together. I think he gave that talk I think at Monitorama a few years back and such, and it explains it really nicely.

So let's take an example. So let's say we're an e-commerce website, right? Everyone has used-- has bought something online probably in the past 10 years or so. Something that's really important for e-commerce websites is how long it takes to process your order to check out your cart. So this is something that if you do all this work, building all the carts, shopping, all the stuff, and then I'm trying to give you money, and you cannot take in that money and process it fast enough, I might leave and abandon that cart, and you lose out on a sale or revenue. It's critically important to that business. It's an important customer-user journey.

And let's say for that example, we say a good experience for cart checkout is 500 milliseconds. And we want this to happen 99.9% of the time, three nines. And we were concerned about this over a rolling one-day window. When you put all that together, you could express the error budget as saying our error budget is 0.1% of traffic over a previous 24 hours is allowed to take longer than 500 milliseconds for cart checkout requests. So it's a different way of framing the amount of reliability or unreliability in which your service can tolerate.

Another way to think about this as well is how bad do we have to be in order for us to lose reputation with our key customers, users, or stakeholders of this capital S service?

STEVE MCGHEE: How bad is too bad? Yeah, When should we step in? Like, is it cool if it's burning a little bit? Does it have to burn a lot? Like, when do we really care? Because, I mean, the other option is, like, Sal, why don't we just, like, pretend it's perfect all the time? Like, let's just. Let's just aim for 1,000,000%. Like, that's a number, right?

SAL FURINO: Right, yeah. So I guess you could potentially be 100% reliable, but then that's going to be extremely limiting. That's extremely limited. You're probably going to have a system in which you don't make much changes, stuff that's very static. You're probably going to be, like, triple or quadruple redundant on different things there, and sensing failure--

STEVE MCGHEE: Sounds expensive.

SAL FURINO: --and sensing and predicting failover. It's really expensive, really costly, and probably doesn't fit your operational model if you're in a competitive space. So, once you start accepting that you're going to fail or you're not going to be perfect for everybody all the time everywhere, then you can start accepting the idea of an error budget or an allowed amount of unreliability. And once you start accepting that idea, you can start coming to think about it as, hey, how much unreliability can we use and play with in order to make experiments and to better understand how our users are using our system and how they're experiencing it.

MATT SIEGLER: So you've set some of these SLOs. Let's say you have one or two of these, and you've decided around, you eyeball these. I think this is a good target. I don't know. Your business leader is like, "sounds good". 99% of people will be happy because within a half a second, it gets pushing. "Buy" actually gets me a "bought".

Let's just imagine I have 99% click buy actually results in the thing getting bought. I think that's not a bad target. Actually, that sounds pretty terrible. 99% is probably not nearly enough. But my running a teeny little shop, and it sounds good to me. And I have another one, which is, I don't know, my website actually is there 99% of the time. And when they come to open the store, it doesn't, like, just crash immediately upon loading.

These sound like good starting points. What are the things we can do now that we've put these there on the ground? We have a website. We have them in the-- and you described a budget. What can we do now that we have these in place? What does this allow us to do?

STEVE MCGHEE: Yeah, that we couldn't do before.

MATT SIEGLER: Yeah, right. We were just kind of guessing. Now we have a thing to measure, and it's telling us a story. What has this story allowed us to do with some control?

SAL FURINO: You mentioned, I think, the two most popular SLOs. And they're probably like the entry-level ones, which I feel are really good, and people are starting to get ideas or accustomed to the idea of SLOs and reliabilities and error budgets. And those are around availability and latency.

Like, it can generally be summed up as, like-- I was talking with Hidalgo recently, and he mentions that there's only really two SLIs. Did we give the users what they want? And did it happen fast enough? But did we give them what they want, is like really, once you unravel that, it kind of gets into a rabbit hole of all the different ones, all the different SLOs that are out there.

But with those, I think they give you-- start giving you insights of are we successfully responding to customers when they give us a request and are we responding to it quickly enough or giving a response quickly enough? We're not just saying if it's correct yet. We're just saying a response, which is generally really important. So, like, you can think about it this way, if you see that your availability error budget is starting to burn, you're starting to burn some of your availability error budget.

But you see, your latency is completely fine. It might just be because it's faster to serve up a 500 than it is to serve up a proper 200 response to actually process an order correctly. So I think there's also some maturity that happens in understanding how your different metrics work together and leading you to troubleshoot an issue where it happens.

So let's go back to your example of that tea shop you mentioned. And let's say you start getting, like, funneled. You start to go viral. Great for business but probably terrible for your infrastructure structure for how it's set up. If you're just a small local tea shop, you have to be able to handle that load and scale it accordingly. So this is where there's other additional SLOs, such as those around saturation and understanding. What are the bottlenecks of your system, and where do they provide poor customer experience?

So maybe you're bottlenecked on, I don't know, disk I/O or network connections or some other way in which you have hard-to-fix limits based upon how you've architected your system, and you start measuring how you're doing towards meeting that capacity. And I will say, saturation SLOs are probably a deviation away from measuring the customer user experience and more so measuring the health of the technical system. But they still are important to know when and how to scale your systems effectively.

STEVE MCGHEE: So this is all really good, Sal. Like understanding what these SLOs are for. And like, people talk about SLIs versus SLA, and it gets a little confusing. But like, it's been documented, and you gave a really good rundown here. And I was going to make a joke about, like, these are simple and, like, it's not like there's an entire book out there that explains this in full detail.

But, like, you mentioned Hidalgo a minute ago, that's Alex Hidalgo. He was the author of the SLO book, and he was a SRE at Google here. Like, we worked with him in the past. You've worked with him in the past too, right? Like he's a known entity out there. But, like, yeah, the point is, like, it sounds simple, it gets hard. Like, there is an entire book. That's kind of my point, I guess.

But, like, your role that I've seen you do in the past has been to help teams parse the book and, like, apply it to their lives. Like, how do you take the time, right, and then apply it to your daily ritual? Like, what's so hard about that? He says jokingly. Like, why is this hard-- like, do you have examples of what's hard or where teams have run into--

SAL FURINO: Oh, so many examples.

STEVE MCGHEE: You go, I'm sure you have a bunch. Yeah, yeah.

SAL FURINO: So, let's actually maybe start it off first. And let's try to maybe step back just a little bit and talk about trying to explain SLOs to non-technical people. And this is sometimes I think making them more relatable to people in everyday experience makes it a little bit easier.

So let's say, for example, you're going out to lunch. If you go out to lunch and you're going to sit down at a restaurant and so forth, you might have different expectations of what level of service you want to get. If you're going over to the local cart in the corner-- I'm in New York City, so there's carts everywhere-- you might have different expectations than if you're going to a popular deli or the local market that has a sushi deal or something that has a long line.

These are different expectations you have around visiting and interacting in these places in your day-to-day life. You also have maybe different expectations of how quickly you will be served. So let's say we go, for example, to the local diner. It's hot outside. I want to go inside and actually sit down in some air conditioning. I sit down, I get served, and I place an order for a cup of soup.

A cup of soup, you know they're not out there like boiling the water and chopping the vegetables and putting it all together, making it to order. No, the soup is already there and prepared. You got to generally just ladle it out, put it on a bowl, maybe do a little touch-up, a little zhuzh, or something, and then go and push it out to you. Versus if you go and order a well-done steak, you know it's going to take much longer to prepare.

So there's some things, based upon the nature of your order, nature of your request, you have different expectations of how long they will take to get to you. Likewise as well, if you go there and it's dead and it takes a very long time, versus if you go there and it's completely busy, it's kind of just more expected that, oh, hey, you're going to wait. It's going to take more time for you to get on queue. So I try to phrase it in that kind of sense to people.

STEVE MCGHEE: It's about meeting expectations, right? It's less about the absolute value of it. Was it fast? And fast is always good. It's like, what did I expect, and did I meet or beat it, right? That's important.

SAL FURINO: Let's take an example, like, if I ordered a well-done steak and it got out to me really quick and I cut it open and then I see it's medium rare. Well, first off, I think that's the proper way to eat a steak. That's my opinion. But someone who actually had that well-done order would probably be upset because, oh, hey, yes, it was fast, but it wasn't what I wanted. It wasn't that right specification.

STEVE MCGHEE: So it's like we would call that correctness or quality, right? Like, it was on time and it arrived, but it wasn't actually right. It wasn't what we asked for. So it didn't meet our expectations.

MATT SIEGLER: It sounds like your latency-- sounds like you're highly available-- sorry. It sounds like your latency availability trade-off you mentioned earlier. Like, we served it really quickly, but it was the wrong thing. OK, there you go.

SAL FURINO: Well, that wouldn't really be so much availability and latency. Availability would be like, oh, I want to place my order, and the wait staff hasn't come to my table yet, and I'm trying to flag them down.

MATT SIEGLER: OK, OK.

SAL FURINO: I think that's more akin to availability. Getting the wrong temperature on your steak or maybe the wrong item go to your table, that's more something we would call data correctness in terms of SLIs.

MATT SIEGLER: OK. Thank you for clarifying that.

SAL FURINO: Yeah, yeah, yeah. And all these examples, I actually gave an internal lightning talk around, like an introduction here at Bloomberg around SLOs. And I related them all to holiday seasons and the different expectations people have of getting together around Thanksgiving and how they all work together in this system and different expectations people have of having, let's call it, a family gathering.

At the end of the day, I like this approach, though, in explaining it to engineers so they understand and put the person and the customer or the user forefront and first. That is the most important thing with SLOs. I gave a talk at SREcon, and I led it off, with the first item being to measure what matters to your users. Don't measure like CPU utilization or disk I/O. That rarely is an important or useful SLI for your

SLOs. It's much better to try to come up with proxies of the user or customer experience and start measuring that.

STEVE MCGHEE: I like to joke with folks that traditionally yield the metrics included, like the temperature of the CPU. And like, at the end of the day, like, I don't care how warm your computer is. Like, what does that have to do with customer happiness? Like, absolutely nothing. And so this is like an obviously silly example. But there's still teams out there that are measuring disk fullness in a SAN where there are many disks that are redundant to each other. Like, what are we doing, guys?

But like, this is, I think, the kind of gold like core of SLOs is that they're meant to be user-centric, right? Like customer-centric and not about the tech itself, but about, like, is the whole system working. Is that something that you think people get? Or is it a matter of not getting it but actually applying it? Like, where is the disconnect, or where's the struggle when it comes to this?

SAL FURINO: So I'm going to answer that in a second, but I'm going to take us on a little detour. So you mentioned two SLIs there. Let's call it the temperature of the device and also the disk being full. I'll say the temperature device. If you're on mobile or providing a small device-oriented experience to your user base, they can have expectations on how hot that thing is running--

STEVE MCGHEE: Oh, yeah. That's true.

SAL FURINO: --your device. So, like, if you're coding for mobile and I notice my phone's getting really hot every time I use this app, I might want to use it sparingly in the future. Likewise, if it's always draining my battery or using a bunch of data, there's some exceptions to these rules that were coming in place. like based upon your instance, that in which they're being applied to.

So this is where it starts, like, oh, hey. Yeah, they're really simple. But wait, wait, wait, hold on. Where does this iceberg end? It keeps getting, like, wider and wider and deeper. And likewise, your example with the SAN and the disk filling up. Hey, if I'm an application team, absolutely correct.

But if I'm the storage team in terms of providing storage service towards everyone else to use. Oh, hey, no, maybe I should be watching these, how our repertoires are doing and how much things are filling up, and how quickly the rate of-- the throughputs or rates are going for that. So I feel this is almost like there's always exceptions to the rules. So when we talk about these things generally, of course, there's always caveats here and there, but I would try really to think about what is my core user or customer experience I am providing.

So now let's take it back to your question you actually asked, and that is do infrastructure teams or do engineering teams sometimes struggle to figure out what is that user-customer experience? And I would say absolutely. This is common across the industry.

There was-- I think it was in the DevOps DORA report in 2023. They had a quote in there. There was something to the effect of engineers generally measure the health of the technical system and not the happiness of its users. And that's just a trend across the industry. And that's something that a lot of people struggle with.

And I feel it's because a lot of engineering teams sometimes don't have a product person there to think about it. They generally think about mostly of this whole Rube Goldberg machine of services and things working together. And they're concerned about the machine and the health of it.

They think-- actually, I'm going to give a call back to Star Trek. One of the most famous episodes. Was it Trials and Tribbles? Where they have all the tribbles are exploding in population on the space station, and so forth. And Scotty gets, like, upset when they insult the ship and not the ship's captain.

And I feel, like, this is where engineers are, they have pride in the things they go off and build, and they generally don't think about how they're used or applied. And this is, I think, just something we kind of have to get over as a whole.

If you're an engineer and you're not measuring your happiness of your users and people using your system, and you're just measuring the health of the system you built, what are you doing? You need to step up your game, and you need to think back of what does it truly mean to be an engineer.

If you're an engineer and people want to go and cross a river, you shouldn't be handing them a book on how to build a boat. You should build them a bridge to get over it to make it easier for them to walk. So in my mind that's, I think, more like an ethos of what an engineer is. We're making things easier for people, and you need to keep your people in mind of who you're engineering them for.

MATT SIEGLER: OK. So with that in mind, let's suppose you've approached a team. Let's imagine you're the oracle. You show up with this team and you've given them the philosophy you just gave us. And you said, all right, I want you to take this approach. I want you to speak for the users. You've stepped up their product game.

You've got them to build a good amount. Will get to the question of how many is the right amount of

SLOs. They have good SLIs. They're measuring them, they're acting upon them. And now the devs are going, man, I feel like these things are there and I don't really understand what they're for. And they're kind of getting in my way.

And there's now a back and forth between these two teams, where the devs are like, I feel like I'm being held accountable to these SLOs, but they're not what I do every day. And there's another team kind of defending and responding to them and going back to the devs, and I feel these two camps exist now in this organization.

Now, how do you build a healthy relationship between these two camps, between the SREs that are working with the SLOs and the devs that are working on the features, so they work together so they feel comfortable and they feel like they're on the same team going in the same direction? Because I feel that is some of the cultural stuff that kind of exists in the friction that exists or could exist if you don't. So talk a bit about building healthy ecosystems with those.

SAL FURINO: At the end of the day, SLOs are a framework to have joint decisions about the reliability or unreliability of your systems. These are-- you need to have product engineering and leadership involved in making, creating, implementing, tweaking, tuning, adjusting, reviewing these SLOs. Otherwise that means that you really just have sparkling KPIs. They are a team thing that everyone must work together to implement and care and feed towards it.

I'm not saying the SLOs we create today are the ones we use tomorrow. Not at all. Like SLOs are living, breathing things that if it's not providing value, it's not useful, kill it off and move on, and try something else, try something new. If there is some stuff that is really important, keep it around and maybe you just have to adjust some targets or some objectives in order to get that additional value in which you seek from it in order to get better burn rate alerts or other percent error budget remaining stuff as well. Like there's various triggers in which you can use together to help make an informed, more useful decision.

And I also feel that, based upon your persona of these different teams and things involved, you're going to be looking at slightly different SLOs, but maybe have the same SLI and objective. The thing you're changing about them now is the time windows. I feel the time windows for people maybe responding to incidents. So these are production engineers or these are SREs or people who are generally on call are probably things in the shorter time windows. There's probably like the 1-hour to 48-hour window kind of range.

If you're in the app or dev team, maybe you're something more concerned or something more aligned to weekly or sprints-- aligned to your sprints and what you're doing. If you're in product or leadership, maybe you want something more monthly or quarterly based, and maybe not even rolling at that point. Maybe you want them to be calendar-aligned because that's how you think and talk about the reliability and unreliability of your services.

STEVE MCGHEE: Yeah. I've heard SLO described as the lingua franca between teams or like sort of jokingly as like Latin in the dark ages. It was the way that you spoke between countries is like through your-- people who could speak this one weird language that was, like, common between your different countries, but basically I like the lingua franca better, in that it's like the interface language between two worlds.

And so, like, we talk about both the numbers in the infrastructure and the data with the numbers part. And we also talk about the users and the happiness in the same sentence. And also, like, they tend to be based on facts that are, like, in a dashboard on-- a graph on a dashboard that we can-- both teams can look at the same time and be like, oh yeah, it is above the number that we said it shouldn't be above, or whatever.

But I really like your point about looking at different timelines for different teams. So, like, if I'm on call and I'm getting paged and I have to do a thing, I tend to care about, like, what has just happened recently and what I can do next. But if I'm the CEO, I'm looking at, like, yeah, quarterly revenue. Those are very different time scales. And you could potentially use a similar, if not the same, type of number to compare, are things OK right now on different timescales. So that's a great point.

SAL FURINO: Yeah. I also like to think about it as well. If you have trouble figuring out what these different personas are, thinking about it as the different actions people take to the reliability and unreliability of the system. People who are on call, they're more here and now, they're logged into production. They go off and do things. People who make features or think about new ways in which people can use the system, they think about in generally longer timescales than the next one or two days, at least, maybe, hopefully. I don't know.

If you're thinking about more of like, oh wait, this third-party service we're using is part of this critical customer journey, and they're contributing largely to unreliability. Wait, do we have to change vendors or do something different here? These are at different levels of the organization. They think about things and solve problems at different time scales.

So, like, this is something where when you're designing your SLOs, you need to think about designing them towards the persona who will be using them and taking action to do something with them. And I know this is like a whole lot, and this is why--

STEVE MCGHEE: Yeah, yeah it's a big topic.

SAL FURINO: It is probably why I have a job, to actually help break this down and understand and break it up into all these different areas. And this is starting to get where my job as a CRE, a Customer Reliability Engineer, is almost sometimes, at times, like being more of a couples therapist in a way, just trying to coordinate between these different personas, these different teams.

And especially if we talk about user journeys that are shared between multiple teams. There's team A and team B, hey, how do we both contribute towards our larger customer journey? What do we share? What do we take apart separately? And how do we both contribute towards achieving the end customer experience?

And the goal at the end of this is to avoid finger pointing and more of just both of us working together and agreeing that these are reasonable.

STEVE MCGHEE: Nice. Let me ask you a question more specific to your current role. Or in general, like, in finance versus any other like a vertical or industry or whatever, like how do you see-- or how have you seen SLOs present themselves in that world that might be different from other industries, or it might be more uniquely interesting, or something like that?

SAL FURINO: So before joining Bloomberg, I was working at a startup that had a lot of interface to a lot of business consumers and business to business customers. But now that Bloomberg, we're essentially a business to business enterprise. People need to spend a fair bit of money to buy a Bloomberg license and to use the Bloomberg terminal and the wealth of information it provides. So it's not just like, oh, hey, I just give you my email and then I could go off and use this thing. So there are different expectations and how that software needs to perform and what it needs to do for people at different times of day.

STEVE MCGHEE: Yeah. So we're back to expectations. Like that helps you describe what those SLOs are going to be. And if your expectations are different, you're going to write different SLOs.

SAL FURINO: So when we talk about some differences between business to business versus business

to consumers, if you're somebody like Google, you have a lot of business to consumer activity, you have a lot more traffic, a lot more data points flowing in day in and day out. I'm sure it probably follows some type of shape based upon daylight hours, or something like that, based upon where your general user population is on the world.

And, I think, in finance that's maybe a little bit more different. There's a lot more concern around when your business users, your actual customers, are using it and not so much at other times. Are those other times still important? Yes, absolutely. And they do inform your SLOs, but there's some different additional steps in which people need to take there. And I think it gets all back to what's the expectations of your user base, where are they located, when are these things happening, and understanding that. Once you understand that, you can better understand the different volumes of your service predictions and stuff like that.

STEVE MCGHEE: Yeah, I used to run two very different services at the same time. I was in charge of Android mobile stuff and then YouTube stuff at the same time. And it was a joke that we could, on a global, one or two-day long graph, you could see the Pacific Ocean like that was the joke, because you could always see there's this dip where no one's awake.

But the funny thing was, in these two services, the ocean was in a different place because people are using their phones to do phony things-- phone things during some part of the day, and then they're watching YouTube at different parts of the day. And so, it was all shifted by two times zones or something like that.

So it was very, very funny. Like it's even within the same company and the same infrastructure, even a different application within the same product, often you'll have just different trends like this that you have to be aware of. And so you got to be really careful not to just copy and paste SLOs and values and time expectations and things like that over time.

That brings me to my last section. And you hinted at this before, which is when you said our SLOs today might not be our SLOs tomorrow. So like, there's got to be, like, a life cycle of this stuff of, like, you got to-- I'm guessing that you get better at this with time. And maybe you start out by making too few or too many, or too precise or too broad, or something like that. And then you just see what fits. Am I right in assuming that's how things work in the wild? Or do people just nail it the first time out?

SAL FURINO: Oh, very rarely do people nail it the first time out. I still don't even nail it the first time out.

And that's the beauty of them. They are living, breathing things. And this is where myself and a few others collaborated in creating something called SLO DLC. That's slodlc.com. And that's a whole methodology in how to think and use and operate your SLOs. Like, I believe it starts out with initiate, then-- which is initially like, oh, hey, I'm starting a project, getting buy-in, what's the general value prop of it.

Then there's the discovery. You're actually working with an individual team and learning more about its service and starting to break down what are our customer user journeys. Where do they start? Where do they end from the customer experience, and then also from the end technical components that correspond to those users start and end bits as well?

Maybe in doing that you realize that, hey, I can't measure when a button gets pressed in the UI, but I know when I receive the request from the service. Oh, hey, maybe you declare some level of telemetry debt there until you're actually able to get that additional measurement out into the UI, or that press on that mobile app, or something, until something then renders on at the end of the screen after that whole process or pipeline of things that happen to go off and produce it.

After discovery, there is then design, which is then taking-- once we have identified what this customer user journey is, oh hey, let's start actually breaking it down and understanding what are the different ways in which we could slice this user journey up horizontally and vertically.

So when I think about horizontally, let's think about our e-commerce example from before. If we go and look at some data and realize all cart requests for my e-commerce website have five items or less and that represents 40% of our traffic, hey, we know we can achieve this result faster because there's less items, less thing for me to process in this cart.

So this is where it might be useful if you say first, oh, hey, my end-to-end customer journey for all cart checkout requests-- and I'm just making up numbers here, folks-- maybe we say, within 750 milliseconds, 99% of the time, all cart checkout requests are processed. That's our global end-to-end SLO for all our orders.

But then if you have five items or less, maybe we could be more performative. Maybe we say we could do a cart checkout request if you have five items or less in 250 milliseconds, four nines at a time. This is something that is much more expected user experience. I click, oh wait, it's done already. It's something they expect, and it's processed. Versus if you have larger items in your cart, it might take

longer.

STEVE MCGHEE: So the way that you phrase that, I thought, was really good, and it shows that you're comfortable with this stuff. Because, like, lots of times, when I talk to people who haven't done this before, they stumble over their definition of what they mean by good.

And the way that you said, I want to hit a certain time with a certain type of request for some percentage of the time, you might even say, as measured over some window by means of log analysis or metrics, blah, blah, blah. Like, that sentence in English, that's the SLO, that's the big thing. And it allows us to be really-- it's like a poem kind of. It's like a way to express what it is that we want the system to achieve.

The other thing that I think is really great about these SLOs when they're-- especially when they're expressed fully like this, like if you leave out the part about, oh, we're going to get it from logs or from metrics or whatever, it's like, they also tend to be pretty implementation agnostic.

And so, like, you can unwind or you can re-implement the whole back side of the system. And the statement can still be true. Like, sometimes I suggest people, like, write your SLOs before you do, like, a migration from the old system to the new system. And this is kind of just your assertion now. As long as you can assert the SLOs slots are still, like, valid and green, like, migration is gold-- or is good. Keep going. But who knows if people actually do that out there. So yeah, the SLO SDLC I think is great. Were there more steps that I cut you off on?

SAL FURINO: Yeah, yeah, yeah, there were more steps there. And I actually wanted to dive a little bit more into design a little bit, too.

STEVE MCGHEE: OK, good.

SAL FURINO: Because, like, we talk about slicing horizontally, like, hey, it has something-- this certain use case measure end-to-end, but then there's also vertically as well. Oh, hey, there's different components in this with it. So this is where if you know, oh, hey, this message queue is in this, and it always is problematic. Let's just measure-- let's carry this whole end-to-end objective for 50 milliseconds.

Let's assign budgets of that 750 milliseconds to each of these components. How performant would it need to be? And then give a part of that to the message queue. So we know when that's

underperforming towards this part and how it's contributing to the larger part of that reliability journey. So that's something if people wanted to get more into measuring the health of the system. Hey, this is your opportunity to go off and do that, but you still need to have that same frame of reference of what is the actual thing we're delivering to our customer or users.

And I think this is where, when we start talking about pulling in metrics and getting data, this is where, like, design and implementation go weave back and forth a little bit, because you need to instrument some metrics, you need to get some stuff together. So there's a little bit of arrows pointing to each other in that step in iteration back and forth between them until you get an SLO you actually can operate and use. And that's the next step of SLODLC, how to actually use this.

And this is where you start setting up different action policies based upon your error budget triggers. So the three error budget triggers I generally like are percent error budget remaining, error budget burn rate, and time to error budget exhaust. There are many others out there, but those are the basics you could do a lot with.

So percent error budget remaining is essentially how much error budget do I have until it's all gone percentage-wise? Error budget burn rate is how much error budget am I using right now. And then the last one, the time to exhaust, this is-- combines the percent error budget remaining and the current error budget burn rate to give you an idea of how much time you have until it's all gone, so you know how much action you have, how much time you have to take a certain amount of actions before it's gone.

So let's maybe-- I just said some theory on you, so let's just maybe weigh in an example. So let's say we're doing a service migration, right? Let's say we're moving from service A to service B and from the legacy onto the new system. And we're measuring the user journey from going from A to B. And as you're doing it, you might-- as you're going off to switch over, you might look at and say, oh, hey, we're going to this release in the next couple days. How much error budget do we have remaining? Do we have enough air budget left? Do we have enough reputation? Have we burnt any reputation with our customers or users who used this recently that we need to be concerned about making this switch now?

If you go and look at it and say, oh, hey, we have 70% of our budget remaining, you're probably good. You're probably good to go push that new release or move that switch over that feature. If you have maybe 20% or 30%, maybe you should give pause. Maybe one action you take is you delay a couple

days to regain your error budget you have. Or maybe you double on call or set up additional telemetry, or you stage your release more carefully if it's still something you have to ship and meet to that date. And this is where SLOs are a suggestion to what you should potentially do before you make that switch.

MATT SIEGLER: So it's a real measure of cushion, like, actually business-based cushion rather than from the gut here.

STEVE MCGHEE: Yeah.

SAL FURINO: Exactly. Exactly. Instead of people-- it's putting math to people's guts. And when you start, actually make that new change to that flip over to that new system, that's when you start using the error budget burn rates. Hey, how much error budget is this new thing actually using right here, right now? If you notice, oh, hey, 0.5x burn rates. Oh, a little elevated, but generally probably OK, within budget. If we start seeing error budgets burn rates at 2, 3, 4, 5, 10X, like, oh gosh. Something's going wrong. We need to flip it back. Maybe we need to roll back or do something.

And lastly, as well, the time to exhaust. If you notice stuff's been going on, but maybe you just have a slow burn effect. You're like, you see that maybe you're just hovering around like 1.5x burn rates and so forth, but you notice your time to exhaust is going off, and it's like, oh, wait, we have five minutes until all our error budget is gone. Maybe this is something you need to proactively just fall back anyway, because you know it's going to take you two or three minutes to flip over to the new service and have that propagated everywhere to your user base. So just do it proactively, just based upon how you know you're going to respond.

So this is like a lot in the operational step and actually using them and making decisions based upon these different triggers and taking different actions towards it. And I think some companies or some methodologies call these alarming or action. I think alarming is just one action-- sets of actions that people can take. There are several automated options and other things you could do. One I love is throwing a banner up on the page saying, oh, hey, some people have reported that there's been an issue with the system every time you have some reliability. So at least it starts getting ahead of some expectations. Like, oh hey, there might be a problem. Not just there normally isn't. And then the last step is a review step in the SLODLC. And that's just generally, hey, how do we take all of what we know and have and then iterate on it all together as a team, and using the framework they provide to help make better decisions about the reliability or unreliability of our services?

STEVE MCGHEE: Nice.

MATT SIEGLER: Wow, that was quite a journey. And to strain the metaphor beyond all reason of Moneyball, I'm going to throw you a pitch, a curveball that's so obvious, even you're going to hit it, you have no problem when you see this coming. So how has AI affected all of this, even the creation, the discovery, the writing, the analysis, any of this SLO creation, whatever? Just what's coming to mind? Everyone's talking about this. Everyone's talking-- thinking it's going to save us something. Have you seen this affect your work?

STEVE MCGHEE: Can I give you an easy answer, Sal? Because I know the easy one, and I want to get in front of you. And then you have to come up with a hard answer.

SAL FURINO: So I got a couple answers here, Steve.

STEVE MCGHEE: OK, OK.

SAL FURINO: If you want to go first, go for it. Go for it. It's your podcast.

STEVE MCGHEE: I think the easy answer that you're going to use is the one that was a product that you already worked on, which was, I just want to say some words in English, and then you spit out you, LLM, will spit out, like, the code for me, the SLO for me. And that's a good one. Like, I'm not saying that's bad, but that one's already done. Like, I thought it was great that if you just want to use English up front and then translate it into, like, SLI parameters and stuff like that. I think that's a good one. But beat that. You can do better than me, though. What else you got?

SAL FURINO: Oh, I think I can. Well, first off, I want to say that all content for this podcast has been 100% human-generated. I believe the only AI we use in producing this podcast-- this episode of this podcast was actually spellcheck in our preparation documents.

STEVE MCGHEE: It's true.

SAL FURINO: Besides that, everything else has been human-generated. And I just want to call that out that this podcast is all human.

STEVE MCGHEE: I even ignored the spellcheck, Sal. So, like, there you go.

SAL FURINO: There you go. Yeah, so with that in mind, I feel, hey, if you can describe your customer

journey in such a way, would that be helpful? Absolutely, yes. But I think, first, you need to have that conversation, or at least that thought of what is my customer user journey? What is the service I'm actually providing to my users and stakeholders?

And I'm not sure if an LLM can answer that for you. That's something that's inherently important to the users of your service, the humans or the other computers that call it. Eventually, you're going to get to a human that has an expectation of how well that has to perform. Hey, could that be in some docs or something? You have it trained on everything. Sure, you could go off and have it go off and find it. You could do that, yeah, but someone still had to write that doc first. Someone still had to understand what is that experience.

So something I've been thinking about, and this is kind of just an idea I want to put out into the world. And it comes up with the idea of analyzing trace data. And let's say if you have trace data for a certain product, so let's say within a namespace or a certain subset of all your trace data, we're focused on a certain product area.

And if there was a way to go and consume all that trace data and tell me what is my user journey, what are some expectations, or what is the P99 or P90 experience for these customer journeys based upon volume, based upon time, based upon some other thing of the web of interactions of all this stuff, consume this information for me and tell me what those are. I think that's something in which I could do well.

It's using a lot of data. It's a lot of statistic points and statistical points, and it's very structured in what that is. That could be a very interesting idea for people to go off and pursue and understand it. I still would heavily encourage people to check the results here in there, but that could be potentially a useful way to do it.

In terms of the other stuff and actually setting the SLO itself, I think this is where I think-- I don't think you need an LLM or AI to tell you this. Like, regular stats are fine. That's what a lot of them are based upon. Like, that's just regular math we've been doing for what, the past 50 years or so with computers. Like, the math isn't rocket surgery. It's just trying to understand what it is.

I know that statistics sometimes can be scary for people, like looking at all these different sigmas, sure, but it's still something that could be a bit useful there. And then, lastly, when we get to actions, something I would really be interested in is something called-- I've been calling like a mimic. If we think

about it-- and I actually wrote a blog post with Niall Murphy on this and a few other folks called Digital Twin. It's the idea of having an LLM ready to go and trained on your data set already. So let's say I have a basic request response service here. So I get a JSON blob, and then I give some JSON back. And if I'm able to parse and understand this JSON, what's the template and what the different values here. And you have the LLM consistently trained on, hey, if I get these results, put these things in, this is generally the values I put out. Maybe you could have it step in and augment the system when you have an availability problem.

Yeah, it's going to be slower. Yes, it's going to be the trade-off of additional latency. But could you potentially serve up a request that's maybe somewhat useful to the customer that might be potentially wrong? Is that potentially better for your user and customer base than giving nothing at all? And this is something that I think is a very interesting idea in which people can maybe try to implement in the world.

Again, it's at a high cost, though. You have to train the LLM, you have to have it ready to go in order to solve this specific use case. So I think there might be potential other things like that, in terms of making-- training up models faster or more accurately and making them not hallucinate when given proper responses there. But I'm excited to see where that goes.

STEVE MCGHEE: Cool. Well, thank you, Sal. This has been educational. No surprise there. People on the internet would like to hear more from you. Where should they look?

SAL FURINO: Check me out on LinkedIn. I'm S. Farina on most socials. You could probably find me in most places like that. I'd also like to plug. I run like-- in New York City, where I'm based, I run a platform engineering meetup as well. So check that out as well. If you're local and you would like to go off and see it.

And if you're really interested in Bloomberg, check us out. We do a lot of really cool tech stuff in the finance world. We have a lot of very interesting problems that people solved, and we worked together to go off and do it. And we have a bunch of engineers here. We have over 9,000 engineers. And I've always felt like Bloomberg is kind of like this hidden tech company out there that might not make it into all the fancy acronyms, but we do cool stuff. So come check us out.

STEVE MCGHEE: All right. Thanks, Sal. Thank you as always, Matt. And--

MATT SIEGLER: Thank you, Steve.

Google SRE Prodcast

STEVE MCGHEE: --I think that's it. Thanks a lot, guys. Until next time.

SAL FURINO: Bye, so long. Let's SLO, everybody.

[MUSIC PLAYING]

—

[JAVI BELTRAN, "TELEBOT"]

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

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