Customer-Centric Monitoring

Silvia Esparrachiari talks about the challenges of monitoring and the importance of understanding your users.

Viv: Hi everyone. Welcome to the SRE Prodcast. This is a limited series where we're discussing concepts from the SRE book with experts around Google. I'm Viv, and I'm your host today. And I am here with MP.

MP: Hello everyone.

Viv: And today, we'll be talking about monitoring with our guest Sylvia.

Sylvia: Hello.

Viv: Hi! Thanks for being here. Do you want to tell us a little bit about yourself?

Sylvia: Yes, for sure. Thanks for the invite, first of all. Hi, I'm Sylvia, and I am the SRE UTL (SRE Uber Tech Lead) for a sorted collection of services in AMP—that stands for Application Modernization Platform. I've been at Google for over ten years, in SRE for about seven years… and monitoring is the center of SRE work—not just for the metrics, but also observability and understanding of how our services are behaving, and addressing our customers' needs.

Viv: Yeah. Cool, thanks. So the SRE book has some notes on monitoring that are more just kind of general definitions to keep in mind as you build your systems—there are some general tips—but I'm wondering what you would define

as monitoring if someone were to say, "Hey, we need to monitor our systems," right? What does that mean?

Sylvia: So the first question is, what's your business goal, right? Traditional thinking around monitoring often says that we should consider error rates or latency, but these are all measurements. They are not targets. They're not goals. So monitoring is nothing if you do not have a goal. It's actually not encouraged nowadays to have too much data, right? 'Cause it can obfuscate the objective information that you are actually looking for. So, first thing when you think about monitoring is, what are your business goals and what kind of information you can extract from your services? [You] should make a decision about whether or not you are meeting those business goals or what kind of insight you are looking for to take the next step in your market segment.

MP: So one of the words you've used is observability. And another word that I've heard around the industry is telemetry. How are these the same or different from monitoring?

Sylvia: I usually tend to think of telemetry as one aspect of observability—so for me, telemetry is the time series for whatever you are trying to observe. And observability means, can you extract this data from your telemetry, right? Again, you may have tons of data in your telemetry, but they say nothing about what you are actually trying to observe.

Viv: Yeah, I like it. It's an interesting approach. So do monitoring systems often account for a number of goals, or is it—I guess this is kind of easy if you have one clear business goal that everything kind of works towards. What if there are a number of goals—does that mean it's a monitoring system that accounts for all of them? What does complexity look like as systems become complex or goals become complex?

Sylvia: Of course, that's an awesome question. 'Cause especially at Google, we are often handling systems that are too complex for a single person to understand, and they often share different goals from different parties, right? So I bet that the goals from business folks, they are interpreted slightly differently

from what engineering usually thinks about their own services. So observability may include all these different perspectives. In engineering, usually, we tend to look at the services as black box, right? And engineers, on a first moment, they will try to observe things like availability, which is commonly defined as the error ratio on the service. So given a certain timeframe, you want your error ratio to be below a certain percentage, and that's called availability. But the truth is, the next question you have to ask is, who is observing those errors? Are they observed by a single user? Is this user a critical user? Is this user intentionally generating errors—for some reason, that's what they actually want? I can give you a quick example.

MP: I would actually love to hear that.

Sylvia: Yeah. So here's the thing. There was a system that would push data into a database. And then, in the second moment, they would intentionally send the same data and count the number of duplicated errors. So data availability from the point of view of that user availability was always 50%. 'Cause in the first moment they would inject the data. The second moment they would try to inject the data again. And in the best scenario, they would get errors, all errors in the second attempt. So if you consider the whole timeframe of their workflow, availability is 50%. So these are the questions that we have to ask ourselves when we look at telemetry: what is this data telling us? If you look at only the general broad view, you may get an answer that makes absolutely no sense and definitely doesn't describe the user experience. So broad data by itself means nothing. You have to understand your users, you have to understand your use cases, and you have to accommodate for user creativity, which is very hard.

Viv: Users are certainly inventive. I know I've seen that a lot as an SRE handling something that is more user-facing. Lots of weird things come up! So, I guess, speaking about users when we talk about business goals, are those typically internal for us—you know, we want our system like this? Or is it more of a user approach where it's: we want our users to see X, Y, and Z? How much of monitoring is something that users notice versus something that your team or your company might want for itself?

Sylvia: That's a great question. Observability may happen at different levels, right? So one important level is the one that we believe is closest to the user experience. That level is defined by ourselves. Actually, the closest level of observability is for you to sit right next to a user and observe them using the system. But that's a very hard strategy to escalate, right? You cannot sit next to your users and collect data from them. So we tend to automate this data collection, either via instrumentation of our client libraries, things that run on the browser, and send those back to telemetry about the usage patterns, usage sequence. So, oh, I click this button, and then next, I click this other one. So you can actually draw the most common interactions for users in a browser. Another layer that is often associated with user experience is at the very API level—the public API that Google exposes to users. In that case, you instrument the calls to the API. It's a little bit harder to trace the interaction across APIs 'cause they're not necessarily related. But this is a common level for systems in infrastructure, for instance, 'cause we really don't have any piece of logic that runs on the browser. We only go up to the API level. Does that answer your question?

Viv: I think so, a little bit.

MP: Yeah. I know a way that this has been in my experience. I tend to think of these as like client-side monitoring versus server-side monitoring are the words I use in my head usually.

Sylvia: Definitely. If you have telemetry coming from a client library or an agent on the browser or agent somewhere, that's usually considered client monitoring. Server monitoring might be misleading 'cause you can still account for different workflows on the server being processed by the server, while "server monitoring" often brings to mind the general performance of that black box. So server monitoring is often associated with black box monitoring, but you can do *workflow* monitoring on the server. And that's the actual challenge in TI/GCP (Technical Infrastructure/Google Cloud Platform) at the moment.

MP: "Workflow monitoring" is a term I have not heard before.

Sylvia: I just invented it. Just to give you something different.

Viv: I love it. You heard it here, first. So you heard it here first! Now we can bring that back to all our teams and talk about workflow monitoring on the server for the server.

Sylvia: For the customer.

Viv: For the customer.

Sylvia: On the server for the customer, yes.

Viv: Yeah. I know—this isn't necessarily a question, just a comment—I know one of the things that our team thinks about often is, you know, if you have these different pieces on client-side, workflow—more server-side, if you wanna say that—how do you connect those to get a full picture of what's going on versus keeping those separate? And how can you apply both of those when you're looking at different incidents or just generally looking at long-term trends?

Sylvia: That's the $1 million question. So yeah, when should we use observability? When should we apply observability? For me, there are two key moments when you need observability. One, as you mentioned, is during outages and incidents: you want to know how that outage is impacting your users. And that's a critical detail information that you must have at the time of the outage. 'Cause you must provide a response to your customer saying, "hey yes, we are aware. We know that is happening right here. We predict it's gonna be fixed by then." There is another moment when you should take a look at observability, which is planning—strategic planning. And you don't have to do that at the moment. You actually can look back in the past and observe trends, look for evidence of new usage or new patterns, or even areas where your service may not be actually delivering what the user wants. You can actually find this—like the case that I gave before when users would push new data, and the way for them to verify that the data was actually stored was to send it again and get errors. Maybe we should just send them a confirmation that the data was stored. They

need to confirm that the data is stored, so they do that creative second push to make that verification. So observability can help you with that kind of insight.

Viv: I was just thinking, you know, in an ideal situation, when do you start thinking about monitoring when you're building a new service or system? Or I guess in your experience— I know monitoring is a core component of what you focus on. Is that something that you then apply? Like, you bring your approach to existing systems and involve that monitoring? Or when do we think about it when it doesn't exist yet? Or when do we think about when to update it?

Sylvia: So during any development process, there's one step which is verification, right? So you go to your users, ask them what they want, and then you have a fail fast iteration, build the model, MVP, check with them if that's what they want, implement it, invite your users to test your thing, and then you push to prod, right? Even after pushing this new feature to prod, you still have to check if the feature is actually working as intended for everybody. And if it's not working, what's the subset of users that is not having the good experience that you designed? So monitoring serves that purpose— the verification purpose— at all times. So the beauty of monitoring is it's also an automation. So you don't have to ask your customers all the time, "hey, is it working? Is it working?" No, you have the monitoring there. You can check this out. So once you think about monitoring, think about what are the verification requirements for your feature or your product and integrate the indicators, the main indicators, the telemetry that you need to verify those conditions into the design plan, so when you launch the feature, it goes with all the necessary monitoring that you need to further verify if it's a success or a failure.

Viv: We talked about being very intentional in what data you're monitoring. How do you know that that data is accurate? Or rather, how can you trust that data?

Sylvia: That's an excellent question. Yeah. Problems with the monitoring data can be very deceptive and hard to actually notice. I'm trying to think if I ever saw that kind of example. I think the most recent example I can think of was something that was always kind of on. So it was always zero or always one. And we could only verify that there was a problem with the monitoring data because of

secondary information or a secondary source of telemetry. For instance, our monitoring was saying, "oh, user names were fine. We're getting zero errors when registering a new user in the system." But then, when we looked at the logs, we could clearly see the error logs being reported there. So it's mostly a matter of feeling, right? You feel that there should be a variation in this indicator somehow. Why is this always 100%? It doesn't make sense. So, always look at your telemetry with a little bit of criticism and don't trust a single indicator, right? Always build a robust collection of indicators that can give you the whole picture with higher confidence.

Viv: Okay. So what I'm kind of hearing is: trust your data by not trusting your data, but understanding your service, potentially. Not that you shouldn't trust your data, but just to be critical about it.

Sylvia: You trust the overall behavior of your data. You don't trust the little piece individually.

Viv: Sure. And I think it's interesting to point out that there does need to be a robust means of monitoring, right? I don't know if that means that there is overlap in what's being looked at, but if you only have one thing to look at it, then maybe you don't know when it's failing, right? There's only one aspect you can kind of approach it from.

Sylvia: There is another way to validate that your monitoring is actually giving you the data that you expect. It's similar to unit tests, right? So you build the test, and then you break it, and then you see it fail. So you know the test is actually reacting to the bad condition. So you can do the same thing with monitoring. If you have an indicator that says when your system is down, you may want to bring something down to verify, probably not in production. Try this in tests, right, not in the prod environment. And verify that your alert actually goes off or that your graphs actually go down. So verify that your intention to monitor is actually what is in place.

MP: Now what about, like, meta monitoring?

Viv: Monitoring your monitoring?

MP: Yeah. Like, the "who watches the watchmen" question.

Sylvia: So that's the beauty of options, right. So if you don't trust your monitoring solution, you could always have two solutions in place, right? So go around the market, choose two, and put them in place. If you offer a monitoring solution, definitely do not use your own solution to monitor yourself. Again, you may do that with a secondary separated system, but also have a different solution to make sure that the most critical indicators for your system do not have a dependency on yourself, right? So avoid cyclic dependencies when you run a monitoring solution and avoid cyclic dependencies for everything basically.

Viv: So I like a lot of the principles that you've talked about—definitely good food for thought. I'm wondering, where are we in those principles? Does current monitoring—do the systems that you have currently follow all of those things? Or where is the room for improvement? Have we not reached that point yet? Like, is this ideal and we're moving towards it, or is this where we are and then there's room to keep going?

Sylvia: So observability has evolved, at least at Google, in the past ten years where we used to look at our services and interpret the information we got as serving an abstract user. So all these queries go to the user. It's a single entity that represents us all. We not often would look at the long tail of performance. So if we have five 9s of availability, we are happy, right? 'Cause that represents that the generic user is also happy 'cause they observe five-nines of availability or latency or whatever indicator we have. Now, the world has changed a bit—actually, a lot. And we are starting to account for different profiles, different users. So we no longer assume that five 9s of availability represent the overall experience for all the customers. And we take a deeper look into the long tail of performance, and that changes the way that we measure things. 'Cause before we didn't need an identifier for the users; we could easily drop that on the floor. Not even an identifier for the users—we mentioned before workflows, the workflows are actually more important than the users individually 'cause different users can perform the same workflow in our systems. So we want to make sure

that all the workflows that are served by our systems are well-performing, and they might diverge in their computational requirements, right? So one workflow may require lower latency, but not so much accuracy. So we have to account for that kind of workflow. Another one may require exceptional accuracy, but they can tolerate a day of stallness or something like that.

MP: I did have a question I wanted to ask earlier. When we were talking about telemetry, and the idea of user-centric telemetry popped into my head— that sounds really expensive in terms of data to try to have telemetry on a per-user basis.

Sylvia: Yes, you are correct. So consider even just 1 million users. You would have one line on the graph for each one of these users. That graph will never render. And also, it would not be so useful, right? 'Cause, we are humans. We can barely cope with ten lines, [let alone] 1 million. That's where the concept of workflow comes in, or profile, or unity of aggregation. Anything that brings us between 1 million and just one, but we can still extract actionable information from it. So we don't need to understand each individual user, but we can understand the classes of users, their workflows. Or even if the data is not necessarily centered on the user, we may choose aggregation that is based on whatever we are serving. So let's say we are serving a photo album for our users. So we may consider checking availability and latency per album or per photo. But I think, in that case, the album would be more interesting so we can see how many albums are healthy, right? We don't need to know exactly which user is looking at the album, but we can count how many albums are healthy or are rendered within a second or so. And even then, if the number of albums is too high, we can aggregate them and say, oh, how are the albums with five photos—up to five photos—behaving? What's their performance? What's the performance of albums between 5 and 20 photos, and so forth?

Viv: Makes sense. So I guess in that case, to continue your example, say most users have relatively small photo albums. Say everybody has under a hundred. But then there are some users who have tons and tons of photos in their albums and maybe that's the long tail where they don't perform as well. Is the point of monitoring to—maybe this is kind of a rhetorical question, but—I'm curious

whether the monitoring serves more to just make sure that you keep tabs on that specific workflow or profile of users? Just, for instance, to know if the system is down for them versus for everybody, or is it more of a way to improve on the system for those particular people because your service may tend to only account for the average person? Does that make sense?

Sylvia: Yes. And that's exactly the point, right. You can get direct insight on where your service is lacking, and you can take an objective decision on whether or not to make your service perform better. So let's say 99% of our users have photo albums under five photos or so. Are we happy with that number? Do we have a customer that is really engaged in having very large photos? Do we feel like we need to satisfy that use case? If the answer is no, we can even disregard that particular class of information, right. But at least we know we are not blind to that.

Viv: Yeah. That makes sense. That was gonna be one of my questions as well. Kind of like, when do we care? Because I could totally see this as being a question of why do we want to spend engineering time on building monitoring for all these particular workflows if the numbers of people in workflows B through Z are so much smaller than A? Maybe that's more of a business question, but it definitely seems like something that could be tricky.

Sylvia: Yeah. And we can also leverage this information to the business folks and help them decide on what's the next step.

MP: Well, I feel like to some degree we've danced around a buzzword buzz abbreviation with talking about a user-focused monitoring, which I guess those would be CUIs and CUJs.

Viv: Critical user interactions and critical user journeys.

MP: Yeah. I've heard those terms. Do those fit into this picture? It sounds like they would fit into this picture.

Sylvia: Yes. The critical user journeys, which is the most reasonable aggregation of data to talk about, [are] usually user interaction like click the button, enter text, submit form. These are too small for us to take any action unless there is a specific problem with the interaction. So, let's say one interaction that was interesting: there was a button in a UI that required a query to an ACL system—so an authorization system—to decide whether or not the button should render, and the whole UI would hang waiting for that authorization because the logic in the JavaScript required to know whether or not that button would show up so they could build a "hold on." Needless to say, there were several users that were impacted by random outages or just high lag in this authorization system, which was not designed for that kind of authorization—like, constant authorization just to render a UI. So, in that case, the click on the button or just the sheer visualization of the button was a problem to the user. That's usually very rare. Just to let it [be] noted here, the solution for that is, yeah, render the button anyway, and if the user clicks the button, then you decide what should happen then. You can even show an authorization error or so, but do not block the rendering of the UI based on a single authorization. Just so folks know about this. In general, you are after an experience, right? 'Cause the experience accounts more for the performance than a single event. And in the same way that you can think of someone sitting in front of a computer and trying to send an email—click here, alter email, send button—the whole set of actions may trigger a collection of calls after the requests reach the API.

So [the] user is doing their thing on the UI. And then, these trigger HTTP calls to the API. These HTTP calls are converted into internal requests to the backend of that API, and that backend, in its turn, sends a bunch of other requests to adjacent systems. So all these three, all this stack of calls, will impact the user journey, right? Not just the call to the API, not just the click on the button. Everything. So when something goes wrong it's really difficult oftentimes to understand what went wrong. 'Cause, there was no problem with the backend to the API. There was no problem with the browser. JavaScript was fine. So what went wrong? You want to know: what were the calls that can impact that user experience? One way of doing that is to build a breadcrumb trail across your systems. So every time you get a request from the user, you annotate it with

individual ID, and you propagate that ID across all the other response requests. This is extremely expensive. So you may want to try to do this via some kind of sampling, or if you know that you have a problem with a specific journey, then you turn this on, you apply this for that specific journey.

Viv: Yeah. I definitely see why user journeys are a good thing to focus on and why maybe that's the, perhaps—I don't wanna say new frontier, but kind of the new thing for monitoring, to lean into what the experience is like. This goes along with a lot of the things we generally talked about in this episode. MP, do you have other questions you want to chat about before we wrap up?

MP: I think I'm all set. I think that was a really fascinating discussion.

Viv: Cool.

MP: I really enjoyed that.

Viv: I'm really glad you came to talk to us about your ideas of monitoring, where it's going, telemetry, observability, workflow monitoring— lots of great things in this episode. So thank you so much for coming and having this chat with us.

Sylvia: Yeah, it was super fun. Thank you so much for the invite.

Viv: Thanks so much for tuning into the SRE Prodcast, and come back next time for Episode 3 on alerting.