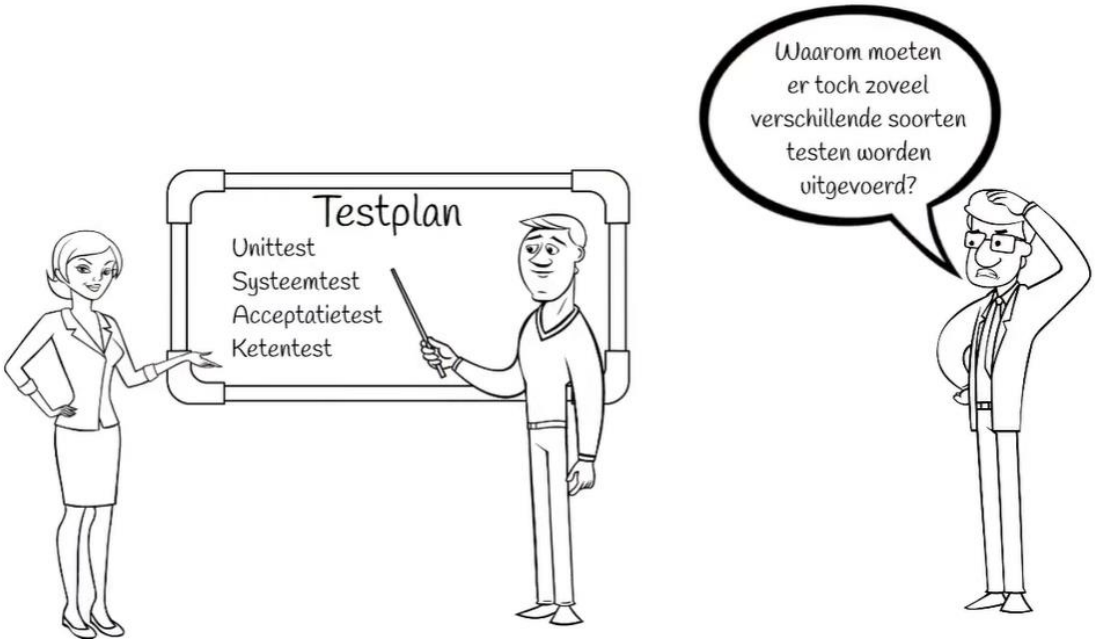
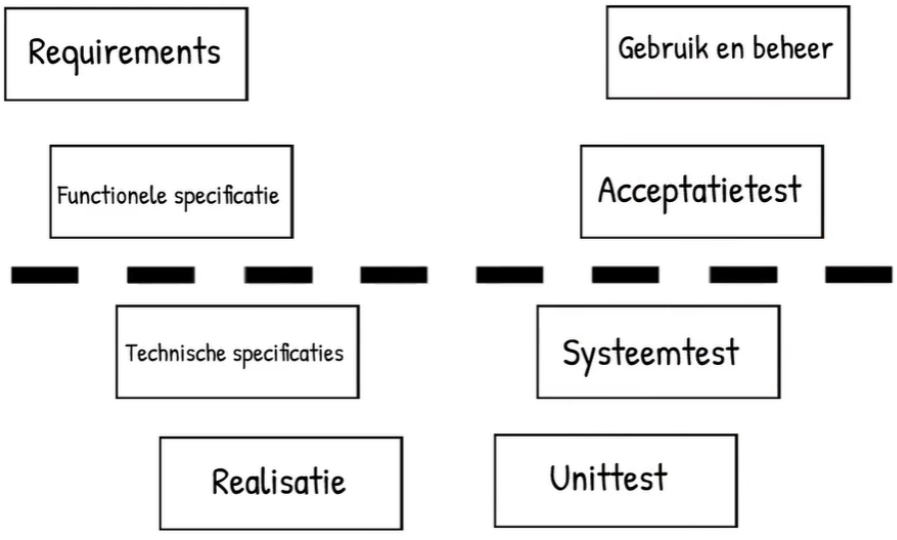
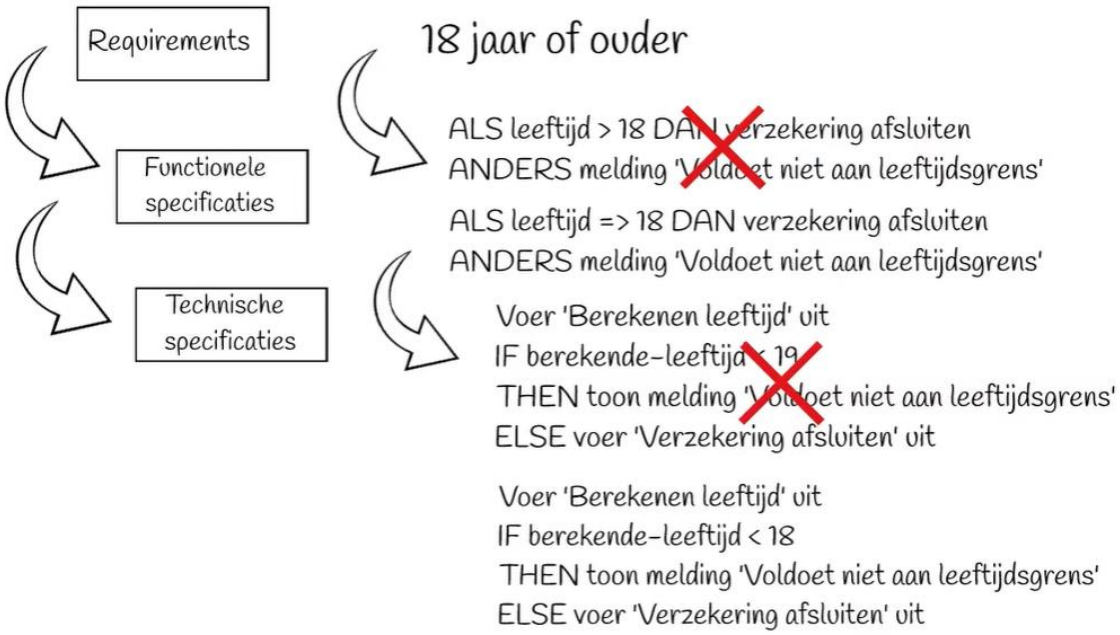
Testen – het V-model

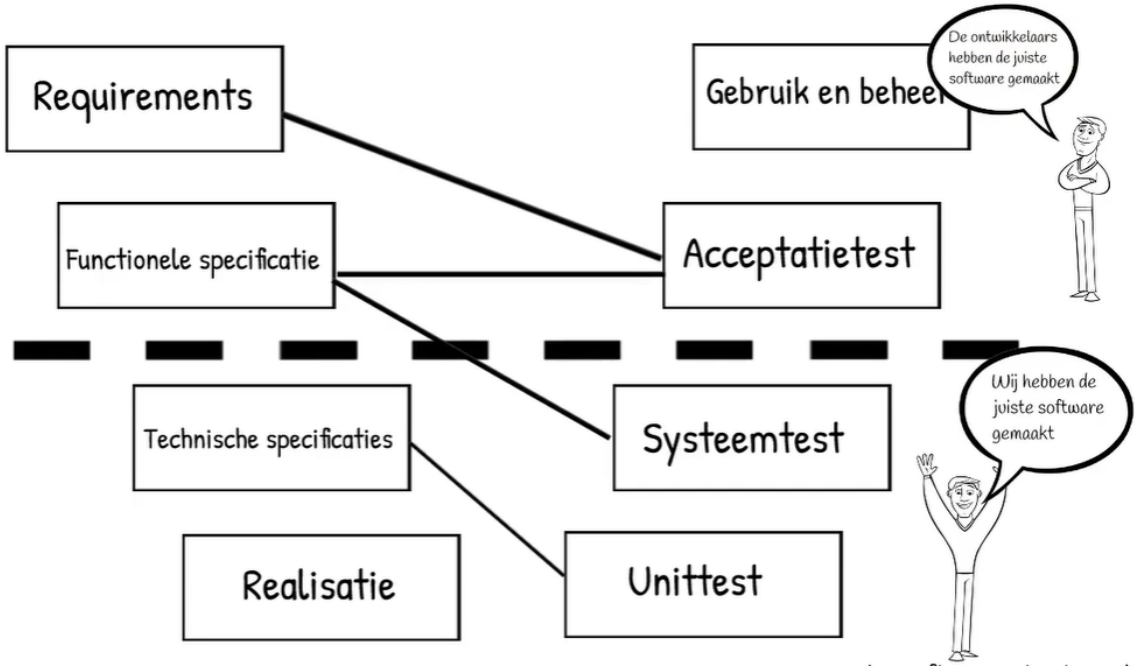




Het v-model

Fouten bij ontwikkelen van software:





Bron: https://www.youtube.com/watch?time_continue=16&v=W-I1pfNmAy0&feature=emb_logo

Testsoorten

In een traditionele ontwikkelomgeving wordt het testen van de software vaak verdeeld in verschillende testsoorten. Deze splitsing heeft veelal te maken met verantwoordelijkheden en belangen.

Definitie
Testsoort is een groep van testactiviteiten die gezamenlijk worden georganiseerd en aangestuurd.

Testsoorten kunnen worden ingedeeld in zogenaamde **White-box** en **Black-box** testen. Hierbij zijn de White-box testen gebaseerd op de **technische en functionele vereisten** en de Black-box testen op functionele en **niet-functionele** vereisten. Daarnaast worden White-box testen uitgevoerd in een **ontwikkelomgeving** en Black-box testen in een "als het ware **productieomgeving**".

De testsoorten kunnen ook worden ingedeeld in:

- **Ontwikkeltesten**
  Deze testen worden uitgevoerd door de ontwikkelaars. En hebben als doel aan te tonen dat **de code voldoet aan de technische specificaties**. Deze testsoort kan nog verder worden **onderverdeeld** in:
  - **Unit test** (UT): aantonen dat een unit aan de in de technische specificaties gestelde eisen voldoet.
  - **Unitintegratietest** (UIT): aantonen dat een logische groep units aan de in de technische specificaties gestelde eisen voldoet.
- **Systeemtesten**
  Een door het ontwikkelteam uitgevoerde test, die moet aantonen dat het ontwikkelde systeem of delen daarvan aan de in de functionele- en technische specificaties gestelde eisen voldoen. Deze test vindt plaats in een (goed beheersbare) laboratoriumomgeving.
- **Acceptatietesten**
  Door de toekomstige gebruiker(s) en beheerder(s) uitgevoerde test, die moet vaststellen dat het ontwikkelde systeem aan de functionele en niet-functionele eisen voldoet. Deze test wordt uitgevoerd in een zoveel mogelijk als-ware-het-productie omgeving. De acceptatietest kan nog worden onderverdeeld in:
  - **Functionele acceptatietest** (FAT): vaststellen dat het ontwikkelde systeem aan de functionele eisen voldoet.

- **Gebruikersacceptatietest** (GAT): vaststellen dat het ontwikkelde systeem aan de wensen/eisen van de gebruiker voldoet.
- **Productieacceptatietest** (PAT): vaststellen dat het ontwikkelde systeem aan de de gestelde beheer eisen voldoet

De teststrategie geeft een totaaloverzicht van wat er getest wordt binnen welke testsoort en met welke diepgang. Hiermee kan voorkomen worden dat onderdelen in meerdere testsoorten met dezelfde diepgang en op basis van dezelfde requirements worden getest. Of dat bepaalde aspecten niet worden getest.

Opmerking

Staar je niet blind op de verschillende benamingen voor testsoorten. Het gaat uiteindelijk om wat er getest wordt binnen een testsoort en door wie. En dat kan per project verschillend zijn.
Binnen Agile/Scrum bestaan er geen testsoorten meer. In een scrum team werken alle belanghebbenden samen om het werk gedaan te krijgen. In een dergelijke omgeving is het niet relevant om over testsoorten als systeemtest en acceptatietest te spreken. Er bestaan bij Agile/Scrum ook geen aparte test teams met testmanagers en eigen budgetten.

TestData

Goede testdata zijn een voorwaarde voor een effectieve test.

Wanneer tijdens de planningsfase te weinig aandacht aan testdata is besteedt, blijkt tijdens het voorbereiden en uitvoeren van de verschillende tests dat de behoefte aan data en de wijze van beheer per testsoort sterk kan verschillen en de algemene beschrijving in een enkele paragraaf in het testplan niet voldoende is. In een systeemtest kan de dataset bijvoorbeeld worden benaderd per systeem of deelfunctionaliteit, terwijl een integratietest vraagt om data-integriteit over meerdere systemen. Het ontbreken van een goede planning en aanpak in testplan of -strategie leidt tot inconsistenties en fouten in de data waarvan de correctie grote gevolgen hebben voor inspanning en doorlooptijd.

Om een brug te kunnen slaan tussen de theorie en de praktijk van testsoorten en testdata, is een model ontwikkeld dat helpt om grip te krijgen op de benodigde testdata per testsoort en zo de voorbereiding en uitvoering van een testtraject beter te beheersen. Het model bestaat uit een beschrijving van testdata langs twee assen:

datasoorten en data-levensfases:

	Systeemconfiguratiedata	Masterdata	Transactionele data
Identificeren			
Genereren			
Beheren			

In ieder informatiesysteem en daarmee in ieder testtraject kan al de benodigde data worden onderverdeeld in drie datasoorten: systeemconfiguratiedata, masterdata en transactionele data.

Systeemconfiguratiedata *bestuurt de werking van het systeem.* Beslissingen en logica binnen het systeem tot het al dan niet actief zijn van hele modules worden bepaald door de configuratie. Om deze data te onderhouden, is over het algemeen een goede technische kennis van het systeem vereist en dit wordt dan ook meestal gedaan door ontwikkelaars of beheerders, niet door gebruikers. Voorbeelden van configuratiedata zijn het *rekeningschema van een boekhoudsysteem* of de *beslissingsregels in een rules engine*.

Masterdata is de referentie voor alle andere gegevens in het systeem en geeft hier betekenis aan. Deze data heeft vaak een uniforme betekenis voor de hele organisatie, heeft een lage wijzigingsfrequentie en een lange geldigheidsduur. Voorbeelden van masterdata zijn: *klantgegevens, leveranciergegevens of productgegevens*

Ten slotte is er **transactionele data**. Dit zijn gegevens die worden gemaakt of getransformeerd tijdens het gebruik van een systeem. Deze data heeft een hoge wijzigings- en aanmaakfrequentie, vaak een korte geldigheid en komt voor in hoge volumes. Deze data wordt vaak gebruikt in managementrapportages omdat het een reflectie is van wat er in het systeem (en dus de organisatie) is gebeurd. Een voorbeeld van transactionele data is: *orderdata, factuurdata, betalingsgegevens of boekhoudgegevens.*

Levensfases

Ongeacht de datasoort, dan wel testsoort, doorloopt alle testdata tijdens het voorbereiden en uitvoeren van een test impliciet of expliciet drie belangrijke fases. Deze drie levensfases omvatten het identificeren, genereren en beheren van de gegevens.

De eerste fase in de levenscyclus van data betreft het identificeren. Oftewel, het bepalen van de belangrijkste eigenschappen van de data. Voorbeelden van eigenschappen zijn: consistentie tussen systemen, benodigd volume, wel of niet geanonimiseerd.

Wanneer de testdata geïdentificeerd is, kan deze worden gegenereerd. Dit houdt in dat de benodigde data wordt gemaakt of gekopieerd uit een bron. Voor het genereren van data zijn er vele mogelijkheden. Het kan handmatig worden gemaakt op basis van testcondities, gekopieerd uit een productie- of testsysteem, gegenereerd door een tool of een combinatie hiervan.

Als de vragen rond identificatie en generatie beantwoord zijn, rest de laatste fase van beheer. Het is belangrijk om vast te stellen hoe gegevens beheerd zullen worden tijdens de voorbereiding en de uitvoering van tests. Hoe wordt data bijvoorbeeld tijdens de testvoorbereiding opgeslagen: in een testmanagementtool, spreadsheet of speciale testdatarepository? Hoe worden de gegevens ingevoerd: handmatig of geautomatiseerd? Hoe wordt de testomgeving ververst en hoe wordt de data onderhouden bij wijziging van een testgeval?

Testdata beheersmodel

Door de beschreven datasoorten en beheersfases tegen elkaar uit te zetten, ontstaat een model waarmee de behoefte aan testdata en de wijze van beheer voor iedere testsoort kan worden onderzocht en vastgelegd (zie schema).

Door toepassing van dit model kan bij het plannen van een testtraject al inzichtelijk worden gemaakt welke specifieke databehoeften er zijn binnen het traject. Hierdoor kunnen alle activiteiten rondom testdata, in zowel de voorbereiding als de uitvoering van het testtraject, in meer detail worden gepland en ontstaan minder verrassingen tijdens het project.

Testdatamodel ingevuld

In de volgende tabel is het model ingevuld voor typisch gebruik in de testsoorten Systeemtest, Integratietest, Acceptatietetst en Non-functionele test. Uit de ingevulde matrix is nu duidelijk af te leiden welke kennis en soort gegevens vereist is voor elke stap in het testtraject. Dit heeft tot gevolg dat behoeften aan bepaalde medewerkers, kennis en tools al in een vroeg stadium kunnen worden geïdentificeerd en een optimale planning van taken en capaciteit kan worden gemaakt.

		Systeemconfiguratiedata	Masterdata	Transactionele data
Systeem test	Identificeren	• Gelijk over alle omgevingen en testsoorten.	• Volgt testspecificaties en –condities.	• Zoveel mogelijk variaties en combinaties. • Volgt testspecificaties en –condities.
	Genereren	• Wordt opgeleverd door functioneel team of ontwikkelteam.	• Handmatig door testers of • Selectieve kopie uit productioneel systeem.	• Handmatig door testers of • Selectieve kopie uit productioneel systeem of • Dmv testautomatisering.
	Beheren	• Onder versiebeheer, net als applicatiecode.	• Vastgelegd in test scripts of • In een Test Data Repository	• Vastgelegd in test scripts of • In een Test Data Repository met laad-/beheersfunctionaliteit.
Integratie test	Identificeren	• Gelijk over alle omgevingen en testsoorten.	• Uniform over systemen heen.	• Minder variatie. • Nadruk op datastroom over systemen heen.
	Genereren	• Wordt opgeleverd door functioneel team of ontwikkelteam.	• Handmatig door testers of • Selectieve kopie uit productioneel systeem.	Handmatig door testers of (Selectieve) kopie uit productioneel systeem.
	Beheren	• Onder versiebeheer, net als applicatiecode.	• Vastgelegd in test scripts of • In een Test Data Repository. • Beheer integriteit over systemen heen is essentieel.	• Vastgelegd in test scripts of • In een Test Data Repository met laad-/beheersfunctionaliteit.
Acceptatie test	Identificeren	• Gelijk over alle omgevingen en testsoorten.	• Gelijk aan productioneel systeem. • Vaak gedepersonaliseerd.	• Gelijk aan productioneel systeem.
	Genereren	• Wordt opgeleverd door functioneel team of ontwikkelteam.	Laden vanuit productioneel systeem.	Laden vanuit productioneel systeem. Vastgelegd in testscripts.
	Beheren	• Onder versiebeheer, net als applicatiecode. • Migratie volgt OTAP model.	• Regelmatige verversing vanuit productioneel systeem.	• Regelmatige verversing vanuit productioneel systeem.
Non-Functionele Test	Identificeren	• Gelijk over alle omgevingen en testsoorten.	• Gelijk aan productioneel systeem. • Vaak gedepersonaliseerd. • Grote volumes noodzakelijk.	• Gebaseerd op productioneel systeem. • Grote volumes noodzakelijk.
	Genereren	• Wordt opgeleverd door functioneel team of ontwikkelteam.	• Vanuit productioneel systeem.	• Dmv testautomatisering.
	Beheren	• Onder versiebeheer, net als applicatiecode.	• In een Test Data Repository met laad-/beheersfunctionaliteit.	• In een Test Data Repository met laad-/beheersfunctionaliteit.

Bron: https://www.computable.nl/artikel/opinie/development/4082946/1509029/goede-testdata-is-geen-gegeven.html

Referenties

https://www.youtube.com/watch?v=I27UHgdNHyc