# Intro testen

# 1 Wat is Testen??

**Opdracht 1:**

**Wat is de geschiedenis van TMAP?**

TMAP, Test Management Approach, is een testmethode voor softwareproducten die ontwikkeld is in Nederland door Sogeti en voor het eerst toegepast in 1995. Hiervoor was het testen meer gebaseerd op het vinden van fouten en deze oplossen, om zo het ‘perfecte’ softwareproduct te leveren. Bij het ontwikkelen van TMAP heeft men de stap van het oplossen van de fouten geschrapt en zich puur gefocust op het vinden en aanduiden van fouten in het product. In 2006 is TMAP NEXT geïntroduceerd, dit is een meer traditionele manier van testen gebaseerd op de watervalmethode. In 2014 is TMAP HD geïntroduceerd, deze is bedoeld voor Agile-softwareontwikkeling. In 2020 is uiteindelijk ook TMAP Quality for DevOps teams ontstaan, de manier van hoogwaardige IT-levering voor de DevOps-cultuur.

**Wat is de kracht van TMAP?**

De kracht van TMAP is het structureren van het testproces (white- en black-box) van softwareproducten en zorgt ervoor dat de tijdlijnen verkort worden en er dus kosten bespaard worden. Hierdoor worden fouten in een vroeg stadium gevonden. Verder is het inzetbaar in elke situatie ongeacht de opdrachtgever of ontwikkelingssysteem. Verder is TMAP erop gericht om met zo min mogelijk testgevallen de vereiste dekking te bereiken.

**Wat biedt TMAP?**

Tmap biedt het meest optimale testproces. Het geeft ook inzicht en advies over eventuele risico’s over de kwaliteit van het geteste product.

**Waarbij helpt TMAP?**

Omdat er een standaard werkwijze is, zijn testspecificaties overdraagbaar en onderhoudbaar.

**Opdracht 2:**

* **Wat is de definitie van testen?**
* **Wat is het verschil tussen Testen en Toetsen?**

Testen is vaststellen in hoeverre het softwareproduct aan de eisen voldoet. Testen doet men dus aan het einde van het opleveren. Toetsen doet men tijdens het ontwikkelingsproces.

**Opdracht 3:**

**Wat zijn eventuele valkuilen die je in de praktijk kan tegenkomen?**

Ongestructureerde testen zorgt voor chaos. End 2 end testen, je moet automatiseren waar nodig niet waar het kan. Het moet de kwaliteit verbeteren van het proces en niet automatiseren om te automatiseren. Binnen scrum ben je geneigd om in korte sprints te werken en daardoor veel te automatiseren. Hierdoor ben je van tevoren genoodzaakt om goed na te denken over wat ga ik automatiseren en wat niet?

# 2 Waarom is testen belangrijk?

**Opdracht 1:**

**Waarom is testen belangrijk in het proces van software development?**

Testen is belangrijk omdat er dan in vroeg stadium fouten kunnen worden verbeterd en er zo eerder een goed softwareproduct wordt geleverd. Als er niet getest wordt, komen fouten na levering aan het licht en dan is het meestal veel duurder om die fouten op te lossen. Doel van testen is voorkomen is beter dan genezen!

**Opdracht 2:**

**Zoek op het internet naar voorbeelden van waar software mogelijk niet goed genoeg getest was en geef ook aan wat de gevolgen daarvan waren.**

Een voorbeeld wat ik zelf ken in de praktijk is de levering van de game Cyberpunk 2077. Dit is een game waar jaren aan is gewerkt en waar de gaming community reikhalzend naar uitkeek. Probleem is geweest dat de ontwikkeling van de game samenviel met de lancering van next-gen gamingplatformen, denk hierbij aan de PS5 en xbox series x. Het spel draait (op een paar bugs na) fantastisch op de next-gen consoles en op een goede game pc. Echter is de game onspeelbaar op de current-gen (PS4, Xbox one X), waardoor het bedrijf genoodzaakt was om na de meest succesvolle verkoop bij lancering, de verkoop van de game op current-gen moest stoppen en zelfs begonnen is met het terugbetalen omdat het geleverde prodcuct (game) niet aan de eisen voldoet. Hier is het dus misgegaan bij het testen van de game op de current-gen, na zo’n test had de game nooit mogen verschijnen op de huidige generatie spelcomputers.

Een ander voorbeeld is de NEST thermostaat, die na een software update uitviel en ervoor zorgde dat het onmogelijk was om de thermostaat aan te krijgen en dus je huis te verwarmen.

Als laatst een voorbeeld wat veel mensen kennen, is het dieselschandaal van Volkswagen. Er was bij het vaststellen van de uitstoot van roetdeeltjes opzettelijk geknoeid met de software van de auto waardoor deze een testrollerbank herkende en zo dus minder uitstoot veroorzaakte en dus de test saboteerde. Bij een goede gedegen test, was deze opzettelijke ‘fout’ eruit gehaald en had de auto nooit geproduceerd mogen worden. Wat uiteindelijk heeft geleid tot miljardenclaims, kelderende beursaandelen, particulieren die massaal hun auto’s verkochten etc.

# 3 Wat is de toegevoegde waarden van testen?

**Opdracht 1 : Zoek op wat de voordelen zijn van testen?**

De voordelen van testen zijn dat er vroeg tijdens de ontwikkelingsfase fouten en bugs kunnen worden gerepareerd en niet tijdens of na lancering. Dit bespaart kosten en tijd. Testen gebeurt meestal automatisch en is in tegenstelling tot ontwikkelen goedkoop en kost veel minder mankracht. Voordeel van testen is dat er ook problemen kunnen worden getackeld waar in eerst instantie niet aan is gedacht.

**Opdracht 2 : Wat zijn de kosten?**

Het kost meestal tot een kwart van de ontwikkelingskosten. Dit bedrag kan veel lager komen te liggen als er meer automatische tests worden uitgevoerd.

**Opdracht 3 : Wat zijn de baten?**

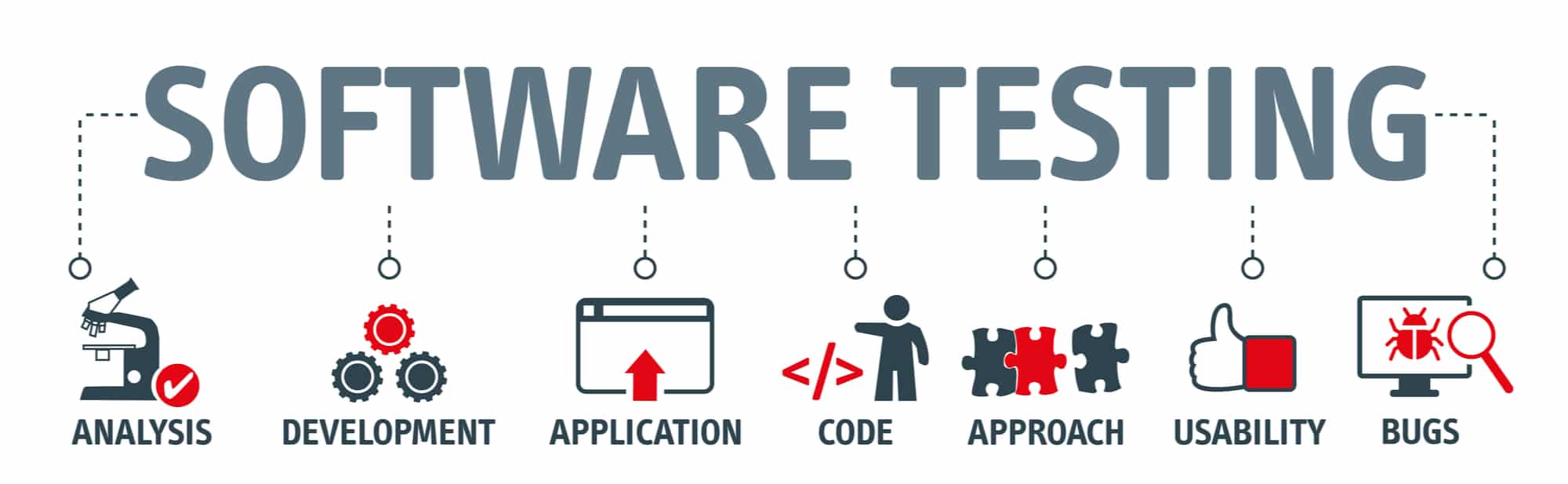
De baten zijn: kosten besparen op fouten na lancering te dichten, een product afleveren wat na goed te zijn getest niet verder ontwikkeld hoeft te worden, hooguit in de vorm van updates, biedt houvast aan de product manager doordat het proces meer tastbaar wordt.

**Opdracht 4 : Formuleer dit verder in wat de Test Opbrengst kan zijn.**

Doordat het test gedeelte wordt uitbesteed aan testers, houden developers, managers meer tijd over voor hun eigen taken. Kwaliteitswaarborging van een softwareproduct wordt gegarandeerd met testen. Fouten die worden opgespoord kunnen in een vroeg stadium worden gepareerd en dus kosten besparend i.p.v. achteraf, wanneer het lastig wordt om de oorzaak van de fout te herkennen.

# 4 Wie voert de testen uit?

**Opdracht 1: Testen is niet alleen uitvoeren. Wat  komt er nog meer bij kijken bij het testproces?**



**Opdracht 2: Beschrijf de verschillende manieren van testen.**

Je hebt functioneel, performance en security testen. Bij functioneel en performance testen denkt men vooral in positieve scenario’s. Bij security wordt er uitgegaan van een negatief scenario. Bij functionele testen wordt getest of de gebruiker de functies die worden geboden door het softwareproduct kan uitvoeren of gebruiken. Iemand die een scenario uitschrijft zal vanuit de ogen van een gebruiker naar het product kijken en daarop een script baseren. Bij performance testen wordt gekeken naar hoe het softwareproduct presteert bij (abnormaal) gebruik van het product. Hiermee verkrijg je inzicht in de user experience maar ook in de belastbaarheid van het systeem. Tot slot wordt er bij security testen gekeken naar de veiligheid van het product, kan de gebruiker van het product info zien die hij niet hoort te zien? Nog belangrijker, kan info van gebruikers door derden gezien worden. Hierbij is een pen test een onderdeel van de security test.

**Opdracht 3: Wie test er?**

Ideaal voeren test engineers testen uit op het softwareproduct. In de praktijk komt het nog vaak voor dat software developers zelf testen uitvoeren of zelfs dat het product bij ingebruikname door gebruikers uitvoerig wordt getest. Uiteraard zijn laatstgenoemde punten niet wenselijk en pleit ik daarom voor meer testen bij ontwikkeling van softwareproducten door professionele testers!

**Opdracht 4: Vind informatie over het Test- en systeemontwikkelproces en beschrijf de invloeden van het V-Model.**

In het V model lopen de ontwikkeling en testen lineair naast elkaar. Voor elke systeemontwikkeling is er een aparte test. Alhoewel afgeleid van de watervalmethode, heeft het V-model tussentijdse testen. Dit is meteen ook het grootste voordeel, elke fase van integratie wordt getest. Nadeel is helaas dat er vanuit wordt gegaan dat de vereisten aan het product niet veranderen, het ontwerp en de vereisten worden niet geverifieerd, de eerste test vindt pas plaats na het ontwerp van een module en niet tijdens.

**Opdracht 5: Beschrijf de linkerkant van het V-model en geef de definitie aan van toetsen. Waarmee kan het resultaat vergeleken worden bij het toetsen?**

De linkerkant van het V-model beschrijft de volgende stappen: Wens – eisen – functionele specificatie – technisch ontwerp – systeemrealisatie. Bij de eisen wordt er getoetst of de eisen wel haalbaar zijn en hoe deze eventueel uit te voeren. Bij de functionele specificatie wordt er getoetst of de functionele en niet functionele verwachtingen naar wens zijn. Bij het technisch ontwerp wordt er getoetst of de functionele en niet functionele specificaties voldoen. Bij de systeemrealisatie wordt er getoetst of de componenten na koppeling nog naar wens functioneren. Deze laatste test wordt meestal door specialisten gedaan.

**Opdracht 6: Welke testsoorten en -verantwoordelijkheden zijn er in een systeemontwikkeltraject?**

Zoals in de vorige vraag beschreven heb je de volgende testsoorten: acceptatietest, systeemtest, programmatest en een integratietest.

**Opdracht 7: Vind nu informatie over de rechterkant van het V-model en beschrijf welke testsoorten hier plaatsvinden en de verantwoordelijke.**

De rechterkant van het v model beschrijft de volgende stappen: verwachting – gebruik & beheer – acceptatietest – systeemtest – programmatest. Voor het de acceptatietest en het gebruik & beheer is de product owner verantwoordelijk. Voor de systeemtest en programmatest zijn de developers verantwoordelijk.

**Opdracht 8: Wat is een testbasis?**

De testbasis zijn alle bronnen waaruit de eisen van een softwareproduct zijn af te leiden. Denk hierbij aan het functioneel ontwerp, het gewenste systeemgedrag, user stories, technisch ontwerp etc.

**Opdracht 9: Welke test vormen zijn er? Wat is een testvorm? Wat is een regressietest?**

Er zijn heel veel testvormen, zie rapport TMAP Rapport NL. Een testvorm is een manier om te bepalen of een product aan een bepaalde test voldoet. Voorbeelden zijn, functionaliteit, Pen test, GUI, Performance, Regressietest etc. De regressietest is een test waarbij na het doorvoeren van een wijziging alle onderdelen van het software product nog correct functioneren.

**Opdracht 10: Wat is gestructureerd testen? Geef hierbij de nadelen van ongestructureerd testen en de voordelen van een gestructureerd testaanpak.**

Gestructureerd testen is testen op een manier zoals TMAP of ISTQB. De voordelen van gestructureerd testen is dat deze toepasbaar is op verschillende projecten. Je vindt in een vroeg stadium fouten. Geeft inzicht in- en advies over risico’s. De testproducten zijn herbruikbaar. In tegenstelling tot ongestructureerd testen waarbij je als tester tegen tijdsdruk aanloopt, geen inzicht hebt in de kwaliteit van het softwareproduct en uiteindelijk dus inefficiënt en ineffectief wordt.

**Opdracht 11: Wat zijn de kenmerken van een gestructureerde testaanpak?**

Het testproces wordt inzichtelijk en beheersbaar. Het voorkomt fouten en testen duren minder lang op kritieke paden.

# 5 Welke test Types zijn er?

**Opdracht 1: Zoek van de categorie Functioneel testen minimaal 5 types en geef hiervan een beschrijving.**

* Unit test: Kleine onderdelen van het product worden getest, bijvoorbeeld een menu balk van een website. Werkt deze naar behoren? Navigeert ook daadwerkelijk naar een andere pagina/sectie?
* Smoke test: Alle basis functionaliteiten van het product worden getest en er wordt gekeken of deze naar behoren werken en dat er genoeg stabiliteit is om verder te ontwikkelen.
* Sanity test: Wordt meestal na de smoke test gedaan om te kijken of alle grote en belangrijke functies van de code wel functioneren. (Als het werkt dan werkt het, maar hoe werkt het?)
* Integratie test: Nadat verschillende onderdelen van de software zijn geïntegreerd wordt er gekeken of deze wel goed werken na integratie.
* Acceptatietest: Bij deze test worden de wensen en verwachtingen van de product owner (eventueel aan de hand van een user story) getest. Hierbij wordt er dus een volledige functionaliteit verwacht van het product.

**Opdracht 2: Zoek van de categorie Niet-functioneel testen 5 types en geef hiervan een beschrijving van wat deze type inhoud.**

* Performance testing: Het software product wordt getest onder een bepaalde belasting om aan te kijken of prestaties binnen de norm blijven.
* Stress test: Hier wordt het product tot het maximale en eroverheen getest om te kijken hoe het zal reageren, in tegenstelling tot de perfomance test die binnen de lijntjes blijft.
* Security test: Hoe goed is de beveiliging van het product, wat als er een DDos aanval wordt ingezet?
* Soak test: Het product wordt langere tijd intensief gebruikt, gekeken wordt er naar de prestaties van het product tov kort gebruik.
* Failover test: Hoe gedraagt het product zich als er een fout is of het systeem raakt overbelast. Hoe herstelt het product zich nadat het weer naar behoren functioneert.

**Opdracht 3: Zoek van de categorie Onderhoud test de voorbeelden en maak hier een beschrijving van.**

* *Correctief onderhoud test: Deze test is de moeilijkste omdat het meestal cruciale elementen van de functionaliteit van het product omvat. Hierdoor functioneert het product niet naar behoren en dient er dus meteen code aangepast worden. Denk aan functionaliteit om foto’s te maken om door te sturen naar de verzekering bij de verzekeringsapp.*
* *Preventief onderhoud test: Hier worden er normale functionele en niet-functionele test uitgevoerd en waar nodig problemen opgelost om zo eventueel onderhoud voor te zijn. Denk aan het vergroten van een servercapaciteit als gevolg van meerdere gebruikers.*
* *Adaptief onderhoud:*

# 6 Wat is testen in software engineering?

**Opdracht 1: Maak een korte samenvatting van 6 punten wat testen inhoud voor de IT.**

**Opdracht 2: Vind de defecten in het volgende voorbeeld en wees gedetailleerd in je antwoord.**



* User Id vs User ID
* User ID is ongewenst qua inhoud
* Linksboven Registration niet afgemaakt
* Kleur verschil header registration
* Margin tussen input balk user id en password te groot
* Sterretjes
* Confirm password zichtbaar
* Onveilig password
* Getallen in name
* Register rechtsonder verschillende fontssizes
* Register rechtsonder zonder hoofdletter
* Email ipv E-mail
* Uitlijning Captcha niet helemaal links
* Captcha niet zichtbaar
* Input captcha letter ipv nummer
* Achtergrond Country India geel ipv wit
* Symbool punt ipv name
* Choose user ID en naast de balk Enter User ID, 2 acties liever alleen het 2e

**Opdracht 3: Vind de defects in het volgende scherm. Dit is een oude google page. (Lay-out inconsistenties, spelling error, etc.)**



* Onderdaan 2 keer done
* Browserbalk pijltje onderaan wordt tweemaal weergeven
* Gooogle met drie o’s in button gooogle search
* I’m feeling unlucky?
* Geen logo google bij tabblad
* News niet in lijn met andere items, web images map etc.
* Zoekbalk geen left border
* Twee werkbalken over elkaar heen bovenaan
* Hopping? In de bovenste balk
* Rechtsboven kruis niet geheel
* Pijltje achter more omhoog ipv omlaag
* Rechtsboven middelste knop niet om te vergroten
* iGoogle?
* Streep niet doorgetrokken onder iGoogle
* Dubbele streep na iGoogle
* Search settings, Settings zonder hoofdletter
* Laadsymbool in werkbalk ipv tabblad pagina
* Automatisch geselecteerde button I’m Feeling Unlucky ipv Gooogle Search
* Uitlijning buttons niet in midden van scherm
* Rechtsonder verbindingstype ontbreekt
* Untitled ontbreekt een haakje
* Browserbalk ontbreekt?
* Home button
* 2e o google geen schaduw rest van de letters wel