R: Dan ben ik nu de recording gestart. Allereerst bedankt voor het joinen van deze sessie. Deze sessie zal bestaan uit twee delen. In het eerste deel leg ik kort uit wat het doel is van het onderzoek, en wat de taken zijn die we gaan doen, en in het tweede deel gaan we daadwerkelijk een discussie voeren over de ISO 25010 standaard. Mijn onderzoek is gericht op software product kwaliteit, belangrijk is dat er een verschil is, we gaan specifiek focussen op software product kwaliteit en niet op code kwaliteit. Uit de wetenschap zijn er meerdere modellen die software product kwaliteit meten, en de meest vooraanstaande, de meest geaccepteerde daarvan is ISO 25010. De standaard werkt als volgt. Het is een hiërarchisch model dat bestaat uit drie niveaus. Software product kwaliteit op het hoogste niveau. Op het tweede niveau acht kwaliteits karakteristieken, en op het derde niveau nog sub karakteristieken waar we tijdens deze sessie niet op in zullen gaan. We zullen dus in gaan op de acht kwaliteitskarakteristieken, dat zijn functional suitability, performance efficiency, portability, reliability, compatibility, maintainability, security en usability. Dat zijn de acht waar we op in zullen gaan, en de taak die we zometeen gaan doen is als volgt. Ik lees een definitie voor van een karakteristiek, en jullie gaan een discussie voeren hoe je deze karakteristiek het meest objectief mogelijk zou kunnen meten. Dus wat voor data heb je nodig om dit te kunnen meten. Belangrijk is dus dat het karakteristiek, of de metrics zoveel mogelijk objectief zijn. Een voorbeeld van een niet objectieve metric is bijvoorbeeld als je aan iemand vraagt: in hoeverre vind jij dat deze applicatie portable is? Dat is dus geen voorbeeld van een objectieve metric. Yes? Ik herhaal dat er inderdaad geen goede en geen foute antwoorden zijn, het is vooral de bedoeling dat we in discussie met elkaar gaan hoe we dit het beste kunnen meten. Is het voor jullie duidelijk wat voor taken we gaan doen
P2: Ja, zeker
R: Oke, dan wil ik jullie vragen om te verzamelen rondom het bord. Ik heb hier ook een mooie set aan post-its en een pen, dus voor elke metric die we bedenken plakken we die onder het karakteristiek. Yes?
P3: En dan voor het persoon met het duidelijkste handschrift
R: Ik heb zelf niet het meest duidelijke handschrift dus vandaar dat ik ze ook zal voorlezen, maar mocht je het nog na willen lezen of de definitie willen begrijpen kan je natuurlijk altijd vragen aan mij stellen. Zullen we beginnen met de eerste quality characteristic? De eerste quality characteristic is functional suitability, en in de standaard wordt functional suitability beschreven als de “degree to which a product or system provides functions that meet stated and implied needs”. Dus dan is nu de vraag en de taak die we gaan doen. Hoe zouden we functional suitability zo objectief mogelijk kunnen meten?
P1: Ik zit even te denken, testen? Dat zou misschien een manier zijn. Dus je zou testen kunnen schrijven of de software wel echt doet wat de eisen zijn
R: Ja
P1: Ik zit even te denken, kijken naar hoe wij het binnen het team doen, is wij hebben dan een manier van testen schrijven die ook voor niet-techhneuten te begrijpen zijn. Dat zijn dan feature files. Dat zou een manier zijn om te kunnen zien of de code ook echt functioneel doet wat het zou moeten doen
R: En wat voor soort metric zou je daaraan koppelen? Wat is de numerieke waarde die uiteindelijk hieruit gekoppeld zal worden?
P2: Goede tests, gefaalde tests toch?
P3: Ja, stap een is hoeveel procent van de tests is groen
P1: Ja
P3: Dan krijgen we natuurlijk de discussie over test coverage

P2: Ik wilde net zeggen, het hangt er dan wel vanaf wat je coverage dan is. Dat is dan ook een metric misschien?
R: Schrijf ze maar op. Wat we kunnen doen is zoveel mogelijk metrics verzamelen en we kunnen altijd discussie voeren over metrics en bepalen: deze is misschien niet heel goed, deze moet in combinatie met elkaar gebruikt worden, deze kunnen juist niet in combinatie met elkaar gebruikt worden
P3: Kan je dit lezen?
R: Ik kan dit zeker lezen: het percentage slagende tests
P3: Test coverage. Ik zat zelf op een andere tak even te denken, over de implied needs. Je hebt bijvoorbeeld ook, wij hebben latency vereisten, dat wij events binnen een bepaalde real-timeness verwerken. Dat kan je ook meten, P90, hoeveel, laten we zeggen metrieken rondom, ik weet niet hoe ik het moet zeggen
P1: Eigenlijk zijn dit de non-functional requirements
P3: Latency, average, P90, weet ik veel wat, maar er zijn allemaal dingen die je eraan zou kunnen meten. Nouja, als we het hebben over responstijden meten. Vaak zijn er eisen over, binnen zoveel seconden. Dit is een beetje onder mijn niveau dit
R: Ze erboven plakken mag natuurlijk ook, zolang het maar duidelijk is aan welke van de acht karakteristieken we hem kunnen koppelen. Wat natuurlijk ook is, we kunnen nu metrics bedenken en uiteindelijk wanneer we er meer gaan verwerken zou het kunnen dat we denken, hij past beter onder een ander kopje. Dus het is altijd mogelijk om later nog metrics te schuiven
P3: Dus ook voor jou, als ik zeg average en P90, wat het kunnen natuurlijk allerlei dingen, wat je er precies mee doet met die data
R: Ik denk dat responstijden het grote dingen zijn die we willen meten, en op welke dimensie je doet
P3: Dan laat ik ze vervolgens weg, dan kan je het daarna
P2: Ik zit even te denken, want ik ben natuurlijk de enige frontender hier. Ik zit even vanuit dat perspectief te kijken. Maar ik vind het wel lastig omdat het heel moeilijk is. Je kan wel requirements omzetten naar end-to-end tests, maar dat is nog steeds subjectief, want je moet nogsteeds zelf bepalen wat je test, dus je kan dan als metric wel nemen het aantal testen, een beetje hetzelfde wat je hier hebt. De hoeveelheid testen die slagen. Maar helemaal objectief is het wellicht niet, want je kan er hier. Je zit nogsteeds met een soort use cases die je zelf bedenkt
R: Het is natuurlijk ook lastig om honderd procent objectiviteit te hebben, maar het is zoveel mogelijk een benadering dan aan iemand vragen: in hoeverre vind jij dit?
P3: Ik heb wel een voorbeeld. Ik heb bijvoorbeeld screenshot testen gemaakt. Je zou natuurlijk kunnen zeggen, je vergelijkt een UX design met pixels
P2: Dat is natuurlijk wel iets heel erg out of the box, dat bestaat nog niet zoiets
R: Het hele idee van een focus group is om te brainstormen
P2: Een screenshot heb je, en in feiten compare je dan met een vorige versie van de code. Je hebt natuurlijk ook een design dat opgeleverd is. Je zou dat ook met elkaar kunnen vergelijken, dat is objectiever. Wat bijvoorbeeld objectief is als je een product maakt voor een bepaalde groep mensen, wij hebben natuurlijk *team naam*, dan kan je heel objectief meten hoeveel mensen zitten daadwerkelijk op de app. Als niemand erop kijkt kan je concluderen dat het niet aan de requirements aan die mensen voldoet.
P3: Dit zijn meer functionele tests moeten we zeggen, die percentage. Maar binnen *naam organisatie* hebben we ook bewijzen dat op productie bepaalde flows werken
R: Hoe werkt zo'n bewijs dan?

P3: Nou elk uur doen ze dat, ze schieten een bepaald event in. Je kan dan meten, hoe vaak gaat het goed. Check. Elk uur, en dan meten het percentage. Of je het kan lezen later nog
R: Zeker, en anders hebben we altijd de opname
P3: Ik zou ook nog zeggen load tests, stress tests
P2: Is dat functional?
P3: Er staat ook implied needs
P2: Ja oke, ik zit al naar de volgende te kijken dat is performance
R: We kunnen nu natuurlijk brainstormen en als we denken, we vinden toch een betere, kunnen we
P2: Kunnen altijd nog zeggen dat die onder beide zit ja
P3: Stress test, dit gaat dan over de gewenste hoeveelheid, de target load, kan je die aan. DIt gaat over het meten van productie over monitoring, maar dit is dan meer vooraf. Onafhankelijk over een systeem, bij hoeveel load breekt het
P1: Wat ook nog wel een interessante is, dit zijn allemaal technische manieren om te meten, dus je gaat kijken naar het programma of het draait. Maar je zou het ook nog aan de gebruiker zelf kunnen vragen met een gebruikersonderzoek
R: Oke
P1: En kunnen meten met bijvoorbeeld een NPS score
P3: Ik wil nog iets zeggen over, ik ga het noemen AWS config rules, voor compliance. Compliance zijn ook functionele eisen
P1: En als je deze niet AWS specifiek opschrijven? Ik denk dat het voor ons prima is om ze AWS specifiek te noemen, en dan kan jij ze vertalen
R: Ik ga ze allemaal verzamelen, en analyseren, en wanneer niet het gewenste niveau van abstractie is toegepast ik ze een niveau van abstractie hoger slash lager
P3: Ik kan het weglaten maar dan wordt het vaag
P1: Dus eigenlijk is dit infra compliance. En met wat complied het zeg maar? Compliant op welk vlak, functioneel?
P3: Er zijn gewoon eisen, een S3 bucket heeft deze eisen binnen *naam organisatie*, en die staan nu in AWS config rules voor het gemak. Wij voldoen niet aan alle config rules momenteel, dus wij zijn incompliant. Maar je kan ook zeggen van per stage wat dan, of je voldoet aan de eisen van je organisatie
P1: Ik snap wat je zegt, maar ik vraag mij af of dat functioneel uitmaakt. Of die wel of niet security compliant is, functioneel doet die nog steeds zijn ding. En de gebruiker heeft niet perse de eisen dat de bucket beschermd moet zijn
P3: We gaan natuurlijk zo hier over praten, maar of die public access heeft of niet, dat is natuurlijk wel. Functioneel is de eis dat we data opslaan op een plek die niet beschikbaar is voor ongeautoriseerde mensen
P2: Dat is hoe je het functioneel ziet toch?
P3: Het implied dat we data wel beschermen toch?
P2: Ja, maar de implied needs in deze context zie ik wel meer als de needs die nodiug zijn om het te laten functioneren, en inderdaad wat P1 ook zegt, dit is meer dat het public is, een eigenschap, maar het impact niet de functionaliteit. Die bucket werkt nog steeds als die niet beveiligt is, het is nog steeds functioneel. Het is alleen niet compliant, maar is het dan nog steeds functioneel. Ik denk dat ik het in die zin wel met P1 eens ben.
P3: Dat is een non functional eis, er zijn ook functionele dingen van, er moet een audit trail zijn. Dat is een non functional eis. Is dat een implied need? We kunnen het allemaal onder security zetten duidelijk, maar het is voor mij wel een requirement van het systeem. Dat bepaalde dingen auditable zijn

P1: Waar we het onder zetten is dan het tweede. Maar wat het voor mij meer is van, valt die wel echt hier onder. Maar wel die dan wel onder valt kunnen we zo kijken. Maar ik vraag mij of of die wel hier onder valt, onder functional suitability
P3: Want je zegt functioneel maar nu zeg je implied, dus als je implied needs weg haalt en kijkt naar functioneel kan je zeggen, oke hij staat ergens anders
P1: Maar die needs, zijn dat dan needs van de gebruiker. Of zijn dat needs vanuit het bedrijf, of overheid
P2: Je kan natuurlijk ook zeggen dat het allemaal is natuurlijk
P1: Dat is dan misschien de vraag, wat is dan onze definitie
R: De definitie voorschrijft niet een specifieke actor erbij, dus daarachter kan zowel een bedrijf zijn, als een gebruiker
P1: Dat is wel het interessante, daar zijn wij het blijkbaar al niet over eens wat dat dan zowel inhoud of niet
P3: Ik denk dat we wel over de acht minuten heen zijn voor dit onderwerp
R: Laten we doorgaan naar de tweede. Performance efficiency, dat is de “performance, relative to the amount of resources used under stated conditions”
P2: Even analyseren hoor. Performance begrijp ik, maar “relative to the amount of resources used under stated conditions”
R: Ik kan natuurlijk niet een concreet voorbeeld geven, want daarover zijn we aan het brainstormen, maar als we het bijvoorbeeld even trekken naar een ander domein, denk bijvoorbeeld aan een auto. Hoe goed performt die
P2: Ten opzichte van de kosten
R: Precies, of hoeveel benzine is er nodig. Dat zijn dan de hoeveelheid resources die gebruikt worden
P2: Dus wat je eigenlijk zegt is performance is een soort spectrum, je mikt op een bepaalde level van performance op basis van je stated conditions, en dat wil je dan meten hoe goed je daar aan voldoet
R: Exact ja, dat klopt
P2: Dan heb ik hem even scherp
P3: Ik doe wat ik gezegd heb dat ik niet ga doen, dat ga ik toch even doen. Ik ga even zeggen server resource utilisation. Dus als je echte VM’s hebt dan kun je zien hoeveel ze staan te doen
P1: Ja dus dan kan je het zien, maar wanneer voldoe je hier wel aan, en wanneer niet
P3: Daar kun je een grenswaarde instellen waarvan je vindt dat die bij utilisation goed is. Bij memory wel maar bij CPU weet ik het even niet
P1: Maar meer van, dat je dat kan zien is al de eis. Of wil je er dan ook echt een bepaalde threshold, onder honderd
R: Ik denk dat dat een hele interessante vraag is: hoe bepaal je dan die threshold of die baseline. Waarmee vergelijk je het zonder teveel subjectiviteit te introduceren
P3: Dat weet ik niet
P2: Goeie vraag
R: Een vraag die ook aan bod komt eigenlijk tijdens dit onderzoek is: is het nuttig om dit tussen verschillende producten te meten die tot een bepaalde klasse horen, of kan je dit eigenlijk alleen maar doen voor hetzelfde product en op verschillende momenten in time. Als je bijvoorbeeld een nieuwe feature implementeert en een van deze metrics verbeterd.
P3: We zijn allemaal geen ops mensen hier, maar ik heb weleens met ops mensen gewerkt, maar die hebben wel een bepaalde notie als je een VM hebt, wanneer die te druk is. Ik weet niet precies wat ze gebruiken, er zijn daar binnen die wereld daar op zijn minst onderbuik

cijfers. Een VM die op 99% CPU zit, dat is teveel, en 30 is te weinig. Er zit iets tussen in, maar we kunnen wel zeggen, 99 is teveel. Dus er zit wel iets wat we meetbaar kunnen maken. En dan maakt het niet uit welk product je het op deployed. Maar er zijn natuurlijk heel veel metrics in VM's
P1: Ja dus zitten er eisen aan de resource, of aan de applicatie die erop draait. Dat zeg jij eigenlijk. Je zou het vanaf twee kanten kunnen bekijken. Je hebt een resource onafhankelijk van wat er op draait
P3: Volgens mij kan je er best wel wat over zeggen. Zeker bij VM's kan je er best wel wat over zeggen. Wat we neergezet hebben, hoeveel is die aan het stampen
P2: Ja, zeker
P3: En het kan dan natuurlijk, als je het hebt over netwerkdingen, krijg je dan wel allemaal dingen en metrieken als je het hebt over servers, het hangt dan wel af van het type resource, dan moet ik even zeggen: zo. Als je wilt kunnen we het uitsplitsen voor je
P2: Ik denk ook gewoon kosten, dus je stated conditions kan zijn: het mag niet meer kosten dan X dollar per maand
P3: Dus als je onderbezet blijft doe je het goed?
P2: Ja maar performance efficiency, performance latent to the amount of resources. Je kan natuurlijk de performance en de resource die je ervoor nodig hebt vergelijken met wat je kosten zijn. Stel je hebt een hele lage server utilisation, en hele hoge kosten, dat heel veel dingen draaien
R: Je zou dan eigenlijk zeggen, je kan er iets mee, maar puur wanneer je hem in conjunction met een andere metric meet
P2: Ja, kosten an sich zeggen natuurlijk niks. Want als je duizend services draait zijn je kosten natuurlijk hoger. Dat cijfer zelf zegt niks, tenzij je het relateert aan andere metrics. Het komt door het woordje efficiency, dat je er wel wat mee kan
P1: Dat je kosten tegen resource utilisation kan afzetten
P3: Ik ga even advocaat van de duivel zijn de andere kant op. Dus hier wat ik even heel erg niet serverless, en dan ga ik even serverless. We hebben een applicatie, we hebben X kosten per maand aan AWS, is dat goed, kan dat minder. Is dat proportioneel? En eigenlijk heb ik daar geen antwoord op. Dus in de serverless wereld vind ik het heel vaag, hoe je dit zou kunnen meten. Er worden lambdas opgespint, die doen wat en als ze klaar zijn stoppen ze. Maar misschien door andere technologieen in te zetten zou het een stuk goedkoper zijn, of zou het. Dus zeker met deze samen, die kosten, vind ik het in de serverless wereld wel spannend hoe ik het zou kunnen meten. Kijk ik even naar P1 ook
P1: Dus de vraag is: waar vergelijk je het mee toch?
P3: Ja, Wij weten bijvoorbeeld: we krijgen X miljoen events binnen in ons systeem per dag, maar we hebben natuurlijk wel specifieke business rules, die niet vergelijkbaar zijn met een ander product. Dus het is heel moeilijk om dan te zeggen van
P1: Ja, het hangt ook af van de garanties die je dan geeft. Als je bijvoorbeeld out of order events, replayen van events, allemaal van dat soort garanties wilt geven dan gaat het meer geld kosten
P2: Dat zijn dan de stated conditions
P3: Dat vind ik wel leuk wat je nu zegt, want number of retries, in een event driven architecture, en dan number of retries is best wel een interessante metriek. Want als het heel vaak gebeurt dan weet je dat iets niet zo efficient is als het zou moeten zijn
P1: Ja, voor een event driven architecture, number of retries. Ja, en als je dan dus, dat is echt het doorgeven van events, en als je dat zou doortrekken naar de serverless database, dynamodb, dan kan je ook kijken naar hoeveel van de data die je op slaat wordt ook echt

opgevraagd. Dus wij schrijven best wel veel data weg, maar als je heel veel weg schrijft, en maar heel weinig leest ervan. Dat klinkt ook niet heel efficiënt. Omdat schrijven duurder is dan lezen. Dus misschien is dat ook wel een teken
P3: Schrijf ik op: dead data. Het is geen term ik verzin het ter plekken
P1: Ik zit te kijken, we hebben er een aantal bij functional gezet, zoals de latency enzo, of die ook weer niet dan hierbij horen
P2: Ja goeie, respons tijden. Om daar even op in te haken, ik heb natuurlijk als frontend jongen. Het is best wel een volwassen space, de frontend specialisatie. Er worden hele frameworks opgetuigd om dit dan te meten. Dingen zoals inderdaad je bundel grootte, dat is een hele concrete metric die vaak gemeten wordt. Time to interactive is iets wat vaak gebruikt wordt
R: Wat houdt time to interactive in?
P2: Je hebt je browser, die laad natuurlijk een bundel in van Javascript en HTML. Het moet eerst nog in je browser geparsed worden. En omdat Javascript single threaded is, pas zodra het hele pakket binnen is en begrepen door jouw browser, dan pas kun je functies uitvoeren die met Javascript gebruikt worden
P3: Hoe noem je dat? Nog een keer
P2: Time to interactive. Dus op het moment dat jij een React website hebt met een grote knop op de home pagina en hij is nog bezig met laden, kan je nog niet op die knop drukken, want die action wordt met Javascript getriggered, en time to interactive is de tijd die jij nodig hebt om te kunnen interacteren met de pagina. Daar komt het op neer. Een hele belangrijke metric
P3: Je triggered mij. Ik ga even out of the box. Als je een mobile app hebt, is battery usage dan interessant
P2: Ja opzich wel
P3: Ik weet niet waarin ik het moet relateren
P2: Heb ik wel gezien in mijn stint als native app developer dat daar wel, dat die metrics wel meegestuurd worden ja, naar de analytische partijen in die space
P3: Dit is dan ook lastig, van waar moet je het dan tegenop zetten. Dus ik zeg niet dat dit
P2: Het is inderdaad meer een soort
P3: Hoe objectief is dit
P2: Hoe je dit moet zien, het is meer een soort tijdsreeks, en op het moment dat je issues hebt van mensen die klagen. Een app zuigt batterij leeg, dan kan je gaan debuggen op basis van deze data waar dat dan door komt
P3: Eigenlijk zeg je dat het misschien vaker zo is bij dit thema dat binnen het product je dit meet en dat je het dan binnen de tijd kan zien
P1: Trend analyse ja
P3: Binnen je eigen product, om dat dan met elkaar te vergelijken is appels met peren
P2: Dus de waarde an sich zegt natuurlijk niks, want je kan op elk willekeurig moment een app openen, en dan krijg je die door. Als je op dat moment een procent batterij hebt zegt dat natuurlijk niks
P1: Eigenlijk zou je dan trend analyse willen doen, gerelateerd aan een van deze metrics, en ook gerelateerd aan welke veranderingen je hebt doorgevoerd. Of dat nou software of infra is
P2: Gerelateerd aan een bepaalde release, of commit
P1: Volgensmij hebben we hier een aardig lijstje toch

R: Oke, ik denk dat we voor tijdpurposes ook door moeten naar de volgende. Portability, en in de standaard is portability gedefinieerd als de “degree of effectiveness and efficiency which a system or product can be transferred from one usage environment to another”
P1: Ja frontend wel
P2: Ja inderdaad, dat is wel een leuke. Maar hoe je dat dan objectief meet. Dat is wel een hele interessante
P3: Als je mobile zit kan je tellen, kan je op Ios, Android, hoeveel daarvan kan je. Zulke portability. Sommige apps kunnen op allebei, sommigen niet
R: Is het altijd, wat voor baseline zou je kiezen. Is het altijd gewenst om op alles te kunnen deployen?
P2: Ja dat hangt van je requirements af
P3: Maar in de regel wil je wel op allebei, tenzij er hele sterke redenen zijn van niet.
P2: Wat je kan zeggen is het aantal platforms wat je support gerelateerd aan het aantal codebases wat je onderhoud. Dat zegt iets over de portability. Als jij zeg maar een tweede platform wilt supporten voor een totaal nieuw project, dan ben je dus niet portable laat ik het zo zeggen.
P3: Mag ik hier nog even een detail over zeggen?
R: Zeker
P3: Ik zet even erbij: OS versions. Het kan best zijn dat je alleen apps hebt die alleen maar van Android 20+ werken. Dus je kan best wel tellen, hoeveel android versies ondersteun je dan. Dat zegt iets over de portability van je applicaties
P1: Zegt dat iets over de portability, of over de compatibility?
R: In dit geval is compatibility, dat gaat over informatie exchange. In die zin heeft compatibility in het model een hele andere definitie
P1: Ja, maar in de frontend mobile wordt heel veel over compatibility, zoals die browser check, en dit gaat meer over kan je het transferren en compatibility is in hoeverre is het compatible, dus daarin vind ik deze kwaliteitsmetric opzich misschien niet de sterkste, dus ik zou eerder compatibility zou interessant zijn, en niet die andere compatibility, maar met hoeveel users ben je compatible, met hoeveel apparaten
P2: Je kan ook zeggen van oke, transferred from one device to another. Of system to another. Je kan zeggen dat je dat als gebruikers, je consumeert het vanuit een ander systeem, of oke: we porten de codebase naar een ander systeem. Bijvoorbeeld in de gaming industrie heb je verschillende soorten videokaarten, dat moet dan weer een andere API supporten, maar vanuit de gebruikerscontext zou ik voor de frontend zeggen, je hebt dezelfde metric die we eerder noemden. End-to-end test kan je draaien in verschillende browser environments. Gefaalde tests kan je weer als metric gebruiken
P1: Ja, dus ik denk bij deze is het heel interessant om te kijken naar wat zijn nou die grenzen tussen bepaalde blokken? Ik weet even geen betere naam. Frontend is een blok, backend is een blok, en hoe meer coupled die zijn, hoe moeilijker het is, of hoe slechter het is voor de portability. Maar hoe maak je zoiets concreet?
P2: Dat is inderdaad wel lastig
P1: Dus een duidelijke API definitie bijvoorbeeld
P2: Aantal microservices ja. Ik snap wat je bedoelt, maar ik weet nou niet hoe je dat concreet en objectief kunt maken
P1: Nee, ik vind deze best wel lastig in de zin van: als je het ook naar de backend zou doen, je hebt best wel veel de discussie of moet het cloud native. Bijvoorbeeld je zit op AWS en je zou alles op moeten kunnen tillen en het zou naar Azure moeten kunnen gaan. Dat is ook een beetje het ding van Terraform. Het is het hele idee dat je het op kan pakken, en ergens

anders neer kunt zetten. Ik heb bijvoorbeeld ook aan een project gewerkt, dan heb je Xamarin en dan kan je heel makkelijk va los naar Android, en vice versa, maar mijn ervaring is een beetje dat het echt een hele erge afweging is van kosten. De kosten die het kost om het helemaal cloud native of os native te houden versus hoeveel kosten kost het om het bij wijze van spreken opnieuw te bouwen of om te schrijven
R: Dus eigenlijk wat ik je hoor zeggen, klopt het, is portability is niet altijd een gewenst attribuut
P3: Precies, dat schrijf ik op, het is een afweging
P2: Het is een spectrum, dat heb ik ook ervaren. Denk bijvoorbeeld aan een groot bedrijf zoals facebook of uber, die zouden nooit voor zo'n oplossing kiezen, want die hebben de resources om native op een platform te bouwen. Maar als een klein bedrijf en je hebt de keuze om een webapplicatie ook te distribueren vanuit een codebase op meerdere platforms, en op die manier portability te garanderen. Maar dat is wel echt een ander beestje, hoe je dat dan zou kunnen meten is toch iets met je kosten, of aantal developers ofzo, aantal uren dat je eraan besteed
P1: Dan is het een kostenvraag toch, kosten van onderhouden, portability onderhouden versus het allemaal omschrijven. Dan zou je eigenlijk twee soorten. Ze zijn weer een beetje soort van subjectief want je moet in man uren gaan rekenen, en het zijn zulke grote dingen dat het echt een beetje een soort schatting is. Maar dan zou je dus
P3: Maar portability is niet altijd gewenst, stel je wilt met data overgaan, dan krijg je wel dit soort dingen toch. Browser versions, Android versions, ik heb opgeschreven die hele Terraform handel, dat hebben wij ook gedaan, Terraform, Kubernetes, nooit geport uiteindelijk ander verhaal
P2: Dat zul je altijd zien
P3: Maar goed je kan zeggen, als je dat doet kan je misschien wel op drie clouds deployen, ben je heel portable. Dan is het wel met minimaal werk. Je kan zeggen het is meer portable. Portable is nooit nul een zou ik zeggen. Een beetje een grijs gebied. Het is meer of minder portable. Ik heb ook opgegeschreven hoeveel runtime versions. Op Node 16 draait een Lambda, op Node 18 draait een Lambda, of weet ik veel wat. Dus daar kan je ook wat van vinden. Hoe backwards compatible ben je met allerlei dingen. Ik zou het niet zo zinvol vinden, maar je kan het wel meten
P1: Maar toch voelt het een beetje met die alsof het een beetje compatibility is, het zegt niet hoe makkelijk je naar een nieuwe runtime versie zou kunnen gaan
P3: Over toekomstige versies weet je het niet
P1: Eigenlijk is misschien hier het aantal regels code bijvoorbeeld dat je hebt is echt een heel erg natte vinger schatting, minder regels code heb je over het algemeen, makkelijker te porten, omdat je minder custom dingen hebt, minder kans, procentueel heb je minder kans dat je ergens vast loopt
P2: Hij is gewoon erg lastig. Hoe ik hem ook ziet is eigenlijk als je hem plat slaat. Je wilt een waarde hebben waaruit je uit kan lezen, hoe makkelijk is je codebase over te brengen naar an dere platformen. Maar hoe je dat weet
R: Voor tijd purposes, laten we nog een minuut besteden aan de portability kwestie. Als we nog wat concluding remarks willen maken kunnen we over naar de volgende
P1: Laten we naar de volgende gaan
P2: Ik ben wel een beetje uitgespeeld
R: Reliability, reliability is in de standaard de "degree to which a system, product or component performs specified functions, under specified conditions, for a specified period of time

P1: Dit voelt echt met de drie keer specified, meten is weten. Dus wat je gemeten hebt vergelijken met wat de eisen op papier zijn. Volgensmij heb je hier de DORA metrics een beetje hier terug voor een deel. Dat is een wat bredere
P3: Ja voor de performance van een team. Ik begin even heel simpel. De papiertjes zijn bijna op. Zal ik zeggen percentage uptime
P2: Ik wilde net hetzelfde zeggen. Uptime is een mooi woord
P1: En hoeveel moet dat zijn?
P2: Dat is specified
P3: Iets met negens
P1: Dat is wel een goede om erbij te zetten, how many nines? Dat is wel een beetje de industrie standaard om het. Maar de meeste systemen specficeren het wel zo natuurlijk
P3: Moeten we zeggen data loss ofzo? Moeten we daar iets over zeggen
P2: Ja, je zou hier ook je retries ook nog
P1: Het is wel interessant waar we de afgelopen twee maanden een beetje mee zitten. Bij een event driven architecture gaan er allemaal events in, en is het een beetje een soort blackbox. Bij sommige komen er dan twee uit, bij anderen plekken kan het zo zijn dat het event ineens ophoud en niet doorcycled. Dus hier is het misschien per plek bijhouden hoeveel events je dropt, in een event driven architecture
P3: Maar het kan wel specified zijn
P1: Ja, maar dan zou je het wel kan vergelijken met wat je verwacht. Dus in event driven architecture het aantal dropped events
P3: Eigenlijk is het dropped versus succesfull
P2: Wellicht
P1: Ja, of absoluut aantal. Ja, maar dat je het in ieder geval gaat meten zodat je het kan vergelijken met wat je verwacht
P3: Just in time post its. Alle dropped dingen die je nu beschrijft zijn functionele wensen. We moesten het
P1: Ja, maar
P3: Ik zou juist eerder de onverwachte drops willen meten, of zijn dit onverwachte
P1: Ja oke, maar waar we nu mee bezig zxjin is het een start zeg maar
P3: Moeten we zeggen error count?
P2: Ik wilde hetzelfde zeggen, error logging. Ik denk een beetje als je er een metric van maakt het aantal errors
P3: Aantal errors
R: Waar zou je het aantal errors mee kunnen vergelijken?
P2: Met het verwachte aantal. Ik verwacht dat je een bepaalde threshold wilt samen stellen, we willen dit garanderen in de frontend dat de applicatie nooit vast loopt. Dat betekent dat je nul errors vewracht
P1: We zijn nu wel bezig natuurlijk om die. Wanneer wordt je uit het bed gebeld. Om die thresholds op te stellen, en daar proberen we wel echt te zeggen. Oke, bij deze eventstroom zijn tien errors oke, maar als het er duizend zijn, dan moet je uit je bed gebeld worden. Dat zijn dan de specified conditions waarmee je het vergelijkt
P3: Maar succesvol kan ook zo zijn, versus specified
P1: Je gaat het dan vergelijken met het aantal succesvol. Het resultaat is dan dat je een percentage krijgt, maar waar ga je het percentage dan mee vergelijken
P2: Dat is ook iets wat je gespecified hebt. Of je nu een absoluut aantal hebt dat je specified of een percentage, dat maakt niet uit
P1: Uiteindelijk moet je het ergens mee vergelijken, wat je specifieert

P3: Ik ben wel zo naief om te zeggen, dat voor de meeste systemen wil je eigenlijk geen events in een DLQ hebben toch? Dus als er iets in de DLQ komt, ookal komen er hier honderd duizend miljoen en hier eentje, is het slechter
P2: Dan heb je honderd procent gespecified
P3: Ik denk dat je dan alsnog nul procent wilt
P2: Maar dat is dan alsnog toch een specificatie
P3: Maar dan is het de vraag, is het voor een product of is het voor alle producten te zeggen
P2: Ja, op die fiets
P3: Ik zou zeggen, voor een heleboel producten, ik heb nog nooit een product meegemaakt waarin ik zeg: DLQ is goed, dat is een gewenste uitkomst
P2: Ik kan mij wel voorstellen, als je het breder trekt dan alleen ons team. Je hebt een team dat een API aanbiedt, en die garanderen bepaalde rates, hij gaat van 95% dat je data doorkrijgt ik zeg maar iets. Als je binnen die threshold blijft heb je geen reden om uit je bed gebeld te worden toch
P3: Maar eigenlijk is dat uptime, maar eigenlijk zeg je ook met je SLA
P2: Ja inderdaad. Je geeft een commitment af denk ik
P1: Gemeten waar vergelijken met je SLA
P3: En een goed ding is uptime, maar een SLA kan meerdere dingen. Je hebt ook dingen zoals requests per second, kan van alles zijn
R: Een vraag die ik bijvoorbeeld heb bij de SLA's, ze komen vaker voor, is: hoe verhoud de SLA zich in dit plaatje? Is de SLA een metric an sich, of is de SLA een soort van acceptable filter die je applied over een metric?
P2: Ja zo zie ik het wel, dus de SLA's zijn eigenlijk de specified conditions
P1: Ik denk ook wel dat wanneer je de threshold van die SLA over gaat, dan is het eigenlijk soort van al te laat. Ik denk dat je eerder al uit je bed gebeld zou moeten worden. Als bijvoorbeeld die SLA zegt van de latency mag zoveel seconden zijn, bij wijze van spreke, als die ineens al over de helft van dat heen gaat zou je misschien wel uit je bed gebeld willen worden, dus dat is niet een op een met elkaar. Lijkt mij dat je wel wat meer punten wilt
P3: Je hebt soft alerts en je hebt hard alerts
P2: Ja eens, maar dan nog is het op basis van de SLA specifieer je je condiities toch. Het is alleen je bouwt dan een buffer in, je halveert bij wijze van spreke
P1: Willen we compatibility?
R: Zijn we klaar om naar compatibility te gaan?
P3: Ik weet niet, ik zit even te denken. We hebben opzich ook wel tijd toch?
R: Ja zeker
P3: We hebben nog een half uur voor nog vier toch, zullen we dan wel door?
R: Laten we door gaan naar compatibility. Mocht je nog iets willen inschieten voor reliability kunnen we altijd even terug gaan?
P3: Je kan dadelijk zeggen waar je de minste van hebt, daar kunnen we op terugkomen
R: More is not always better. Compatibility is de "degree to which a product or system can exchange information with other products or systems", dus dat is dan een ander type compatibility dan wat P1 net beschreef. Er is frequent wel discussie over deze
P3: Het is eigenlijk dus interoperability, dat is wat ik lees
P1: Je komt gauw bij de contracten uit, dus API's
P2: Ik wou net zeggen
P1: Maar in hoeverre ga je API's meten
P2: Je zou kunnen zeggen, we hebben een bepaald contract. Hoeveel systemen ondersteunen dit contract

P1: En dat is dan binnen een bedrijf, dan heb je een percentage
P2: Ja dan zou je dus kunnen zeggen, dat dat nummer de compatibility meet van alle systemen die eronder hangen, op een bepaald domein, of een bepaald event. Maar eigenlijk alleen maar waar dat contract op dat moment over gaat. Je kan het niet breder uitmeten
P1: Ja oke, precies. Je zou m wel om kunnen draaien natuurlijk. Je kijkt vanuit het bedrijf of afnemer van is er een bepaalde standaard in hoeverre voldoe je daaraan, en in hoeverre is dan nog vaag. Maar je zou wel kunnen meten van, je hebt tien API's, hoeveel daarvan voldoen aan de afdeling slash company standards
P3: Dus API compliance
P2: Al vind ik wel dat dat nummer an sich zegt niet zoveel. Want als je een API bouwt waar maar een call in zit, is dat aantal een. Dan moet je het eigenlijk ook relateren aan het aantal endpoints. Dan kun je een percentage berekenen
P3: Ja ik heb percentage
P1: De vraag is nog, hoe ga je zoiets meten. Ga je het automatisch meten, is het een handmatige check
R: Ik denk dat dat inderdaad niet de concern is voor deze sessie. Hier identificeren we ze gewoon en een volgende stap van het onderzoek gaan we interviews met mensen afleggen waarin we gaan kijken wat voor informatie en wat voor bronsystemen hebben we nodig om deze metrics te kunnen meten
P3: Ik ga even een hele domme opschrijven, maar goed ik ga het gewoon opschrijven. Ik kom uit de E-health wereld: number of supported standards. Dat ga ik even iets concreter maken. Bijvoorbeeld HL7
P2: Health life 7?
P3: Nee, niet health life 7. Hoe heet dat andere ook alweer. Dit gaat over domein, domain standards
P1: Volgensmij heb je hier nog wel andere ISO standaarden voor, of niet?
P2: Volgensmij wel, allemaal security standaarden enzo
P3: Dan gaan we iets anders bedenken. Jullie kennen HL7 niet. Er zijn bepaalde soort van bouwblokken die je kan afspreken met elkaar. Nictis, Aorta standaarden, ik ga allemaal buzzwords neerzetten ja
P1: Hier kan je ons alles
P3: Ik ben vergeten hoe dat ookalweer heet. Dan heb je een domein en dan spreek je met elkaar af, een medicijn gaan we zo over het lijntje sturen altijd. Dat werkt voor apotheken, huisartsen. Daarom zeg ik interoperability is volgensmij meer wat hier staat. Als je voldoet aan die standaard dat in dat domein van toepassing is, dan ben je meer interoperaable
P1: Ja precies, dat is het precies per domein. Want ik weet vanuit rail en telecom wereld dat je bepaalde standaarden hebt van het verkeer moet zo lopen, dus zeker in de railwereld, in Europa heb je standaarden van, het gaat van 4g naar 5g zijn ze bezig. En dit is gewoon de standaard en daar dien je je aan te houden. 4g, 5g, dat is zo breed. Dus ik denk dat ook het protocol waar je over praat. In algemene zin
P3: Dit wordt wel heel vaag, wat bedoel je met protocol?
P1: Ja dus bijvoorbeeld dus of je
P3: Is HTTP een protocol?
P1: Ja
P3: Maar dit is specifieker he, dus meer per applicatie niveau
P1: Ja, maar je kan op verschillende lagen, je OSI, en op elke laag heb je wel protocollen
P3: Moeten we daar iets over zeggen dan?

P1: Ja, ik zat even te denken wat, dus meer gewoon van je kan verschillende layers van je OSI hebben, dus in hoeverre, hoeveel lagen ben je compatible, maarja compatible met wat
P3: En het leuke is, is meer beter? Ik support zowel UDP als TCP, ben ik dan beter?
P1: En REST, en GRDC
P3: Ik heb meer smaken gemaakt, ben ik meer interoperable. Ik ga het even weghalen, ik ga even opschrijven hier wat je idee was. Ik schrijf gewoon even number of protool, maar ik schrijf het gewoon op. Je kan hier al zeggen, binnen e-health zijn we gebonden aan vijf standaarden, en ik interpreter er drie
P1: Maar dat komt uit wetgeving dan?
P3: Ja soms wel
P1: Dat is een intersesante dan
P3: Maar iets vanuit het domain, en soms is dat dan by law, en soms niet
P1: Dan moet je kiezen of je de outlaw wordt
P3: Dus je moet dan wel binnen het domein kiezen va wat zijn de standaarden dan, maar dan kan je daarna meten. Het is lastig misschien maar als je uber bent, of je bent een innovatief bedrijf en er bestaat geen competitor dan is het ook geen standaard, dat weet je dus niet. Even heel dom gedacht: als je een systeem hebt zonder API's, ben je dan interoperable? Heel closed. En is dat dan erg?
P1: Wat ik hier een beetje mis, over het algemeen bij deze kwaliteitscriteria, van deze protocollen en dingen van, waar vergelijk je dit mee. Dus je kan het ook wel weer meten en sommigen zijn ook echt required by law, dus daar kan je het echt wel duidelijk bij
P2: Ook wel omdat we er met onze eigen bril naar kijken. Als je het droog koopt en objectief kijkt. Stel je zou deze metrics allemaal meten, en je compabililty score is nul. Dat is niet per definitie erg als dat geen requirement is. Maar wij vinden het erg omdat we er met onze bril doorheen kijken, maar het hoeft niet perse zo te zijn. Dit gaat volgensmij alleen er maar over het meetbaar maken van
R: Dat klopt inderdaad, ik denk dat je op een ander abstractieniveau kijkt als je gaat kijken van is dit belangrijk om uberhaupt te meten of niet. Maar wat je uiteindelijk natuurlijk kan doen is als je deze acht al deze metrics verzamelt, uiteindelijk tot een mogelijke score voor product kwaliteit, dan kan je gewichten toe gaan voegen. Dus de metrics die erbij hangen zijn inderdaad gebaseerd op het feit: we willen dit meten
P2: Dus in die zin denk ik dat protocollen inderdaad, het kan best toevoeging leveren aan het meetbaar maken. Of het waardevol is is afhankelijk
P1: Ik zit te denken als je een data platform hebt, GraphQL, een websocket aanbieden
P2: Ja inderdaad ja
P1: Weet ik veel allemaal wat
P3: Maar moet je en GraphQL en REST aanbieden
P1: Ja dan is het dus de vraag welke eis leg je hieraan. Maar dat tellen. Ik weet niet of je ook nog het aantal gebruikers per protocol zou willen zien
R: Laten we nog twee minuten trouwens nemen voor deze
P1: Ik denk voor een aantal protocollen, dat het dan al wat redelijk wat zegt. Het is dan wel de vraag welke, en wat zijn je eisen
P2: Dat is inderdaad wel waar. Ik zeg GraphQL maar dat is eigenlijk nog een abstractieniveau te hoog. GraphQL support meerdere protocollen in die zin. Omdat je allemaal sockets daarmee verstuurt
P3: Teveel plat slaan, zou je inderdaad. Hij doet het weer. Number of connected systems, en connected zou ook kunnen zijn, supported systems. Dus ik noem maar wat, ik kan de apothekerswereld toevoegen. Ik kan vijf apotheek systemen, ik kan er drie van de vijf

koppelen. Dat zegt wel iets over hoeveel soort systemen je mee kan koppelen. Ik kan ook zeggen: ik heb vijfhonderd instanties van apotheeksystemen gekoppeld, dat is een ander soort metriek en kan je ook meten. Met allebei kan je wat. Het zegt wel iets over als je niet gekoppeld bent met wie dan ook dan zou je vast niet zo interoperable zijn
R: Een soort proof van interoperability is het dan
P1: Je zou hem ook nog de andere kant op kunnen zien. We hebben hem nu voornamelijk een kant op. In hoeverre ben jij interoperable. Maar je kan natuurlijk ook naar de andere: in hoeverre zou je richting andere kunnen gebruiken. Dus hoeveel protocollen kan je consumeren, of hoeveel dingen kan je consumeren. En eigenlijk zou je deze allemaal kunnen gebruiken en ook nog de andere kant op. Dus je hebt hoeveel biedt je aan, maar ook hoeveel kan je consumeren
P3: Incoming
R: Oke, ik denk dat de tijd voor compatibility op is
P3: Jammer, mijn favoriete onderwerp
R: Laten we doorgaan, maintainability. Maintainability is de degree of effectiveness en efficiency with which a product or system can be modified by the intended maintainers
P2: Oke
P1: Ja, toch weer regels code, of niet?
P3: Ik ga hier wel een DORA metric opzetten, ik ben even vergeten hoe die heet, die wel hier over gaat. Wat is het, time to fix ofzo?
P1: Mean time to delivery, of niet? Dus de tijd van feature definitie naar dat die in productie staat
P2: Doorlooptijd in feite
P3: Dus je hebt een idee, en je wilt de nederlandse tekst aanpassen. Typo fixen, hoelang duurt dat
P2: Ik denk ook totdat het op productie staat, neem ik aan
P1: Ik denk misschien hier ook, als je kijkt, we werken natuurlijk met Scrum. Hoeveel procent van de tijd ben je bezig met ops versus dev, in een DevOps team. En dat is meer van hoeveel procent van de tijd ben je echt bezig met de features bouwen, waarde opleveren, en hoeveel procent van de tijd bezig met de boel onderhouden, en draaiende houden
P2: Ja
P3: Als we het hebben over alerts, dat gaat ook over ops eigenlijk. Als je heel veel false positives krijgt, je gaat dingen onderzoeken die niks opleveren, zit die hier ook al in? Of is het een eigen metriek waard, hoeveel alerts zijn daadwerkelijk echt iets?
P1: Dat is eigenlijk een beetje een gevolg ervan, even kijken hoor. Je zou hem ook wel expliciet kunnen opschrijven
P2: Number of issues triaged of zoiets
P3: False positives. Ik vind hem niet heel sterk hoor
P1: En als je hier nou gewoon aantal regels code, gewoon heel plat
P3: Maar jij schrijft best wel veel regels code met je yarn.lock files, heb je het door?
P1: Ik mag ze ook zelf maintainen toch
P2: Ik denk ook het aantal regressie bugs dat geintroduceerd wordt. Ik denk dat dat ook wel
P1: Dus het totaal aantal bugs
P2: Dat terug keert.
P1: En hoe bedoel je terug keert?
P2: Gefixte issues, en zeker in de frontend is dat vaak een issue. Regressie, dus issue gefixed, bouwt een nieuwe feature, deployed, bug is weer terug omdat je iets hebt vernacheld

P3: Recurring bugs
P2: En dat zegt iets over hoe goed je cruciale dingen hebt afgedekt in tests, en dat zegt iets over je maintainability, omdat je dan weer extra werk moet doen om iets te fixen. Je maintenance time is meer, het is langer
P1: Ja, precies
P3: Gaan we door met deployment frequency, heb ik erbij geschreven versus target. Een hele discussie over releases, maar als je elke dag wilt kunnen deployen, maar je kan het niet, zegt het wel iets over je maintainability. Als je een mobile app hebt en je wilt maar een keer per maand deployen, dan heb je target gehaald. Dus het gaat over hoe vaak je deployed versus waar je zou willen zijn
P2: Ja, dat is een goede
P1: Het aantal technieken dat je gebruikt? Het aantal technieken, frameworks, hoe meer je er hebt hoe moeilijker te onderhouden
P3: Number of programming languages. Cognitive load wil je eigenlijk meten
P1: Dus als ik kijk naar onze omgeving hadden we Kotlin, Python, Typescript, ja dat wordt wel leuk om te maintainen
P3: Dan kan je ook zeggen number of clouds
P2: Number of frameworks
P1: En hier is eigenlijk less is better
P2: Ja, tot op zekere hoogte. Nul is zeker hoog toch?
P1: Puur HTML en Javascript, dan heb je nog steeds. We hebben nog een kwartiertje. Dan gaan we nog even lekker door
P3: Precies, je gaat nog even langzaam *inaudbile*
P2: Je zou het kunnen meten, ik denk wel dat het heel veel overlap heeft met die mean time of delivery, want het zit daarin
P1: Daar ben ik het wel mee eens. En misschien de onboarding tijd van nieuwe mensen
P2: Daar zat ik ook al aan te denken
P3: First commit
P2: Time to first commit, maar dat is ook wel
P1: Maar voor mij is de uitdaging altijd, hoe snel kan een nieuw iemand die een clean laptop krijgt iets naar productie uitrollen. Ook al is het maar een typo fix
P2: Het is wel lastig als je dat als metric gaat doen, want wat ga je dan geven als onboarding taak. Dus het is in die zin wel subjectief, dat het afhangt van het werk. Dus de waarde van de eerste onboarder zou nooit hetzelfde zijn als degene van de tweede onboarder, en dat zegt eigenlijk niks
P1: Nee, maar wat je wel kan doen is om een zo klein mogelijke story te kiezen, en dan zal er misschien wel een klein beetje verschil in zitten, maar
P2: En anders misschien helemaal objectief zou het niet waarden
P1: Hmm?
P2: Helemaal objectief zal het niet worden
P1: Ja, dat is dus de vraag, voor mij zegt het wel wat, qua maintainability
P2: Ik vind het een interessante. Ik had hem ook in mijn hoofd, maar ik heb hem bewust niet gezegd, om deze reden
P3: Ik zet er wel een vraagteken bij
P1: Volgens mij is het wel
P3: Mag ik er nog iets over zeggen. Ik ga iets heel doms zeggen weer: percentage dead code
P2: Dat vind ik ook een lastige. Define dead zeg ik dan

P3: Quotes zet ik er om heen
P2: Dat is ook subjectief hoor
P3: Nee hoor, het is gewoon code die nooit uitgevoerd wordt, maar er wel in staat. Daar moet je omheen kijken. Dus ik zit te denken ik ga programmeren, wanneer ben ik nou effectief en efficiënt, wanneer ik niet om allemaal dingen heen moet kijken, dingen die allemaal troep zijn
P1: Live code coverage ja. Reverse live code coverage
P3: Als je code gaat schrijven ben je niet zo effectief en
P1: Het is makkelijk om nieuwe code te schrijven, het is moeilijk om code weg te halen
P2: Ja tuurlijk ja
P1: Het is moeilijker om code weg te halen misschien
R: Laten we de discussie moven richting security. Security is de “degree to which a product or system protects information and data, so that persons or other products have the degree of data access appropriate to their level of authorization”
P2: Number of pen tests succeeded. Zoiets? Of in ieder geval. Security probes
P3: Pen test issues
P1: Je hebt OWASP 10, OWASP 10 complaint
P3: Maar hoe meet je dat?
P1: Dat moet je dan gaan meten. Maar je kan meten of je wel of niet compliant bent. Het is wel objectief
P3: Is dat zo?
P1: Je hebt, je kan externe bedrijven ervoor inhuren om dat te doen
P3: OWASP 10 compliant
P1: Je hebt nog ISO 9001, en nog een aantal andere maar 9001 is wel
P3: Is het een metriek?
P1: Nee, het is ook certificering meer
P3: Number of open security, vulnerabilities kan je zeggen, issues, incidents
P2: Ja number of incidents
P3: En dan zeg ik even, per severity. Dus wat we moeten doen voor die DevSecOps controles. Gewoon dashboards open zetten hoeveel findings staan er nu open, zoiets. En ik zeg er ook even bij: number of, dit is eigenlijk op code niveau kan je zeggen, en dan heb je hetzelfde op infra niveau. Sec issues infra, even voor R in hoeverre je daar naar hebt gekeken. Maar die trust advisor dingen heb ik het dan over.
P1: Hoeveel datalekken je hebt gehad?
P3: Ja goeie
P1: Zou nul moeten zijn, en is het hoger
P2: Ook wel interessant, want het is wel zo stel een systeem is zo lek als een mandje, ik heb een datalek gehad, en ik heb grof geinvesteerd in het verbeteren, ik heb nog steeds een datalek gehad. Zegt dat iets over de security van mijn applicatie nu?
P3: Ja oke, last year
P1: Maar dat heb je bij heel veel van deze toch. Maar je zou dat dan over een bepaalde tijd kunnen meten, en dan een tijdje later, en dan kan je zien
P3: Hoe kunnen we zien dat teveel mensen toegang hebben tot iets? Dat er iets open staat. Number of people with full access
P2: Number of people with admin privileges
P1: Je zou ook kunnen kijken, met die elevated privileges, gaan monitoren wanneer er bepaalde permissies gebruikt worden
P3: Number of times break the glass is wat je doet

P1: En dan voornamelijk ook door wie
P3: Maakt dat uit? Want het maakt toch niet uit
P1: Misschien is die dan wel anders, dus misschien ook voor bepaalde rechten monitoren wie daar gebruik van maakt
P3: Sensitive rights usages
P1: Dus als je zo concreet kijkt bij ons hebben we de elevated privileges, we kunnen gewoon elke keer gaan loggen als iemand het doet, en dan voornamelijk ook wie. En als er opeens een naam bij staat van, hey, bijvoorbeeld een *naam oud collega* staat daar nog bij. Hoe zeg je dat?
P3: Laten we zeggen, meestal is een development team pizza size, en als daar uit komt 45 mensen is er wel iets aan de hand, als het sensitive data is, hangt van je data sensitivity af natuurlijk
P1: Volgens mij zijn het wel twee verschillende invalshoeken, die gaat gewoon kijken van wie staat er in die database met toegang met permissie, en die gaat kijken naar de acties, en die ga je monitoren
R: Laten we de discussie rondom security zo ook afronden
P3: Jammer
P1: Deze is voor jou
R: Usability is de “degree to which a product or system can be used by specified users to achieve goals with effectiveness, efficiency, and satisfaction”
P3: Kunnen we daar wel de NPS op schrijven?
P1: Ik zit even te denken of die ook hier bij zou moeten of dat die hier naartoe moet. We hebben nog niks gemoved toch?
P2: Ik vind dat altijd zo moeilijk, dit is binnen de frontend natuurlijk heel erg belangrijk, maar het is heel moeilijk om te testen, omdat het altijd afhangt van je requirements
P3: Retentie, hoe noem je dat?
P2: User retention kan je gebruiken, maar dat hangt ook van je context af. Bij ons is het niet relevant zou ik zeggen. Maar bij een commercieel product natuurlijk wel
P1: Retention wil zeggen?
P3: User retention
P2: Dus hoelang gebruikers op je website zitten. Bij ons is het zo dat onze applicatie is gemaakt om ‘s ochtends op te zetten, en ookal gebruiken ze hem niet staat die de hele ochtend open
P3: Daarom is ons geweldige user log, we hebben een tien uit tien score bij ons
P1: Als je een webshop hebt is het heel interessant
P2: Of youtube, hoelang mensen erop zitten is hoeveel geld ze verdienen
P3: Kunnen we iets zeggen in algemene zin van compliance with accessibility?
P2: Ja, zeker, zeker ja. Bij ons speelt dat niet maar
P3: Accessible for blind people.
P2: Ik zat in mijn hoofd te lachen hoeveel verschillende kleuren er op het scherm staan
P1: We doen een screenshot test toch? Je zou daar wel ver genoeg in kunnen gaan dat je een screenshot maakt en voor alle colorblindness het wel compatible is
P2: Maar dat valt daar inderdaad ook onder, accessibility
P3: Je had het over retentie net, moeten we zeggen van aantal mensen die succesvol tot een transactie komt
P2: Dat is natuurlijk een ding, je kan bepaalde acties meten, maar dat zijn altijd custom metrics, dus dat kun je nooit over software development als geheel brengen, want niet elke softwareproduct is een webshop. Ik kan wel zeggen van hoeveelheid mensen die een

bepaalde KPI bereiken, maar dat zegt niet perse iets over de software, maar meer over het ontwerp
P3: Ik zou wel hierbij zeggen, hebben we ergens anders ook, aantal errors in de frontend
P2: Ja zeker
P3: Errors moet ik zeggen
P2: Ik denk dat dat ook overlapt met performance, en laad tijd
P3: En ik zou ook zeggen API
P1: Het aantal terugkomende gebruikers
P2: Ja, recurring users
P1: Is misschien net iets anders dan de rétention
P2: Ja, zij zeggen misschien active users, recurring users yes, min of meer hetzelfde. Focust wel heel erg op het product wat ze hebben
P3: Er staat daar iets over effectiveness en effiency, als het heel langzaam is de applicatie. De laad tijd
P2: Ja time to interactive, ik zou het er gewoon lekker overheen zetten. Als het niet wilt laden, als het vast loopt, dan is het niet usable
P3: Ik zet er dan ook even bij, API responstijd. Ik heb ook iets voor mobile apps
R: Laten we daarna wat closing remarks maken
P3: Offline mode, or not
P2: Dat geldt ook voor webapplicaties tegenwoordig. Service workers
P3: Je wilde afronden?
R: Laten we ze afronden. Oke heren, super bedankt voor al jullie info
P3: Jij bedankt
R: Dan ga ik nu de recording afsluiten