R: I've started the recording and I will share my screen with you. First of all, are you able to see my screen?
P: Yes.
R: You should be able to see two windows. So first of all, the research objectives.
I'm conducting a study on software product quality, so specifically the quality of products and not of code. For that we've chosen a model from science and it's one of the ISO standards, ISO 25010, and the standard is a hierarchical model that basically says that a good software product follows eight quality characteristics, which we will discuss later.
So during the first part of my research, I conducted discussion groups and in these discussion groups, I asked the participants how they would measure this abstract concept. So you can see on the screen reliability is one of them. So I asked the participants to have a discussion on how you could possibly collect data to say something about the reliability of a software product. So we found measures of reliability. Those are 15.
We discussed probably two of them during the session, two quality characteristics, and now we want to enrich the results from the discussion groups by adding a source. So where can you possibly find this data value? So that could be a repository, that could be maybe on JIRA or Rally or whatever is used for project management, could be in logs, could be in dashboards. So that's the question. And we'll also discuss the measures based on two dimensions.So the first one is the difficulty of obtaining data. So is the data readily available? Or do we require some pre-processing and cleaning to make the measurement available? Or is it unavailable and extensive effort is needed to make data available? So for that a skill is used that usually you would say is low when minimal effort is required and it would take about one hour to one day of work to implement it. And for moderate that would be between one day and a week. And high is data being unavailable and you can only obtain it after over a week of effort. And the same is done on the dimension of technical expertise. So for low, that would be that basic technical skills are required.
You wouldn't really require any specific knowledge on the tool. You can maybe just read the measurements from a dashboard or find it anywhere without being
really technical skills in some aspects. And then we have moderate, which is that you do require some technical skills. But most software developers working with the tool would have that knowledge readily available. And then high would be really advanced and in-depth knowledge on a specific technology are needed for setting up the measurements.
Is is clear for you?
P: Yeah.
R: Okay. Also important to state that during this interview we will not discuss the quality of the proposed metrics. So maybe you find some metrics that you do not agree with.
We will not discuss that during this session as that will be evaluated during different parts. So for each measurement it will be a pretty structured format. So it kind of looks as if you're filling in a questionnaire, but then you can provide some more insights into why you would say, for instance, that measuring uptime would be low or moderate difficulty obtaining data. Right?
P: Yeah.
R: Okay. So let's move on to the measurements. Can you, is the text big enough for you?
P: Yes.
R: Okay. So first of all, the first quality characteristic we will discuss is reliability, as previously stated. So the standard defines reliability as the degree to which a software product performs specified functions under specified conditions for a specified period of time. So during one of the focus groups, the participants agreed that reliability could be measured

by measuring uptime divided by the SLA of uptime. And now we'll answer the questions. Where can we possibly find this value of uptime? How difficult would it be to obtain this value? And when setting up the measurements, how much technical expertise required? And it is important that we're looking at the subject from your context.
So the work you're doing within <name organisation> and also the technologies that you're using, because obviously this will be different when you're working in a serverless environment versus in an on-premise environment. So here the question, when measuring uptime and a service level agreement, where could we possibly find this information?
P: In the context of the work we do in <name team>, it's interesting because we kind of don't monitor uptime because most of our products are being consumed by different teams.
For the things we do run, which the technology is already changing, we have external dashboards or external vendors that is kind of reliable on uptime. For example, one of this would be the usage of Hive for the federated graphs, which means if they go down, so the goal of the federated graph or Hive in a sense is when you request information, it kind of directs you towards what you're all you should fix information
From. So I want the information for this structure, it says, okay, cool, you can get it from this URL. If they go down, then we don't get that information, so you can't resolve the URL.
R: So you're dependent on the uptime of the external tool used in this case, Hive?
P: Yes, then I would say the biggest secondary part of it is because we don't work a lot with actual server-based things in the sense of EC2, most of our things are serverless Based. I would say we have containers that are getting the measurement of uptime in regards to not implementing anything for it. I would say our measurement of uptime, funny enough, I'm not even sure if it's a bad thing to say, is the reliability of the AWS vendor. For example, if our lambdas would fall over, our reliability is to expect that AWS or at least have some systems that runs and make sure, okay, cool, things actually stays running. The last point, I would say the lowest point of reliability for uptime is, from my experience, what I've seen within our team itself, I don't think we have any cross-region things, but in my experience, where I would add a flavor to the AWS things, is to, you're still kind of vendor locked to AWS, so if AWS is the single point of failure, if they fail, then it just fails, but to prevent this single line of failure within this single vendor, you can create multiple by having multiple regions, so for example, if something would fall over, traffic would be redirected through there, which means technically speaking, your uptime would have a higher uptime, let's say 99.9% for that, 0.01% it takes to direct to the new.
R: You're saying that you're dependent on the reliability of your vendor. How difficult would it be to obtain the data for the reliability?
P: To physically see their reliability in general, because the thing is, it all depends on the vendor itself. For AWS, I've never actually done the research to find the reliability for them. From personal experience, I've had various times where deployments fail throughout the December time where they roll out their new products, but there is no mention of that something would happen like that, especially a few years ago with containerized things. Finding the information on their downtime, I'm pretty sure they would be very specific around that. They would say, cool, this is our downtime, this is what you can expect, this is how you need to handle situations, this is how you need to route data through multiple regions.
R: Do you find this information in the docs of the vendor?
P: Sure, it might be also more public facing things, which is I would say docs refers very much more to if you would say almost

developers jump into it, but for normal people doing normal research like managers or people owning companies, I think it might be more of a friendly public facing information also to get that foot in the door to make them feel more comfortable. As this is publicly accessible,
R: you would say that there wouldn't really be an effort into obtaining the data and no technical expertise would be required?
P: No, not for uptime.
R: That makes sense. Okay, let's move on to the second one.
P: The technical expertise required, is that based on research for uptime or the implementation of finding like getting the end result of uptime? Because for example, I would say the technical expertise, I would not put it low in the sense of if it is just to go see what is the uptime, sure, but to actually make sure the uptime actually conforms to the 99.9%, they would probably have some guidelines and rules to apply your technology.
R: Exactly, this is specifically to measure the uptime.
P: Okay, yeah, just to measure it, do it below, yeah.
R: Yeah, so not to implement any measures that improves the uptime, just to make sure that the data on current uptime is available and that applies to all of the metrics that we'll discuss. So next one will be a mean time between failures. We will not discuss how to improve this value. We'll just look into where can we possibly find the information, find this point of data and how difficult would it be to obtain this data? So maybe you'd have to implement some logging to obtain the value and then how difficult would it be to set that up? That would be the technical expertise. Does it make sense?
P: It depends what you mean by failures though, is it because you get multiple levels of failures. It is developer failures, it's company failures, it's vendor failures.
Yeah, with failures the participants meant usually a functionality not working. So let's say you have a laptop function that needs to process an event every hour. When it fails, one of your functionalities is down and that's considered a failure. Okay, so when the actual lambda fails. Yeah, exactly, yeah. Yeah, I mean you kind of have two parts to that because I see that mean time between failures as the lambda as a service and the lambda like the point to where it actually reports failures. Yeah, so as a service personally from my personal experience again to the same vendor AWS when things usually go down or at least a service doesn't function as what it should. From their side the fastest or the slowest I've seen things and I'm gonna put the measurement as the slowest because the fastest would always be like relatively super fast. It has been 24 hours before something was fixed and you could continue there so the gap would be 24 hours. Personal side of things it all depends on the amount of time that was used to implement the functionality to actually when was the failure reported, how was the failure reported and when was picked up and that is all dependent on you need some technical experience there you need to for example if we take a sample size again and we take lambda cool lambda can log its errors but it's just basically a log sitting there it doesn't mean anything so you would potentially want to set it up where you have services for example queue or anything that actually a message is being pushed in or at least the log is being read and then a sufficient email or slack bot or whatever might be required to actually allow tending to this failure and then when a failure does occur usually it all depends on where it sits if it's production side or fixed type of things R: you're looking at probably a 48 hour window around fixing that but here we're just looking at not fixing the failure but just looking at how can we obtain the time between failures do we have to implement logging can we just look at the dashboards to do that

P: Yeah it would be it would be the start that i said yeah so it would be everything for the last sentence that you would you can physically go look at logs i mean it's always readily available so yeah it depends on the the it depends on the person that's looking at the problem that's the thing it's a meantime between failures it's like it's too many variables um i mean it's the speed of when people go look at the errors it is if it's implemented to send out an email and all depends on when people look at the emails when errors occur i yeah i don't i can't think of it like a straightforward answer towards meantime between failures yeah it's it's such a broad it's it's very dependent on so on how you define technical expertise required to set something up to see failures yeah i would say can be low but to correctly handle failures and allow a company or the the quality of a product to be better i would say it is it should not be low it should be moderate yeah but here we're just looking at at finding the measurement and not doing anything or not doing any interpretation with the measurement so that would mean within this context it would be low but if you want yeah but there is also it's there's also it all depends we have such a broad system of of technology for example let's say there is a let's say there's a message on a queue which is being consumed by lambda and the lambda takes the message but does not for example acknowledge
that the message has been consumed right the queue would take this message and make it visible again and it will happen over and over until it goes to a dead letter queue if it is set up so that is potentially error which means your queue is going to get flooded with messages slowly but surely and they will slowly but surely start to disappear which is a problem it's more cost but there is no visibility on on that is going wrong and yeah and that is kind of a failure of the lambda to to acknowledge that correctly so it all depends on where you're looking in the field for technical expertise required yeah if you're just kind of like looking at the base products can you can you if you go look at the queue just stare at the queue and see the numbers slowly but surely going up yes you can i mean yes no technical expertise required so exactly so it depends on the implementation it depends on yeah i would start to handle yeah okay but yeah that's uh the time between failures is a very loose term actually
R: exactly yeah i think most of these are so that's why we're really i think you're doing a very good job at describing your context on what this would look like within your context yeah so we're trying to find more background information on further specifying these these measures so i think you're you're providing very good information let's try to cap the discussion on two minutes max per measure because else we won't be able to discuss them all okay the next one would be the amount of data lost on a failure and here again let's take the same interpretation as failure as we did for the for the last one only we're now looking at the data loss
P: um usually 100 uh unless a developer uh set it up in a way where data is captured and can be can be rerun or reread yeah so for yeah if uh if if you have sufficient enough or good enough architecture design so and then if you have a queue then the main question is and then if you have a queue then the main question is how do we uh how do we how can we measure data loss where where can that information information be found uh that's difficult though um usually where the data if the data comes from a source yeah um you can you can contact that source to get the information again so you can visibly then kind of see what information was not uh passed through the system but if your source is out in the wild for example the public using it um there you can't um there if you don't have if you don't have the correct solutions attached to actually um know what you received then it's just kind of lost in the the bus like yeah yeah um technical expertise required and difficulty of obtaining data difficulty this is definitely not low um you're going to be stuck by whatever or whoever should have provided the data yeah um so it's going to be like you maybe send a

mail you have to wait for a response more data possibly has to come into the system it is just can be a little bit chaotic um yeah the the thing would be yeah like to like to maybe fix the problem would be a low but to actually get the yes the amount of data lost i would say moderate the technical expertise required to actually try and find the amount of data lost i would say no uh because you don't feel you don't need technical expertise to know where data come from i need to process it
R: Yeah okay i understand uh let's move on to the next one and i think we're doing we're doing a good job now i'm capping it to two minutes so we can get everything done uh so the next one provided is someone proposed that you could measure reliability through the one percent downtime duration so uh whenever one of your surfaces or one of the dependent services is down what is the duration in time of the one percent longest downtime so basically the worst that can happen situation so then then here the question would be how how do we measure downtime where can we find this this time that the system is down and what what can we set up to to perform this measurement
P: Wait isn't uh what this feels like a repeat from the question at the top i don't think i understand fully the questions at the very top uh the uptime
R: Yeah well the answer can be the same thing only here we're interested in uh the one percent of worst downtime so uh let's say usually once a week you have a downtime of maybe maybe an hour and then uh suddenly once a year your system is down for an entire 48 hours that would be the one percent downtime so the worst downtime you could possibly experience
P: So what do you mean by how do you measure it because uh the worst downtime would yeah so where do you want to measure the impact of the downtime or i would
R: Yeah here we're interested in the time so the time the the measure would be the 48 hours where can we find the information for the 48 hours is basically the question but like does that make sense so like what went wrong then for example this one percent downtime or would we be able to identify that this is the one percent downtime or exactly yeah it would be would we be able to identify it and then if we identified it we need to have a number as output so the one percent downtime would be 48 hours maybe or maybe it would be 30 hours or maybe it would be 20 hours
P: Um yeah you know downtime usually it's just kind of uh uh yeah like it all depends on how important the systems are who looked at it who's monitoring things in regards to it if it is a a smaller system that goes down people don't generally have an X amount of time around it you just know it went down for then um to identify if this is like yeah like i don't know how to answer the one percent downtime thing because like you have if you have 10 down times or 20 down times a year i've never seen a company identify: cool this is our biggest down time so to even determine this is our one percenter is going to be probably just a guess from someone that was around learning the time of all the other down times yeah if you have sufficient um logs or uh at least um well not i'm talking about logs like people that actually write logs um that that log okay cool we went down turning these times um and afterwards you have a you dissect what went wrong you can kind of figure out cool this this is the impact of uh what happened during this downtime yeah um but the the like the difficulty of obtaining data and the technical expertise to require for the one percent downtime duration i don't i can't answer that for you um because like it's again it's such an open-ended question is it the difficulty of obtaining data for is this the one percent down the biggest one percent downtime is it obtaining data about like what happened then in the bigger the one percent

downtime um how big is the one percent that's like it's it's too open-ended for me to tell you what is the difficulty or technical expertise required on one thing i understand yeah
R: Well that's that's also an answer so that's that's good as well um okay next up we're looking at uptime again uh but here we're throwing an extra dimension on it and we're looking at the uptime per functionality so let's say uh let's provide an example you have 10 10 laptop functions and here for each laptop function we want to provide we want to measure the uptime so then again where can we measure this information and how difficult would it be to obtain the information
P: uh easy uh you can create dashboards that you uh that watches your lambdas and uh you can from there uh show metrics based on successes or error rates and based on that you can determine what is what would be a sufficient uptime or downtime and you can probably uh determine that by uh all the bends on your services you're looking at but in my experience you can take a lambdas average running time you can take the amount of errors uh when it eroded out and you can determine how long it has been down for yeah so then that you can get an impact of your personal uptime that it should be okay um so that would be then the difficulty of obtaining it and the technical expertise would be uh low but the technical expertise uh you need to implement the correct dashboards to see it so that would i would say moderate so moderate to make it slow to obtain the data yeah okay then again R: we have a another one that was proposed i have no clue what my keyboard is doing it's completely spacing out um we then again have the um the discussion on the downtime um is that not applicable anymore as you think it's it's too vague uh if some system is down is it close to maximum down for sure exactly so so what what they mean here is how does a time relate to the maximum downtime that a system can be so let's say the distribution will be different if you have 40 outages of one minute and your maximum would be an hour or if you have 20 outages of one hour when your maximum is also one hour then more more of your downtime is close to your maximum downtime
P: I feel that a measurement of um a measurement of downtime is almost a so a measurement of uptime is you saying cool your thing is either running or it's not uh but i feel a measurement towards downtime um is more about the um impact of when a downtime happened so let's say for example you have two lambdas they both turn off for five minutes yeah um but the one that turned off as a bigger impact of um potentially having schedules that trigger different things in butterfly effect so a a downtime can be measured by the impact of um of what the actual service kind of caused so you can get a physical amount
of downtime by doing exactly the same thing i mentioned with the uptime function exactly
it's going to be the inverse uh but you don't know what's the impact of the downtime at all yeah so uh yeah um i could i would say the same dashboard thing if you want uh to look at downtime if uh if it's part of the one percent downtime thing it's again difficult answer to yeah question to get because yeah but now this is not talking about the one percent you'd propose you could use dashboards to uh to set this up and really graph graph the downtime over time um yeah um to yeah to kind of see it uh to handle it i would obviously have different routes like uh emails and snack bots and things like that but to visually see the actual downtime i would exactly you were just looking at seeing seeing it yet and not doing it doing anything about it yeah so if we're if we're using a similar dashboarding solution um
R: What would be the difficulty of obtaining data and technical expertise required
P: it would be would that be the same or exactly the same so all my perspectives i'm doing based on uh cloudwatch dashboards which uh has accessibility to all the aw services which i'm placing all this on

R: yeah okay cool uh the next one is um considered within uh within a group that specifically specified it in an event-driven architecture so they say that reliability can be measured as the number of failed events divided by the number of successful events so to really know the number of field events and the number of successful events what do we need to set up um would you say the reliability is based on number of failed events versus number of success movements
P: What is equal to a failed event so let's say a failed event something failed and it goes on a dead letter queue uh and it's being reprocessed and the second process or the third cycle is able to successfully process that message yeah is it then seen as a failure or a success and that's
R: That will be seen as a failure in this case yes which is something feels and then when it's when it's when it is then sequentially processed successfully then it will be uh
then the number of successful events will be incremented with one that's what the participants said during during interview
P: interesting okay i feel uh the title the that's that this is going to give some uh invalid uh data um because your data set if you pass in five you still sit for five at the end between five equals five is 100 percent success right even though two might have bounced and then be reprocessed but um in regards to the vendor architecture um the actual just looking at the statistics of the amount of fail versus amount of success events is easy uh if it's event driven or most event driven applications or at least interfaces has um already built dashboards especially in the aws environment yeah that shows you exactly what uh what uh how many successes there is how many fails um yeah and um yeah i mean without if you
don't jump in and fill their data queues and things like that um your technical expertise is also low um but i'm curious if my answers to your uh to all your questions is a bit
i don't know what the word is bias because i'm basing everything in aws aws this entire job is
R: yeah that's no that's that's completely that's completely okay that's that's a consideration that we've made um okay because like if i would be implementing some of these we'll be implementing some of these dashboards and that will also be our dashboards or measures rather and that will also be in an aws environment so that's why for these interviews we've specifically chosen people who are working in an aws environment so it will fit the case that will be later chosen okay yeah because otherwise it would be it would be a too difficult question to answer because this would be uh very different to measure in a in an on-premise environment as opposed to in the cloud then you'd probably have to set up much more logging so all of these both the source the difficulty of obtaining data and the technical expertise would then be basically very different answer so that's why everything is good that we're looking in the aws context
P: Yes so the max downtime and the uptime one at the top both that says moderate i'll take them both down to low if you're not physically creating a dashboard because you can
go to the lambda you can see the lambda file based on the lambda statistics because aws does it for you
R: Exactly yeah yeah i think we're getting the right uh the right hang of it so let's move on to the next one and that's the amount of requests sent to your service in regards to the service level agreement of requests and here the participants noted that if a system is very close to the maximum amount of requests it could handle or that is agreed to on the product it's more prone to reliability issues so again it's not about whether you agree or not with this measure it's about where can we find this information so how do we how can we find the number of requests and how difficult would it be to obtain the information

P: I have never ran into this issue or researched into this so i can't give you a viable answer on that yeah
R: Okay um so then we'll discuss the uh the number of backup options so when your system goes down how many how many um how many solutions do you have in place to to back it up without it failing so what what would it look like in your context
P: Yeah in my context uh uh aws itself is a reliable source but then you have a cross region um uh and a multi multiple availability zones that you kind of use to prevent downtime so um the difficulty of obtaining the data i don't know yeah how you would obtain the data i mean it can be uh like a proof of implementation saying that this is our backup options which is easy it's low uh the technical expertise to um to also obtain the information about the backup options is is low but if we're talking about implementation it would be moderate at least or even yeah even for your case this would be this would be low yeah just the information yeah if everything is just information it's just uh
R: Yeah okay so next up we're looking at the number of functions that always give the same result on the input divided by the number of functions used in total so basically the number of functions that is consistent and do not include any randomized so considering the same input so do not consider any randomized um no side effects no side effects exactly
um versus uh all of the functions yeah so
P: that is that would be extremely that's high yeah the problem with that is um there is no there is no definitive um uh restriction to when a developer creates code to say cool although things should be side side effect free most of the times a purpose of a a lambda
always go from a to b hopefully but anything can go wrong within that so to identify the actual problem of a function potentially being um in the left group of same input some output or the right group of it's going to be opposite to just identify that is going to be uh high
i feel like and that it's the difficulty of obtaining and technical expertise is also um high because it all depends on it's going to be various parts in your systems it's going to require a developer that knows uh the different languages the for example aws um the expertise required do and you actually physically have to know the code potentially because let's say you take a developer of this experience drop them in front of python they might say cool this looks like your boss and a it's going to give b uh but the code can potentially have side effects in it if they don't fully understand it so i hope that i also
R: yeah exactly yeah understandable all right. So then someone someone stated that you could just look at the number of users using your surface indicating that then we had a discussion on two aspects with which one on one of the discussants said well if you have a lot of num if you have a high number of users that basically means your system works right whereas another discussant said well if you have a high number of users you're more likely to face reliability issues so we're not talking about which view we agree with but just measuring the number of users where can you even find the number of users of your platform
P: You can't just find your number of users without having implementation like front end implementations which is entirely off the system because the thing is uh let's say it's a system that uses api key it one api key does not guarantee one user um so you can't say we have a million users you can kind of say we have a million requests yeah um so giving a definitive number of users is unobtainable unless your system was created to actually present that number i.e your front end for example but that's not that's not within your context within your context this is unmeasurable within my context no i would not be able to tell you how many users i can just tell you amount of data

R: Yeah okay um then the next one we have one uh two three four more then the next one um we're discussing is the number of functions working under a stress test versus the total number of functions so you're putting the system under under a certain load how many of your functions are still working that's the measure
P: What is the expected result of this question
R: The result is how can we uh how can we find out how many functions are still working under stress test how many functions are still working uh again
P: this would be similar to the one we saw at the top um for it's quite easy to just go to all your functions and look at their built-in dashboards uh and just see cool is what's the error rate what's the success rate um and even if we run into even if we run into a um a let's say a limitation where can only only run so many concurrent things um you will still see your failure depending on where it's happening so if it's for example a api gateway trying to access a lambda you'll see the failure on the api gateway side of things um so you you have all these uh yeah for every single thing in aws uh you have the measures to see what's going wrong especially in server listing side of world yeah ec2 is probably the only point to you but that's diverging the question um yeah so again on the built-in dashboarding yeah i wouldn't recommend like if you take all of these questions and you go back and you're like okay cool let's just use all the dashboards built-in for all the functions i wouldn't recommend that i'm just that's the easiest way to get it i would rather recommend that like custom solution actually having a cloud watch dashboard being built by technical expertise required moderate at least uh to get it all on like one spot
R: Exactly but it would be it would be readily available if you're just looking
at you just quickly need to go and nobody wrote the dashboard and you need to find this information and it's it's easy to go find and see that information
P: Yes
R: Yeah okay, during the next one someone proposed reliability in a very different light
and they said okay your system is reliable when it accurately represents data so it's basically the precision of numeric values so if you if you have a decimal value um and the true value so the true source is maybe a value is 70.752 but in your system or in your output you you only provide 70.75 as a value that will be less reliable than the other one that's basically what someone proposed here
P: It depends on the company and the context of what data is being used for I mean if it is an informative thing with factual statements uh inaccuracy would equal reliability uh but if it is for example uh uh something where it says a movie is 45 minutes and 60 uh 52 seconds into 30 seconds that is not seen as reliability i mean that is just misinformation uh i don't know what the topic for that would be so i wouldn't say that's a quantitative measurement for every single case uh but sure uh when you get into points where it is for for data science like in a sense then yes uh that is 100% reliability if your data misses one decimal it might skew the entire system um if we look at uh the world we're moving in like machine learning just one single parameter being off a little bit can again make the entire system act entirely different um so the difficulty of obtaining data in the sense of knowing of a i'm guessing if the precision of the numerical values is correct or incorrect exactly that's
R: Exactly that's exactly the question um it's it's easy if you have the source results and you can see the target results yeah but like this also this one is difficult to answer because it's all the perspective of do you have the data available that was fed in yeah so so maybe maybe assume here assume it is not available and we want to make make it available for the purpose of this measurements
P: Okay, it's uh like on the one side you can easily retrieve the data you don't need any

technical expertise to see the information there's uh like inside the database itself if you need the data and for example the dynamo db you can easily go there you can click a button and it shows you all the data in the tables so to find the source of the the actual data result is easy uh if you don't yeah if you don't have the the data that went in and then also was there a function that that calculations based on the initial data and then the problem with that is it's very difficult obtaining the data uh obtaining um the output because the output would look a lot different for example if you fit in 0.01 and zero instead of 0.012 into a big function it gives you a value out which you can never map back to where it initially came from because uh you need to know the functionality itself but yeah yeah if if you kind of feed in 0.012 but you see in the database it only says 0.1 exactly then you can you can physically see that quite easily so it all depends on what's the the the flow that it took uh yeah yeah to get to a point of difficulty so I can't give a difficulty um of of expertise required uh the obtaining the data i would put as low because you can quite easily get the data where it came from uh depending on the top part uh the previous question and this is the dependent yes and you can also see the result which is also easy but the technical cases yeah
R: Yeah okay that makes sense so the second to last one which we will discuss is the average response time to incidents so you have an incident within your context of what an incident is how long does it take until a response is given to the incident measured in time
P: depends how critical the issue is but yeah how it all depends on the level of the incident though if if the incident is a for example a production or a uh system the system lower yeah the response is like everyone's gonna drop instantly what they're doing and jump on it
R: But here we're not looking at we're not looking at actually measuring it so not looking at okay we're we're only doing this for production instance we're looking at the overall how do we know this response time how can we find out what the response time to an incident is and not whether it's like unless someone decided to say
P: I would say getting the data of how long it took for uh responding to a certain error it would be the quantitative measurement of when did the error happen and when was it fixed uh so it doesn't necessarily mean the average response time to the incident would be cool this is the time we responded to it and yeah uh fixed it and got it out there like actually getting that first sample size of knowing okay cool this is what happened this is when we jumped on it um it's very difficult data to obtain unless uh it is locked somewhere we recently had an issue like i don't know three four months ago um where where um um you could you could get past the uh a front end without the key um and if i now should tell you exactly how long our response time was to actually get to that issue i can tell you it took us 48 hours to get to the issue but when it was logged and when someone started looking at it in between that 48 hour time i can't give a it's very difficult to give that so just based on that information i'm gonna say it's uh yeah it's difficult of obtaining the actual data
yeah uh and then the technical expertise required to get the data which we know because you don't need it's just plain text type of information
R: Yeah exactly that makes sense. Okay let's move on uh move on to the last one which is the mean time to recovery so here the participants meant that uh whenever your uh your system goes down for a particular reason how long does it take you to recover it
so you can say that it is quite similar to to the last one but then the difference for this one is that you're specifically looking at it actually being fixed
P: yeah it's going to be the same exact thing um so the um the comparison to time to recovery versus uh all the other things we just spoke about is um is the data recoverable uh what is the impact of it uh what is the obviously the complexity of the uh problem that needs to be solved with code so all of that gets down to giving you a time and it would be vastly

different where if though if it contains data that's missing you need to try and get the data from the source and recover that and restore that if it's just code that's uh that's horrible but the data was not we don't care about the data we can just consume the next one that comes in it's much easier to to yeah to basically fix that so the i would say the technical expertise required is required is not low in this one uh i would say this is at least moderate um and the difficulty of standing would be again um no this one i wouldn't say is high because this one has uh especially in the team approach you have tickets attached to things so it's created a thing is done ticket is uh yeah so you can really what you're saying is you can really backtrack what has happened and that makes it not too difficult to set up the measurement but effort is required
R: Why again was the technical expertise here moderate instead of
low for the previous one?
P: Because the the amount of things that potentially is required and the information to get i feel stretch over bounds of just for example um the so for example in the technical side of things they say data is lost you kind of need to now know how do you recover this data how do you restore this data how do you not destroy things that goes wrong with it understanding the actual code or something that caused the problem and go fix that yeah so so i feel that it can be aligned between low and moderate it all depends on how it is but i wouldn't put the default response in the sense there to say no yeah because then you'll have a higher fail rate than putting in a moderate there
R: Yeah that that does make sense okay cool uh those were all i wanted to discuss for this session so as we have five more minutes left i wanted to ask you for for an evaluation what did you think of the interview things i could do better during the next interview things that were well things that didn't go well so what did you think of the interview overall
P: The interview is interesting maybe you think the one thing that i would um kind of make a bit more clear is a lot of the questions it's probably on purpose very open-ended but to kind of like state what is the boundaries around the open-ended questions um so um and then i think at the start of the interview kind of also state that um to like the following questions apply directly to the environment you work in and then if you're familiar for example that's uh you know the person works in aws or the applications around aws um having the mindset immediately knowing okay cool for everything in aws as easy as possible that you can do them um which should equal obtaining the data because when you look at the dev aspect looking at the wording of difficulty of obtaining data you immediately think about okay what's the mutations around it and where you push things to and yeah so i think that's the only thing but everything was yeah i think all the questions are valid and they follow each other and yeah they do um they do uh i don't even know what's the correct words for this pair with each other quite good
R: Yeah they do align it that's why that's why for me it's important to discuss all the same ones for a certain characteristic with the same person because you also have the
learning effect of not getting getting into the material. Okay cool well thank you so much i'll stop the recording thank you
P: All good