Verslagen van de Lean coffee aan 8 tafels

TestNet Thema-avond “Hoe praat ik over testen?”

Nieuwegein 20 November 2019

Capture Testnet meeting 20-11-2019 - Tafel 1

Sanne Visser, Kevin Groot, Isa El Doori, Jan Feenstra, Astrid Bleeker, Hilda Philipse, William Looman, Ries van Aken

Ronde 1, onderwerp 1

Testen is: de juiste aanpak op de juiste plek

- Juiste techniek op de juiste plek
- Idem voor Way of Working

Mensen overtuigen van de aanpak, maar als ik mensen niet mee krijg dan is die aanpak niet juist.

Jouw expertise heeft toegevoegde waarde!

Als de klant er anders over denkt (‘zo doen we het altijd’), dan hebben wij als testers de plicht om de klant mee te nemen in een betere testaanpak. Wel moet je daarvoor de juiste positie hebben.

Ronde 1, onderwerp 2

Kijken of ik ’t kapot kan maken voordat de klant ’t doet

Het is een sport om klantgedrag na te bootsen. Testers zijn hier heel goed in.

Gaan we de ontwikkelaars pesten? Zo ver willen we niet gaan. We proberen ervoor te zorgen dat ze kritisch naar hun eigen werk gaan kijken. Let daarbij ook op non-functionals.

Klanten doen altijd dingen die je niet verwacht. Richt je dus vooral niet op de happy flow als tester. Als die namelijk niet goed is, houd dan maar op. Dat is toch minimaal nodig.

Testen is meer explorerend, Business erbij betrekken. Leuk! Pairtesting bijvoorbeeld. Kan ook met een developer.

Ronde 1, onderwerp 3

Tester heeft een verbindende rol in het team

De tester kent vaak (wél) alle ins en outs van het team. En is ook in staat het te vertalen naar de Business. Dat gaat ook op in waterval trajecten.

‘Kwaliteit’ is niet alleen iets voor de testers, maar voor het hele team.

De tester haalt bij iedereen informatie op, bouwt bruggen en slecht muren.

Ronde 1, onderwerp 4

De juiste tool gebruik je voor de juiste toepassing.

Blijven leren, ontwikkelingen m.b.t. inzet van tools gaan razendsnel en er zijn ook heel veel tools. Alle tools moet je kennen. Meegaan in de flow. Chill!

Er zijn idd veel tools, maar er is ook veel tweestrijd. De ene wil het ene tool en de ander de andere. Terwijl dat eigenlijk helemaal geen interessante discussie is. Het maakt namelijk helemaal niet uit welke tool je gebruikt. Een tool is een hulpmiddel. Leer vooral de principes achter het tool!

Onderhoudbaarheid van een testframework is erg belangrijk!

Ronde 2, onderwerp ‘De tester moet zich verdedigen’

Sticky 1

Stoïcisme:

- Wisdom
- Courage
- Justice
- Etc.

Het principe wordt bijvoorbeeld ook gebruikt bij AA meetings. “Help mij te accepteren waar ik het niet kan veranderen, loslaten, etc. <Sanne: kun jij dit aanvullen? Ging te snel voor mij om bij te houden>.

Rechtvaardigheidsgevoel is hoog.

Waarom bij een organisatie werken waar er geen waardering is voor een tester? Maar ja, wat als je niet weg kán (door omstandigheden)?

Copingmechanisme: waardering zoeken.

Waarom moet ik mezelf toch steeds verantwoorden? Toon courage!

Sticky 2

Alles automatiseren? Wie bedenkt dan wát er geautomatiseerd moet worden?

Het gaat om veel meer dan automatiseren. Gaat om regressietesten.

Developers denken vaak dat je alles kunt automatiseren. Verdedig jezelf door je werk goed te doen. Meer bugs vinden. Developers ook mee laten kijken. Goed framework bouwen en laten zien aan developers hoe goed ’t in elkaar zit.

Je kunt wel alles performancetesten, maar wat wil je daarmee bereiken?

Happy flow automatiseren, maar de rest niet.

Sticky 3

Call his/her bluff

Achteraf aantonen dat iets fout was. Tegenwoordig moet het echter veel meer aan de voorkant gebeuren (dus niet achteraf). Als het van begin af aan al niet goed zit dan dáár de vinger op leggen.

Call your bluff op het whiteboard. Nuttige feedback vooraf geven.

Exploratory Testen is heel belangrijk, maar daar is ervaring voor nodig. Daarnaast is een vernieuwende/frisse blik ook nodig.

“Laten we het een sprint proberen, daarna zal ik laten zien wat de meerwaarde was”

Pas op met een vijandige houding richting de developer. Naar een PO of een manager kan dat wel meer.

Sticky 4

2 testjes is toch nog genoeg?

Door de snelheid wordt de chaos steeds completer.

Die testers zijn alleen maar vervelend.

Stereotypen zijn niet veranderd in de loop der jaren. Testers moeten zich áltijd bewijzen dat het waarde heeft om te testen.

Software testen is niet tastbaar, tastbare resultaten ontbreken.

Coverage van wat je weet is evt. nog wel aan te tonen (ook nog best lastig) maar coverage van wat je niet weet is totaal niets over te zeggen. En wat je niet weet is helaas verreweg het grootste deel …

Soms zijn 2 testjes wél genoeg. Al was het alleen maar om aan te tonen dat er meer testen nodig zijn …

Tafel 2 Verslag Testnetavond 20 november 2019

Ronde 1: wat is testen en hoe leg ik dat uit

Geeltjes:

- ❺ Bruikbaarheid
- ❹ Communicatie
- ❷ Zo veel mogeliijk kwaliteit met zo weinig mogelijk moeite
- ❷ Automatisering
- ❶Documentatie
- ❼ Klantwens
- ❼ Verwachtingen
- ❼Testtooling
- ❼ Testbaarheid
- ❼ Focus
- ❼ Cultuur (internationale omgeving)
- ❼ Risicobeperking
- ❼ Teamsamenstelling
- ❼ Requirements
- ❼ Sprint lengte?

❺ Bruikbaarheid

- Voor wie maken we de software => business
- Voor (de klant) is het belangrijk dat het werkt, het 'hoe' is niet van belang voor de klant
    - Het product moet de klant helpen
    - op het product moet met op terug kunnen vallen
- Van belang is dat het resultaat goed is vanuit het standpunt van de gebruikers
- Probleem is vaak dat anderen de beslissingen nemen dan de personen die er mee gaan werken, het nodig hebben
- De vertegenwoordiger (PO) dient te bepalen voor de business wat er gebouwd moet worden
- Is lastiger als de groep gebruikers bestaat uit 'kopers' van een product en daardoor geen beslissende stem hebben kover het product
- Je weet vaak achteraf pas of je gelijk had over of je het juiste gebouwd hebt
- Zou je het zelf gebruiken als het zo in productie staat? (als tester)
- Een ieder heeft er een eigen mening overtuigen
- Terug naar de gebruiker

❹ Communicatie

- Gebrek aan communicatie
- Halfbakken beschreven oplossing in userstories
- Geen definition of ready of hier wordt niet aan voldaan
- Aanpassen testplan/teststrategie gedurende de sprint
- Op welk deel van de communicatie gaat het mis?
    - gewenste oplossing is niet beschreven vanuit de klant

    - gekozen wordt voor "one solution fits all" waarvan het moeilijk is duidelijk de oplossing te bepalen
    - het helpt om het om te draaien, wat heb je als klant nodig en de klant laten meepraten
- Verschil bestaat tussen: wordt het goede gemaakt of wordt het goed gemaakt
- Als het slecht beschreven is dan is het lat=stig goed te testen/bouwen
- Gebruik maken van contact met klanten vanuit het team (bijv. architecten) of een klantengroep
- Maak het werk van het team beter zichtbaar voor de klant

❷ Zo veel mogelijk kwaliteit met zo weinig mogelijk moeite

- Kwaliteit maak je niet door te testen maar je maakt het wel (beter) zichtbaar
- Heeft te maken met de duivelsdriehoek (geld-kwaliteit-tijd)
- Het is mogelijk het slimmer te testen => meer testen in minder tijd
- Als tester kun je de kwaliteit enkel vaststellen maar
    - zonder testen kun je de fouten niet oplossen, zie je ze niet
- Je staat er als tester niet alleen voor wat betreft kwaliteit, ook een ontwikkelaar (het hele team) en de klant hebben er baat bijhouden
- Risico's worden goed inzichtelijk gemaakt
- Het meten van kwaliteit is voor een groot deel afhankelijk van klantperceptie
- Kwaliteit kan gemeten worden met tooling (bijv. sonar) maar geeft antwoord of je het goed doet maar niet of je het goede doet
- Onderzoek naar gouten en bugs na live-gang, wordt dat gedaan? Ja, want:
    - kunt zoeken hoe de fouten er in komen
    - komt er achter dat niet alle belanghebbenden/gebruikers voldoende zijn betrokken => kijken in welke fase je de fouten had kunnen vinden of de gebruikers had kunnen betrekken
    - Is een goede GAT uitgevoerd?
    - Zijn de juiste vragen gesteld?

❷ Documentatie

- Wanneer is het goed of niet goed?
- Vormt een startdocument (functionaliteitsbeschrijving) waar je je testen op baseert
- Requirements en acceptatie criteria vallen hier ook onder (userstories die verbeterd worden in refinements)
- of een beschrijving hoe iets zou moeten werken
- Testen kan op al de bovenstaande documenten gedaan worden
- Wat te doen als er onvolledige documentatie is?
    - prototyping/schetsen maken
    - voorkomen dat het verkeerde wordt gebouwd
    - BA + UX + FE: schets maken van nieuwe scherm en laten beoordelen. Benodigde documentatie komt later
- Gedeelte via ET om het systeem te verkennen en te zien wat het doet
- Documentatie is moeilijk te vertrouwen omdat het niet mee verandert
- Wordt niet gelezen of uit de code gehaald
- Alleen het essentiele staat beschreven
- De manier waarop je werkt wordt zelden beschreven

- Ook in documentatie het KIS-principe toepassen, scheelt op de lange termijn kosten en kan door testen (reviewen) gedaan worden

❶ Automatisering

- Het testen wordt geautomatiseerd dus waarom zou je nog handmatige testen moeten doen?
    - Testers heb je dan niet meer nodig
    - het betreft hier checken en geen testen
- Onderhoud en beheer van de automatische testen wordt vergeten
- Het maken van de scripts vergt veel tijd
- Automatisering is een onderdeel van de life-cycle van het ontwikkeltraject
- Kunst is om de automatische testen zo eenvoudig mogelijk te houden

<u>Ronde 2: hoe verdedig ik mijzelf</u>

- Verwachting, veelal buiten het team, is dat je als tester 'comb-shaped' moet kunnen zijn, oftewel een duizendpoot die al het werk in een team goed kan uitvoeren
    antwoord: De ervaring die daarmee gevraagd wordt kun je niet zomaar leren. Wat wordt er dan van de rest van het team verwacht?
- Voorbeeld: tester + BA ontwierpen en schreven de testen waar de ontwikkelaar vervolgens de automatische scripts mee ging maken. Werkte erg goed
- Bij de 'klant' heerst vaak een beeld dat je als tester ook gaat programmeren maar in werkelijkheid ben je in het team veel meer analisten werk aan het doen
- Voor een ontwikkelaar is het lastig een keuze te maken wat wel of niet getest moet worden. Wat kan helpen is eerst ET uit te voeren en daaruit de te automatiseren testgevallen te bepalen
- Zou je willen weten wat er aan unittesten is en hoe deze gemaakt worden?
    - weinig/niets weten van de unittesten is 'waste'
    - maak unittesten een gespreksonderwerp in het team zodat iedereen het belang ervan inziet en er gezamenlijk gekeken wordt hoe daar het beste resultaat uit kan worden verkregen
- Situatie bij het overgaan naar een ander platform en een PvA voor het testen te maken met daarin opgenomen de unittesten. Dmv reviews door de testers het plan opleveren
    - Vraag: Kun je daar als team dan invulling aan geven en geeft dat een tevreden gevoel van een goede aanpak

Tafel 3

Thema-avond "Hoe praat ik over testen?"

Tafelnummer	3
Facilitator Naam	Julian Baars

Lean Coffee - Ronde 1:

1 - Waarin onderscheid de tester zich in het scrumteam?
2 + 4 - Test Automation is een einddoel (alles automatiseren) of hulpmiddel? Hoe leg je dit uit?
3 - Duidelijk maken - hoe ga ik testen?
5 - (Hoe) voldoen (we) aan de klantwens?
6 - Wat zijn de problemen bij testen?
7 - Dezelfde taal spreken in het scrumteam
8 - ‘Wat heeft de klant nodig’ vs ‘wat wil de klant’
9 - (On-)zin + effectiviteit van test automation
10 - Wat doet de tester, waarvoor en waarmee?
11 - Hoe maken we duidelijk wat de rol van de tester juist wel en vooral ook niet is?
12 - T-shaped of niet?

Besproken items na voting via dots:

8 - ‘Wat heeft de klant nodig ’vs ‘wat wil de klant’ - 5 dots
Maatwerk? Is oplossing wel inpasbaar? Is dit wel de juiste oplossing?
Communicatie is noodzakelijk
Geen tot weinig klantcontact helpt niet

9 - (On-)zin + effectiviteit van test automation - 4 dots
Wat wil je met test automation bereiken?
Snelheid
Beter resultaat

Hoe zet ik test automation op?
Start met een POC

11 - Hoe maken we duidelijk wat de rol van de tester juist wel en vooral ook niet is?
Framing van de tester, duidelijkheid scheppen
Ligt aan welke omgeving je zit! Traditioneel vs Agile/Scrum

Lean Coffee - Ronde 2 - Track Wat doet een moderne agile tester?

1 - Monitoring van test automation
2 - Nog steeds scenario's schrijven / analyseren
3 + 4 - Aan het begin van de sprint staan; Samen bepalen wat de sprint is
5 + 6 - Automatisch scripten / Automatische test maken
7 - In korte sprint werken
8 - Omgeving bewust maken dat kwaliteit van iedereen is
9 - Klanten en programmeurs op weg helpen naar een doel
10 - Op het juiste moment de juiste vraag stellen
11 - Bijsturen waar nodig, wel altijd opbouwend
12 - Wat bereik je met agile testen niet / wel tov traditioneel testen?
13 - Praten met functioneel beheerders / stakeholders

Besproken items na voting via dots

12 - Wat bereik je met agile testen niet / wel tov traditioneel testen? - 6 dots
Agile is hip!
Wat kan ik er niet mee?
Omgevingsafhankelijk

8 - Omgeving bewust maken dat kwaliteit van iedereen is - 5 dots
Kwaliteit is van iedereen
Iedereen bewust maken van kwaliteit

5 + 6 - Automatisch scripten / Automatische test maken - 3 dots
Wanneer doe je test automation? Wat automatiseer je? Wie bepaalt wat je gaat automatiseren?
Bij Agile hoort regressietesten
Wel veel onderhoud

3 + 4 + 9 Aan het begin van de sprint staan; Samen bepalen wat de sprint is; Klanten en programmeurs op weg helpen naar een doel - 3 dots
Invloed als tester aanwenden

Tafel 4

Ronde 1 “Hoe praat ik over testen?”

Onderwerp 1: Risicoanalyse

Hoger risico is meer testen (zou basis moeten zijn)

Er zijn twee verschillende soorten risico: Projectrisico en productrisico. Projectrisico’s hangen af van of meerdere dingen zoals of er iteratief gewerkt wordt of dat er deadlines zijn. Bij productrisico’s ga je vaak uit van hoe vaak een fout kan voorkomen en hoe groot de impact zal zijn.

Context bepaald welke kwaliteitsattributen belangrijk zijn. Je kunt werken vanuit kwaliteitsattributen of vanuit de stakeholders. Vaak staat functionaliteit bovenaan en de rest is vaak een kwestie van uitzoeken.

Wat de stakeholders ‘bedreigd’ zijn de risico’s.

Hoe ga je om met risico’s in een agile-omgeving?

De meesten willen door de komst van agile minder meewerken aan uitgebreide statistieken. Voorheen werd er een hele analyse gemaakt en kreeg je weinig tot geen feedback. Nu worden er 3 kwaliteitsattributen gekozen en daar wordt dan feedback op gegeven.

Ook kun je gebruik maken van een Burndown-chart voor de risico’s.

Een andere optie is vragen aan key-users “wat kan er fout gaan?” en ze te laten testen.

“Is er tooling om risico’s in kaart te brengen?”

Voor web-apps zou je gebruik kunnen maken van Google/ Adobe Analytics, Power BI of queries op de database. “Waar is de meeste activiteit?”

“Praat met iemand in het team die kennisdrager is voor de eerste opzet.”

(kennisdrager = iemand die het systeem of de applicatie heel goed kent)

Het kan voorkomen dat sommige risico’s niet echt risico’s zijn. Soms kan het helpen als iemand anders het toetst en beoordeeld.

Het is wel altijd beter om iets te testen wat later minder belangrijk blijkt te zijn, dan iets over het hoofd te zien.

Onderwerp 2: Testtechnieken toepassen

Vaak worden ze niet expliciet maar impliciet gebruikt zoals bijvoorbeeld grenswaardeanalyse.

Agile geeft minder tijd om de technieken volledig toe te passen.

Tegenwoordig is er in agile niet echt een test design. Er wordt vaak gekeken naar “de groene geautomatiseerde tests”

Betrek developers en ga in gesprek zodat je de developer op een laagdrempelige manier testtechnieken bijbrengt.

Onderwerp 3: Hoe teststrategie bepalen?

Op epic- niveau, user-story-niveau en feature-niveau moet het bepaald worden.

Er zijn geen vaste rij tests als security- of performance tests, maar je kijkt wat er nodig is.

Tijdens amigo-sessies bepalen en altijd de Product Owner betrekken.

Het begint allemaal bij de refinement en planning.

Ronde 2 "Wat doet een moderne agile tester?"

Onderwerp 1: Testcoaching

Niet zozeer zelf doen maar door vroeg betrokken te zijn developers motiveren om zelf te testen met bijvoorbeeld unit tests. Kwaliteit benoemen. Developer en gebruiker kunnen ook testen, maar de tester is en blijft het aanspreekpunt van kwaliteit.

Maar coachen kan niet iedereen.

"Hoe ga je vanuit een niet-authoritaire rol hier advies over geven?"

Mindset bijbrengen. Door het met elkaar te bespreken. Neem de developers mee en vraag "Wat zijn de drempels voor je om te testen?"

We moeten ook proberen te leren van de bugs: "Waarom kwam dit defect voor?" En dit bij reviews te bespreken. Maar het is belangrijk om niet als politie te fungeren. En houd er rekening mee dat het tijd kost.

Onderwerp 2: Testautomatisering en technisch skills

Ontwikkelaars kunnen ook automatiseren. En keyword-driven maakt het makkelijker voor veel testers om toch te automatiseren zonder al te veel technische kennis.

Sommigen vinden dat je beter kan testen als je ook kan programmeren.

Je moet de testautomatiseringsprincipes wel goed kennen. Als je tegenwoordig geen ervaring hebt met testautomatisering wordt het moeilijk om aan de bak te komen.

Het is vaak zo dat als de vaste tester weg gaat bij een team de testautomatisering omvalt.

Vaak zijn de tests ook te uitgebreid. Iemand wil een onderdeel checken maar controleert ook niet belangrijke dingen. Onderhoudbaarheid moet door het team gewaarborgd worden.

De testautomatisering zou behandeld moeten worden als ieder ander stuk software en een development-traject moeten volgen.

Technische skills zorgen ook voor een drempel. Een manager wil dat de tester bepaalde tools kent anders wordt hij/zij niet aangenomen. Het is dus belangrijk om te blijven leren.

Jezelf anders positioneren is ook van belang. Wat noem je jezelf? Functioneel tester is niet gewenst. Je doet namelijk veel meer dan dat.

Door jezelf goed te presenteren kan je alsnog ergens binnen komen.

Onderwerp 3: Teststrateeg

Tester wordt/is het morele kompas. Niet meer alleen maar aan het testen achter zijn/haar pc.

Vroeg betrokken zijn en verder kijken dan je eigen team en ook kijken naar bijvoorbeeld andere teams, je eigen bedrijf, concurrenten en de wet.

Vereist wel een bepaald niveau van ervaring, maar het wordt wel verwacht dat de tester het voortouw neemt.

En daar lijkt meer behoefte naar te zijn in teams. Soft skills zijn heel belangrijk.

Lean Coffee tafel 5

Track 1 - discussie gaat over:

“Hoe praat ik over testen en hoe leg ik testen uit”

Genoteerde onderwerpen op volgorde van stemming zijn:

1. Meer aandacht voor velocity dan voor kwaliteit.
2. Testresultaten;
   - wat is een goede aanpak om onder tijdsdruk testresultaten te rapporteren?
3. Hoe ga je om met tijdsdruk?
   - Hoe leg je aan ontwikkelaars uit dat ze nog niet klaar zijn want er moet nog getest worden?
4. Auto testen, waarom?
   - Hoe leg ik aan ontwikkelaars en stakeholders uit dat testautomatisering wel/niet altijd nodig is?
5. Testen in multinationale setting op meerder locaties.
6. Testen onder security restricties;
   - hoe communiceer je met over testen onder strikte security restricties?
7. PRISMA – alles is hoog!
8. Cultuur Clashes in test.
9. Test executie, uitvoeren van testcases.
10. Praten met collega’s.
11. Impact in kaart brengen.
12. Duwen op testen om snel live te kunnen gaan.
13. Geen aandacht meer voor risico inschatting -> dus wat en hoe testen ook lastig inschatten.

Besproken onderwerp: 1. Meer aandacht voor velocity dan voor kwaliteit

- De enige die op tijd leveren met voldoende kwaliteit (door overwerk).

- In de initiële planning rust momenten opnemen.

- “jij kan nu gaan testen. Dan gaan wij alvast de refinement voor de volgende sprint doen”.

- Wie accepteert het opgeleverde?

- Er wordt gewerkt met een sprint van een 1 week.

- Men geeft geluk dat bepaalde “bugs” nog niet boven water zijn gekomen.

- Wie zijn je medestanders?

- Ontwikkelaar (externen) moet opleveren en naar productie gaan want daar worden ze op afgerekend.

- Tester is verboden een “brede” blik te hebben.

Besproken onderwerp: 2. Testresultaten

- Wat is een goede aanpak om onder tijdsdruk testresultaten te rapporteren?

- Hoe deel je de testresultaten? Hoe doen jullie dat?

- In de demo 1 powerpoint sheet met daarop het “onderzoek” dat ik als tester heb uitgevoerd, de bevindingen (die hoeven niet alleen bugs te zijn), een conclusie en een advies.

- Testrapport met daarbij aangeven wat het belangrijkste hoofdstuk is (conclusie en advies).

- Van te voren vastgelegde testscenario’s met de acceptant uitvoeren en laten goedkeuren.

- Moet je bij ET zwaar inzetten op vastleggen? Er is dan heel veel getest en de conclusie is dat het goed is maar hoe leg je dat vast?

- Tijdens het testen vastleggen wat je uitgevoerd hebt. Gebruik Charters, formuleer het testdoel “Test of ….” , ga uit van de aangegeven hoge risico’s en rapporteer wat je daarbij onderzocht hebt.

- Tijdens het testen vastleggen van je stappen is alleen interessant als er ook een bug gevonden is.

- Als het risico al heel hoog ingeschat is wordt er meestal wel meer tijd voor gereserveerd waardoor je meer formele testontwerptechnieken kan toepassen.

Track 2 - discussie gaat over:

“Hoe verdedig ik mijzelf”

Genoteerde onderwerpen op volgorde van stemming zijn:

1. Functioneel tester vs. Technical tester.
2. Meerwaarde tester;
   – hoe leg ik aan anderen (en ook buiten het werk) uit wat de meerwaarde van een tester is?
3. Product owner;
   – hoe in gesprek te gaan met de PO als de tester vind dat deze meer tijd nodig heeft.
4. (Gebrek aan interesse) Geen noodzaak mezelf te verdedigen;
   - er wordt nooit doorgevraagd op de opgeleverde testrapportage,
   - gebrek aan interesse maar testen wordt wel heel belangrijk gevonden.
5. Dev Skills? Heb je die als tester nodig?
6. Ontwikkelaars;
   - hoe in gesprek te gaan met ontwikkelaars als het om een bevinding gaat.
7. Het is net een (eind)examen.

Besproken onderwerp: 1. Functioneel tester vs. Technical tester

- Laat je horen, door vervelende en kritische vragen te stellen waar ze over na moeten denken, vraag ook door.

- Als ik de user story al op drie manieren kan uitleggen, dan verzint de ontwikkelaar wel een vierde manier.

- En wat als de gebruiker dit doet? En wat nou als er dat gebeurt?

- Het is wel handig om code te kunnen lezen want dan kan je naast de ontwikkelaar gaan zitten en vragen stellen.

- Als tester ga ik niet leren programmeren want dan ga ik ontwikkelaars-taal spreken en ben ik niet meer onafhankelijk.

Besproken onderwerp: 2. Meerwaarde tester

– Hoe leg ik aan anderen (en ook buiten het werk) uit wat de meerwaarde van een tester is?

- Funtionele tester letten ook op styling en lay-out

- functionele tester hebben veel contact met de gebruiker en daardoor heel veel domeinkennis en echte kennis over hoe het systeem gebruikt wordt.

- Trias Politica: de wetgevende macht is de klant, de uitvoerende macht is de ontwikkelaar, de controlerende macht is de tester.

- Test automatisering gaat altijd via hetzelfde pad(den) door het doolhof heen.

- Testpiramide; hoe hoger je in de piramide komt, des te minder je hebt om te automatiseren omdat alles al geautomatiseerd is.

- Repeterende testen zijn saai en het zou mooi zijn als deze zoveel mogelijk geautomatiseerd worden.

Besproken onderwerp: 3. Product owner;

– hoe in gesprek te gaan met de PO als de tester vind dat deze meer tijd nodig heeft.

- Als tester kijk je vaak breder dan wat in de user story staat, met goede argumentatie kan de PO om meer tijd vragen en deze krijgen.

- argumentatie op impact.

- Je bent een steunpilaar en adviseur voor de PO.

- Nooit hoeven verdedigen als tester.

- Je accepteert samen met de PO bepaalde bugs, die dan wel op de backlog gezet worden. Maar ze weten beiden dat deze vanwege de tijdsdruk nooit opgepakt gaan worden.

Tafel 6 Verslag Testnetavond 20 november 2019

<u>Ronde 1: wat is testen en hoe leg ik dat uit</u>

Geeltjes:

- ❺ Explain value of testing / wat verkopen we als testers
- ❹ Ervaren ontwikkelaars versus beginnende testers
- ❷ Handmatige testen automatiseren werkt niet
- ❶ Hoe leg ik uit wat testen is aan iemand buiten de IT
- ❶ Wat niet getest is werkt niet
- ❶ Agile testen anno 2019
- ❼ Vage/niet gedefinieerde testcriteria
- ❼ Testen is de PO zekerheid geven inzicht om te beslissen om naar productie te gaan

❺ Explain value of testing / wat verkopen we als testers

- Zonder testen kun je niet zeggen wat goed is/voldoet
- Vaststellen van de kwaliteit van wat naar productie gaat obv acceptatiecriteria
- Specifieke informatie over het product
- Acceptatiecriteria helder krijgen en checken of eraan voldaan is
- Niet-happyflow denken
- Andere perspectieven om naar het product te kijken
- Contextafhankelijk
- Vertrouwen verkopen dat het werkt
- Inzicht over de status van het product obv van wat een klant belangrijk vind om te weten

❹ Ervaren ontwikkelaars versus beginnende testers (testers kunnen het niet bijhouden)

- Bij de scrummaster leggen als impediment
- Laat een ervaren tester/testcoach meelopen
- Maak duidelijk wat de verwachtingen zijn en of die met deze groep testers kunnen worden waargemaakt
- Laat bouwers meetesten
- Test samen met de ontwikkelaars
- Ga in gesprek met de PO over wat jullie wel kunnen
- Inzetten van Behaviour Driven Development /Specification By Example
- Ga van het testeiland en betrek het team erbij
- Huur een testcoach in
- Hou 3-amigo sessies
- Leer je applicatie kennen, dan kun je sneller testen

❷ Handmatige testen automatiseren werkt niet (je krijgt flaky tests)

- Denk eerst na over wat je wilt bereiken met testautomatisering
- Maak een strategie
- Denk na over je tools en doe een aantal POC´s
- Doe je geautomatiseerde testen in de sprint (en niet erna)

- Denk ook na over performance en security testen
- Wat stop je wel of niet in je integratietesten en wat functioneel in de UI
- Kijk ernaar vanuit een ander perspectief, b.v. de keten of klantpaden
- Laat het over aan dedicated testautomatiseerders

❶ Hoe leg ik uit wat testen is aan iemand buiten de IT

- Je rijdt in een auto, die is getest (daar ga je vanuit), hetzelfde geld voor software
- Vergelijk het met iets dat mensen herkennen
- Je controleert of alles juist is, je salaris wil je aan het eind van de maand ook ontvangen en het moet ook nog juist zijn
- Vergelijk het met niet-IT gerelateerde voorbeelden, b.v. het bouwen van een huis
- Stel de vraag: gebruik je zelf wel eens software en loop je dan wel eens tegen problemen aan? Wij proberen dat te voorkomen door vooraf al actief op zoek te gaan naar die problemen
- Vertel een smeuïg verhaal over een leuke bug die je zelf wel eens gevonden hebt. Hoe beeldender het verhaal is, hoe beter.

<u>Ronde 2: hoe verdedig ik mijzelf</u>

Geeltjes:

- ❺ Minder waardering voor testers
- ❹ Hoe (test)volwassen is je organisatie
- ❸ Het gesprek aangaan/Hard er tegenin gaan
- ❷ Tegen verkeerde verwachtingen
- ❶ Niet (zonder gesprek)

❺ Minder waardering voor testers (minder betaald, lagere schaal)

- Geef in een gesprek met de lijn aan wat je waarde is en wat er zou gebeuren als jullie er niet zouden zijn
- Onbekend maakt onbemind. Blijkbaar kunnen we onvoldoende uitleggen wat ons belang is als tester (zie ronde 1)
- Ga met alle testers als collectief het gesprek met management aan of stap allemaal collectief op
- Mensen hebben weinig kennis over testen, maar wel vaak een mening
- Maak de organisatie testbewust

❹ Hoe (test)volwassen is je organisatie

- Groepeer je richting het management
- Positioneer de tester als ´ik ben een spiegel van wat jij hebt gemaakt/bedacht´
- Gebruik ´don´t shoot the messenger´ in je communicatie
- Krijg je team mee zodat zij jou ondersteunen (de tester mag niet weg uit ons team, want hij/zij levert een enorme toegevoegde waarde)
- Deel je teststrategie met je team en betrek het team ook in het testen

- Als een ontwikkelaar niet test en zaken half oplevert en verwacht dat de tester het wel even test -> ga samen een Unit test / Integratie test doen zodat het de ontwikkelaar ook duidelijk wordt wat hij/zij oplevert.

❸ Het gesprek aangaan/Hard er tegenin gaan

- Maak als team de juiste start ipv de tester als pleister op de wond te gebruiken (quality built-in)
- Gebruik TDD/SBE om te forceren dat er beter getest word
- Gebruik codereviews (dat is ook testen)
- Probeer niet alleen te overtuigen op argumenten, met name managers zijn daar ongevoelig voor
- Vraag wat belangrijk is zodat je daar iets over kan vertellen
- Choose your battles, niet trekken aan een dood paard
- Gebruik net NOSE model om mensen mee te krijgen, dat wil zeggen dat je vanuit de behoefte van de persoon (manager,PO,ontwikkelaar) aankijkt tegen testen
    - Need (wat heeft die persoon echt nodig, welk probleem los je op)
    - Outcome (wat is de gewenste uitkomst, veelal dat het probleem weg is)
    - Solution (wat is de oplossing om in de Need te voorzien)
    - Evidence (lever bewijs dat de solution ook werkt)

Placeholder – Tafel 7

"Lean Coffee - Wat is testen" – Tafel 8

Hoe Awareness krijgen voor Shift-Left, bijv bij Requirements Engineering

Vraag over requirements die onjuist/onduidelijk zijn en via lange weg gaan van Tester -> Developer -> Analist -> P.O. -> Business

- Gebeurt onder druk om te leveren
- 3-amigo's (avant la lettre)
- Vragen stellen
- QA = Questions Asker
- Voorbeelden (laten) geven; Visueel maken
- Gaan doen
- Ervaart het team de last? Kun je support krijgen voor het idee?
- Ga je voor 'Groot en minder compleet' of 'Klein en completer'

Testen is een mindset

Gegeven voorbeeld: Uitnodiging voor een intake op een onlogische combinatie van dag en datum.

- Contra: Moet je wel altijd scherp zijn?
- Lezen wat er NIET staat.

Rapporteren op 3 niveaus gericht op vertrouwen

- Besef dat je verschillende doelgroepen hebt
- Vraagt domein- en ketenkennis

1. Smiley's - Directie
2. Summary - Management
3. Details - Teams/testers

Volgende onderwerpen zijn niet besproken want te weinig stemmen:

- Kwaliteit aantonen
- Aannames toetsen
- Verwachtingen managen
- Zekerheid geven
- Consensus bereiken (over acceptatiecriteria)
- Kwaliteit en risico's vertalen in :(en :)
- Belangrijke bugs/issues vinden voor productie
- Testen = Risico management
- Er voor zorgen dat de verschillen te niet worden gedaan
- Verschillen bepalen hetgeen de software volgens de beschrijving moet doen en doet
- Inzicht krijgen / geven
- Status van product vasstellen
- verbinden

Topics "Lean Coffee - Hoe verdedig ik mijzelf" – Tafel 8

Vragen terug stellen / 'Domme' vragen stellen

- Vragen terug stellen
- Bij een statement, op zelfde wijze van repliek dienen
- Eigenwijs zijn
- Jezelf niet te serieus nemen
- 'Ik ben hiervoor ingehuurd'
- Jezelf niet aangevallen voelen
- Verdedigen of meewaaien, bijv bij DevOps en neiging tot afschaffen testers

De enige zijn in een team met (te)veel developers

- Miniwaterval
- Hoe ga je van de rol tester naar de rol testregiseur?
- Is de enige tester ook de enige die testen doet?
- Begeleid de dev-engineers
- Stuur op kruisverwerking: de een bouwt, de ander reviewt
- Gebruik ervaring
    - doe een steekproef op opgeleverde testen
    - 'Usual suspects'
- Je moet kunnen coachen
- Mijn creatie roept aversie op

Niet

- De niet-automatiserende tester -> golfbeweging
- Positioneer een tester op niveau solution architect voor krijgen overzicht op testactiviteiten

Een andere mindset helpt het team

- *(... een vervolg op het vorige item ...)*

Volgende onderwerpen zijn niet besproken want te weinig stemmen:

- Developer overtuigen
- Testen is een vak :)
- Zoeken naar een toegevoegde waarde en die verkopen
- Win-win-situatie creëren
- Met feiten
- Door een (succes)verhaal te vertellen