PROGRAMMEERTECHNIEKEN EN TESTEN–TESTLANDSCHAP

MATTHIAS DRUWÉ
SAM VAN BUGGENHOUT

Go to **www.menti.com** and use the code **24 26 99**

Waar denk je aan bij "software testing"?

Mentimeter

Slide is not active | Activate

0

Wat is testen?

- Een verzameling **activiteiten** die uitgevoerd wordt om een of meer **kenmerken** van een product, proces of dienst vast te stellen volgens een gespecificeerde **procedure**. [ISO/ IEC, 1991]
- Proces van het aantonen van **afwijkingen** tussen de **werkelijke** werking en de **verwachte** werking van een systeem
- Activiteit waarbij de **kwaliteit** van het gehele systeem of product wordt gecontroleerd
- Proces waarmee de **correcte werking** van een systeem of product wordt aangetoond
- Nagaan of software voldoet aan de vooropgestelde **vereisten** (“requirements”) voor het product

Wat is testen?

- Samengevat:
  - Voldoet software aan vereisten en ontwerp?
  - Werkt de software zoals verwacht?
  - Bevat de software ernstige “bugs”?
  - Voldoet de software aan het beoogde gebruik volgens de verwachtingen van de gebruiker?

Wat is testen?

- Twee belangrijke concepten: **validatie** en **verificatie**
  - **Validatie:** are we doing the right thing?
  - **Verificatie**: are we doing the things right?

- **Validatie:** nagaan of wát ontwikkeld werd, is wat de gebruiker wilde
- **Verificatie:** nagaan (of testen) of een artefact (inclusief de ontwikkelde software), conform (en consistent) is met de vooropgestelde specificaties

Wat is testen?

- Testen is meer dan enkel software uitvoeren en resultaat controleren!
  - Ook activiteiten zoals testplanning, analyseren, ontwerpen en implementeren van tests, rapporteren van testresultaten en de kwaliteit van testobjecten evalueren
- Sommige vormen van testing vergen dat software uitgevoerd wordt (= **dynamic testing**). Andere activiteiten (bv.: review van requirements en andere werkproducten) vereisen dit niet (= **static testing**)

Waarom testen?

Waarom testen?

- In 1985 zorgde een bug in de Therac-25 ervoor dat drie patiënten een dodelijke dosis bestralingen kregen. Drie andere patiënten waren in kritieke toestand.
- In april 1999 veroorzaakte een software bug het mislukken van de lancering van een militaire satelliet. Kostenplaatje: 1,2 miljard dollar.
- In 2015 moest Starbucks +- 60% van zijn vestigingen in de VS en Canada sluiten omwille van een software bug in hun POS-systeem

Waarom testen?

- Testen bespaart tijd en geld
  - Hoe sneller een fout kan hersteld worden, hoe goedkoper
  - Problemen oplossen voor schade aangericht wordt
- Veiligheid
  - Beveiligen van persoonlijke gegevens
  - Veiligheid van het systeem (bv. software besturing vliegtuig)
- Klantentevredenheid
  - Vertrouwen in het product
  - User Experience

Waarom testen?

- Kwaliteit (ISO 9126-standaard):
  - **Effectiviteit:** doet software wat het moet doen?
  - **Betrouwbaarheid:** vermogen om prestatieniveau te handhaven (bv.: veel gebruikers), “up-time”
  - **Gebruiksvriendelijkheid:** inspanning die gebruiker moet leveren om met software te werken
  - **Flexibiliteit:** mogelijkheid om de software te wijzigen (bv.: configuratie, …)
  - **Onderhoudbaarheid:** benodigde inspanning om aanpassingen aan te brengen
  - **Beheerbaarheid:** hoe gemakkelijk kan systeem operationeel gebracht en gehouden worden?
  - **Beveiliging:** persoonlijke gegevens, brand/diefstal/verstoringen/…
  - **Efficiëntie:** verhouding resultaat t.o.v. verbruikte middelen

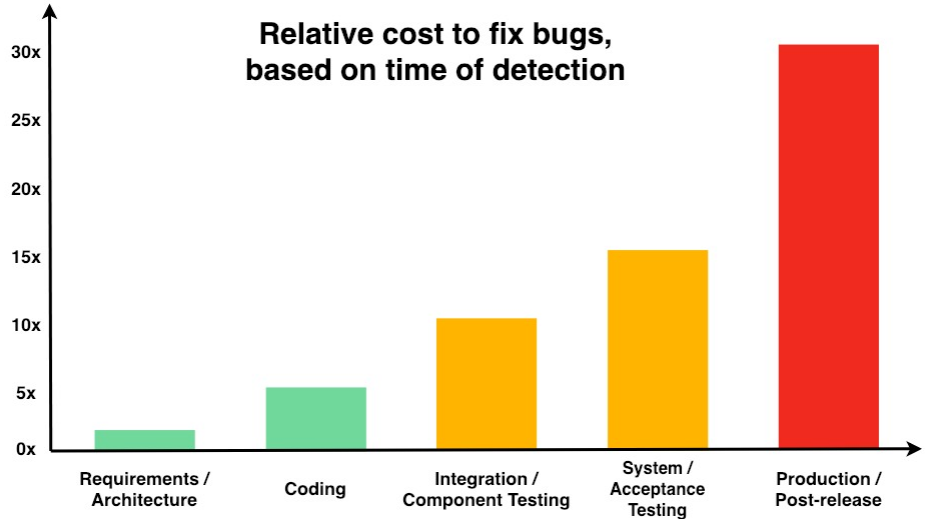
Testprincipes

1. Met testing kan de **aanwezigheid** van defecten aangetoond worden, niet de afwezigheid ervan
    - Testen verkleint de kans op de aanwezigheid van defecten
    - Het is niet omdat er geen defecten gevonden worden, dat de software volledig vrij is van defecten!

2. **Exhaustief** testen is onmogelijk
    - **ALLES** testen (alle invoer-combinaties en pre-condities) is onmogelijk
    - Focus d.m.v. risicoanalyse, test-technieken en prioriteiten

Testprincipes

3. **Vroeg testen** bespaart tijd en geld



Bron: https://deepsource.io/blog/6-benefits-continuous-quality/

Testprincipes

4. Defecten **clusteren** vaak samen
   - Een klein aantal modules (onderdelen) bevat vaak de meeste defecten die ontdekt worden tijdens het testen
5. Kijk uit voor de **pesticide-paradox**
   - Dezelfde tests telkens opnieuw uitvoeren, zal uiteindelijk geen nieuwe defecten aan het licht brengen
   - Door telkens oude tests uit te voeren, kunnen nieuwe bugs onopgemerkt blijven

     → update regelmatig de bestaande testscenario's!

Testprincipes

6. Testen is **context-afhankelijk**
   - Afhankelijk van het type applicatie dat getest wordt, de omgeving waarin het gebruikt wordt, …
   - Afhankelijk van de organisatie die de testen uitvoert (ontwikkelingsproces)

7. **Absence-of-errors** fallacy
   - Zelfs indien software 99% bug-vrij is, maar niet voldoet aan de gebruikersvereisten, is de software onbruikbaar!

Het Testproces

- Er bestaat **geen universeel** testproces
- Maar: wel een aantal **activiteiten** die typisch doorlopen worden bij het testen
- Deze activiteiten samen vormen het “testproces”
- Activiteiten die uitgevoerd worden, manier waarop ze uitgevoerd worden etc. is afhankelijk van **verschillende factoren**, bijvoorbeeld:
  - Product- en projectrisico’s
  - Business Domain
  - Budget en resources
  - Planning
  - Complexiteit
  - Contractbepalingen
  - …

Het Testproces

- Een testproces bestaat meestal uit de volgende (groepen van) taken:
  - Test Planning
  - Test Monitoring & Control
  - Test Analysis
  - Test Design
  - Test Implementation
  - Test Execution
  - Test Completion

Het Testproces

- Test Planning
  - Testdoelen definiëren
  - Testtechnieken specificeren
  - Planning opstellen (deadline)
- Test Monitoring & Control
  - Werkelijke vooruitgang aftoetsen met de vooropgestelde planning
  - Op basis van “exit criteria” (= wanneer is een test “afgerond”?)
  - Schatting van kwaliteit op basis van testresultaten → meer testen nodig?

Het Testproces

- Test Analysis
  - Testbare features identificeren (**wat** moet getest worden?)
  - Analyse van de testbasis:
    - Requirements (functionele en niet-functionele)
    - Informatie over het design (UML, ERD, flow-charts, …)
    - Implementatie van de componenten (code, query's, …)
    - Risicoanalyses, …
  - Evalueren van de testbasis: identificeren van defecten, zoals:
    - Dubbelzinnigheden
    - Weglatingen
    - Inconsistenties
    - Onnauwkeurigheden
    - Tegenstrijdigheden (contradicties)
    - …

Het Testproces

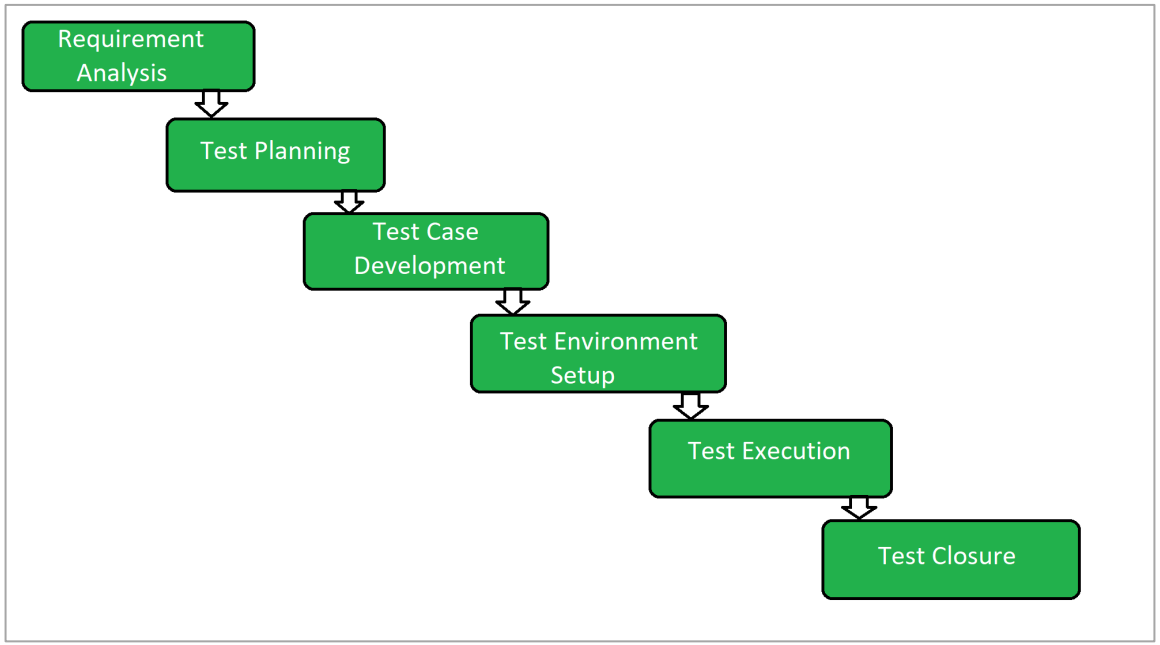
- Test Design
  - Test Analyse = **WAT** testen?, Test Design = **HOE** testen?
  - Ontwerpen en prioriteren van test cases
  - Identificeren van de data die nodig is om test cases uit te voeren
  - Nodige infrastructuur en test-omgeving identificeren
- Test Implementation
  - Is alles klaar om de testen uit te voeren?
  - Ontwikkelen en prioriteren van test procedures (geautomatiseerde scripts)
  - Test suites creëren op basis van test procedures (en eventuele scripts)
  - Test omgeving opzetten (inclusief virtualisatie, simulators, …)
  - Testdata voorbereiden en controleren of deze correct ingeladen is

Het Testproces

- Test Execution
  - Test suites die in de vorige stap ontwikkeld werden uitvoeren
  - Testen uitvoeren (zowel manueel als via tools)
  - Werkelijke resultaten vergelijken met verwachte resultaten
  - Anomalieën analyseren en de oorzaak van een probleem proberen te achterhalen
  - Defecten rapporteren (gebaseerd op observaties)
  - Testresultaten loggen (“pass”, “fail”, “blocked”, …)
- Test completion
  - Verzamel gegevens van de voorgaande activiteiten
  - Zijn alle defect-reports gesloten? Change request voor defecten die nog niet opgelost zijn
  - Samenvatting van de testresultaten
  - Ervaringen gebruiken om testproces te verbeteren

Het Testproces

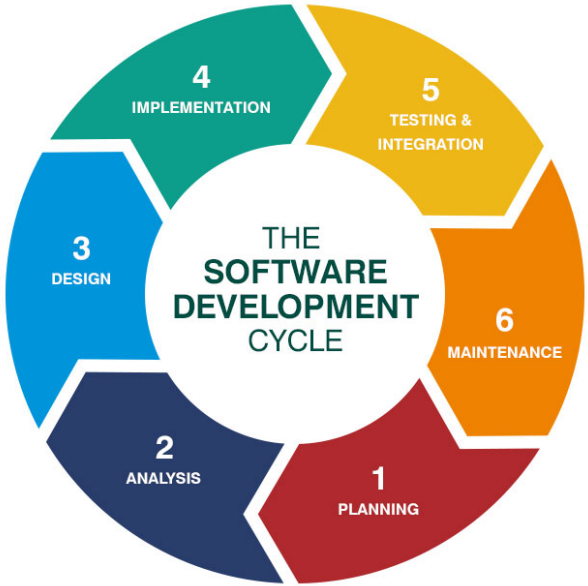
STLC (**S**oftware **T**esting **L**ife**C**ycle)



Bron: https://www.geeksforgeeks.org/software-testing-life-cycle-stlc/

Software Ontwikkeling en Testing

- Net zoals bij testing, bestaat er **geen universeel proces** voor de ontwikkeling van software
- De volgende activiteiten komen typisch aan bod bij software ontwikkeling:



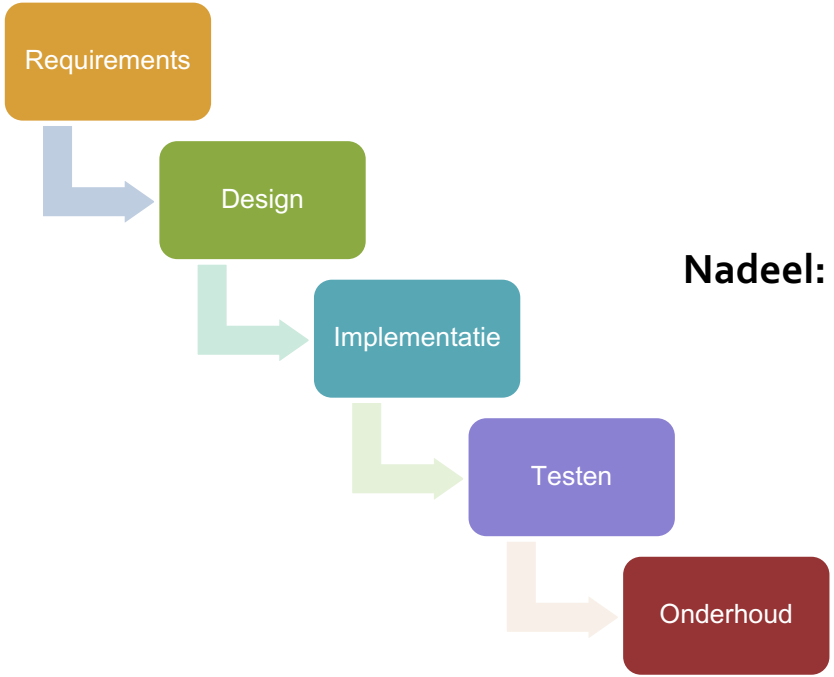
Bron: https://datarob.com/essentials-software-development-life-cycle/

Software Ontwikkeling en Testing

- Er bestaan verschillende **S**oftware **D**evelopment **L**ife**C**ycle modellen (SDLC)
- Elk model heeft **eigen visie** op testen
- SDLC's kunnen ingedeeld worden in twee **categorieën**:
  - Sequentiële modellen
  - Iteratieve en incrementele modellen

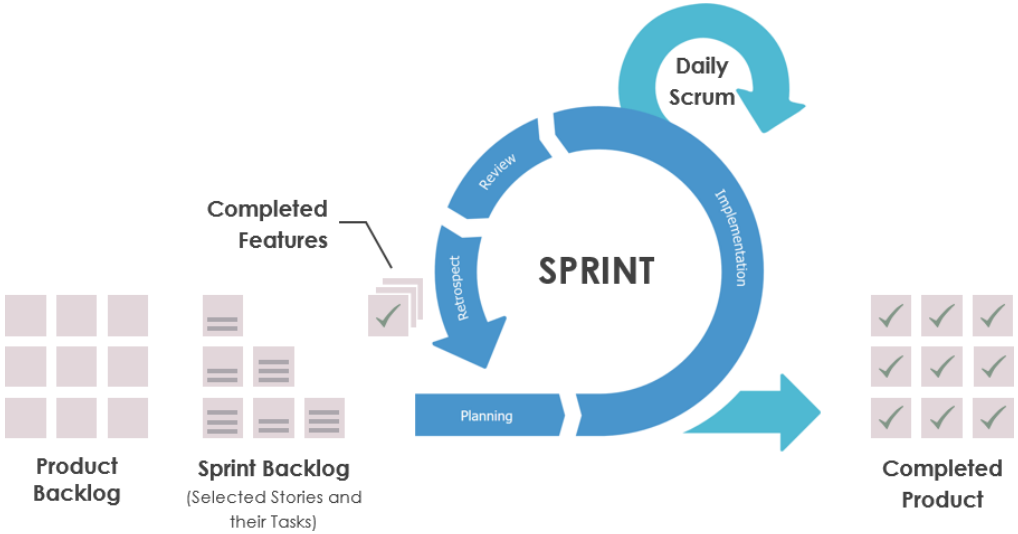
- **Sequentiële modellen:** volgende fase start pas wanneer de vorige fase volledig afgerond is (bv. Waterfall-methode)

- Voorbeeld **sequentieel model**: *Waterfall*



Nadeel: testen gebeurt zeer **laat** in het proces!

- **Incrementele** en **iteratieve** ontwikkelingsmodellen
  - **Incrementeel:** het bouwen van software gebeurt in "stukjes" (software "groeit")
  - **Iteratief:** ontwikkelingscyclus wordt meerdere malen doorlopen
- Voorbeeld: *Scrum*



Bron: https://www.visual-paradigm.com/scrum/what-is-sprint-in-scrum/

- In elk model zijn er een aantal **karakteristieken** van goede testen:
  - Voor elke **ontwikkelingsactiviteit** is er een overeenkomstige **testactiviteit**
  - Elk **test level** heeft doelen die specifiek zijn voor dat level
  - **Test Analyse** en **Test Design** voor een bepaald test level beginnen tijdens de overeenkomstige **ontwikkelingsactiviteit**
  - Testers nemen deel aan overleg om requirements en design te definiëren en te verfijnen. Wanneer eerste versies van requirements en design klaar zijn, “reviewen” ze deze.

Testlevels

- **Testlevels**: groepen van **testactiviteiten** die samen **gepland** en **beheerd** worden
- Voor elk level wordt de **test-lifecycle** (de verschillende activiteiten) doorlopen
- In elk testlevel wordt de software op een **ander niveau** getest (van individuele componenten tot volledige systemen)
- Veelgebruikte testlevels zijn:
  - Component testing (ook "Unit-testing" genoemd)
  - Integration testing
  - System testing
  - Acceptance testing

Testlevels

- Component testing (= unit testing)
  - Focus op componenten die **afzonderlijk** kunnen getest worden
  - Doelen:
    - Risico's verkleinen
    - Vertrouwen opbouwen in de kwaliteit van de component
    - Voorkomen dat eventuele defecten doorsijpelen naar "hoger" niveau
  - Wat wordt getest?
    - **Code** en datastructuren
    - **Classes** en **methodes**
    - **Databasemodules**
  - Typische defecten:
    - Fouten in de **logica**
    - Verkeerde **functionaliteit**

Testlevels

- Integration testing
  - Focus op **interactie** tussen verschillende componenten (of systemen)
  - Doelen:
    - Risico's verkleinen
    - Vertrouwen opbouwen in de kwaliteit van de component
    - Voorkomen dat eventuele defecten doorsijpelen naar "hoger" niveau
  - Kan toegepast worden op twee "niveaus":
    - **Component integration testing:** focus op interactie en "interface" tussen geïntegreerde componenten
    - **System integration testing:** focus op interactie en "interface" tussen systemen (ook extern, bv.: API's)

Testlevels

- Integration testing (vervolg)
  - Wat wordt getest?
    - Infrastructuur
    - Interfaces
    - API's
    - Databases
  - Typische defecten:
    - Verkeerde/ontbrekende gegevens
    - Verkeerde volgorde/timing van interface calls
    - Interface mismatch
    - Problemen bij communicatie tussen componenten

Testlevels

- System testing
  - Focus op het gedrag (en mogelijkheden) van een **volledig systeem** of **product**
  - Doelen:
    - Nagaan of systeem voldoet aan functionele en non-functionele vereisten
    - Valideren of systeem volledig is en werkt zoals verwacht
  - Wat wordt getest?
    - Applicaties
    - Software/hardware –systemen
    - Operating Systems
    - Systeemconfiguratie
  - Typische defecten:
    - Foutief/onverwacht gedrag van het systeem
    - Systeem werkt niet in de omgeving
    - Systeem werkt niet zoals beschreven in handleiding

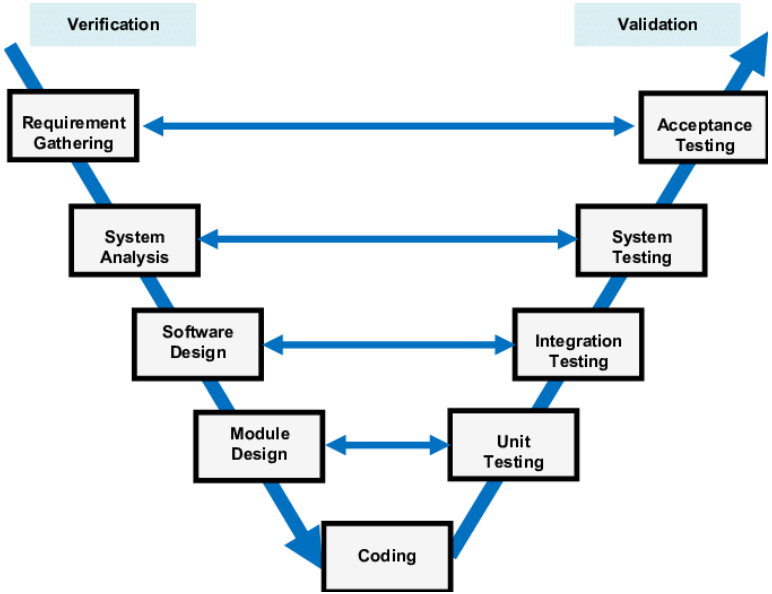
Het Testproces

- Acceptance testing
  - Focus typisch op gedrag van **volledige systeem** of **product**
  - Nagaan of systeem **"klaar"** is en kan gebruikt worden door **eindgebruikers**
  - **Vier vormen van acceptatietesten:**
    - **User Acceptance Tests (UAT):** kunnen de gebruikers het systeem gebruiken en hun doelen ermee bereiken?
    - **Operational Acceptance Testing (OAT):** vertrouwen opbouwen dat systeem administrators het systeem operationeel (werkend) kunnen houden (bv.: performance testing, security testing, backup en restore, …)
    - **Contractueel:** voldoet software aan criteria die in contract vastgelegd werden?
    - **Alpha en Beta Testing:** alpha testing wordt door development team uitgevoerd, beta testing door eindgebruikers

Het Testproces

- Voor elke **ontwikkelingsactiviteit** is er een overeenkomstige **testactiviteit**

  Het V-model:



Bron: https://www.researchgate.net/figure/V-Model-life-cycle-for-the-automotive-software-testing_fig1_314665883

Soorten testen

- Een **test type** is een groep van testactiviteiten die gericht zijn op het testen van **specifieke eigenschappen** van het systeem
- In deze cursus maken we een onderscheid tussen vier test types:
  - Functionele testen
  - Niet-functionele testen
  - White-box Testing
  - Change-related Testing

Soorten testen

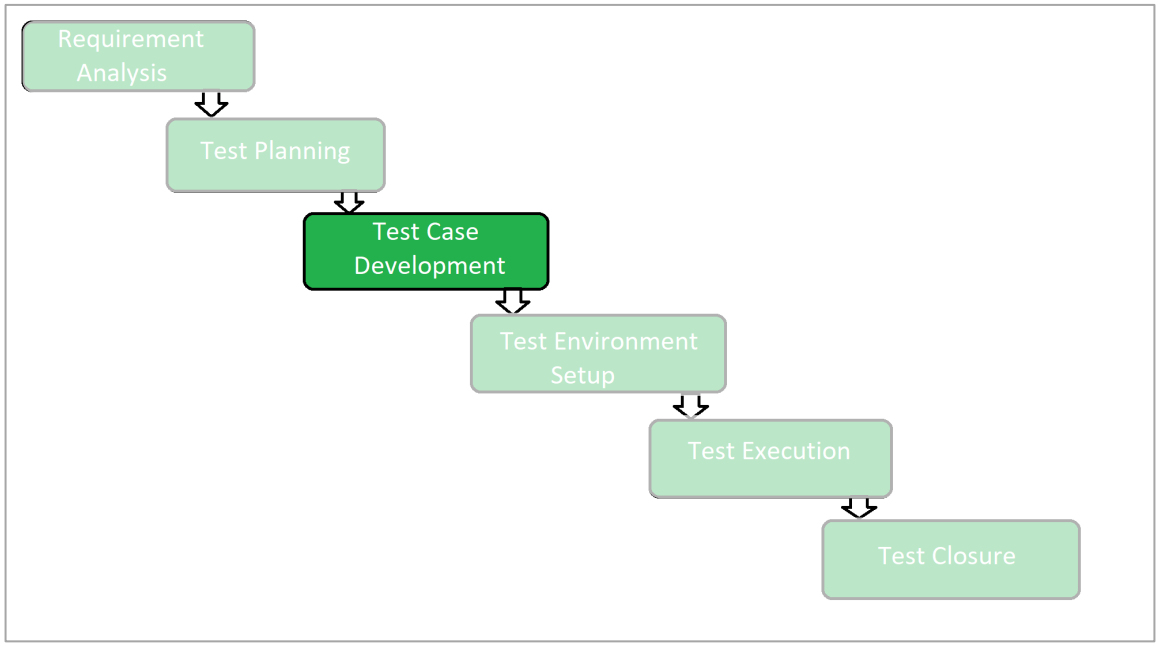
- Functionele testen:
  - Evalueren de functionaliteiten van een systeem (= "wat" het systeem moet doen)
  - Basis: functionele vereisten ("business requirements")
  - Moet worden uitgevoerd op alle "testlevels"
  - Vergt vaak kennis van business domein waarin software werkt
- Niet-functionele testen:
  - Valideert de niet-functionele vereisten
  - Bv.: gebruiksvriendelijkheid, performance (efficiëntie), security, …
  - "Hoe goed" presteert het systeem?
  - Zo snel mogelijk in het ontwikkelingsproces
  - Op alle "testlevels"

Soorten testen

- White-box testing
  - Tests afleiden van de interne structuur (implementatie) van het systeem
  - Bv.: code, workflows, dataflows, …
  - Volledigheid kan nagegaan worden met bepaalde metrieken (code coverage, statement coverage, branch coverage, decision coverage, …)
  - **Black-box testing:** testen zonder kennis van interne structuur (bv.: geen toegang tot de code)
- Change-related testing:
  - Testen gerelateerd aan wijzigingen binnen het systeem
  - Twee vormen:
    - **Confirmation testing:** na een fix tests schrijven die bevestigen dat het probleem opgelost is
    - **Regression testing:** testen die nagaan of na een wijziging (bv.: een her-organisatie van de code), de **bestaande** code nog steeds werkt

Test Scenario's en Test Cases

Test Cases: situering



Test Scenario vs Test Case

- Een **Test Scenario** is elke functionaliteit die getest kan worden (use case)
- Een **high-level** beschrijving van **WAT** getest moet worden

- Voorbeeld (webshop):
  - **Scenario 1:** Verifieer de login-functionaliteit
  - **Scenario 2:** Verifieer de zoek-functie
  - **Scenario 3:** Verifieer de betaal-functie

Test Scenario vs Test Case

- Een **Test Case** is een verzameling van **voorwaarden** of **variabelen** waaronder een tester bepaalt of de software aan de vereisten (requirements) voldoet en naar behoren functioneert
- **Eén test**, die door de tester kan uitgevoerd worden
- Een test case **"gidst" testers** door de verschillende **stappen** die nodig zijn om de test uit te voeren
- Bevat **stap-voor-stap instructies** om te verifiëren of de software doet wat ervan verwacht wordt

Test Scenario vs Test Case

- Voorbeeld:
  - **Test scenario:** *Verifieer de login-functionaliteit*
    - **Test case 1:** verifieer het gedrag wanneer het e-mail adres én het wachtwoord geldig zijn
    - **Test case 2:** verifieer het gedrag wanneer zowel het e-mail adres als het wachtwoord ongeldig zijn
    - **Test case 3:** verifieer het gedrag wanneer het e-mail adres geldig is, maar het wachtwoord ongeldig is
    - **Test case 4:** verifieer het gedrag wanneer het wachtwoord geldig is, maar het e-mail adres ongeldig is
    - **Test case 5:** verifieer het gedrag van de "keep me signed in"-optie
    - …

Onderdelen van een Test Case

- Een Test Case bevat meestal de volgende onderdelen:
  - Een titel
    - Zorg voor een **duidelijke** titel
    - Geef elke test case eventueel een uniek ID
  - Een beschrijving
    - Maak duidelijk **wat** er getest wordt
  - Pre-condities
    - Voorwaarden waaraan moet voldaan zijn **alvorens** de test kan starten
    - Bv.: de gebruiker is ingelogd, correcte PIN-code werd ingevoerd, …
  - De verschillende stappen
  - Verwachte resultaat
    - **Post-condities:** waaraan moet voldaan zijn **ná** het uitvoeren van de test?
  - Werkelijke resultaat

Test Case: voorbeeld

- **Titel:** 001.Login Pagina – Succesvol inloggen op de webshop
- **Beschrijving:** een geregistreerde gebruiker moet zich kunnen inloggen op de webshop
- **Pre-condities:** de gebruiker moet reeds geregistreerd zijn met een email-adres en wachtwoord
- **Stappen:**
  1. Navigeer naar *www.brol.com*
  2. Vul in het veld "email" het email-adres in van de geregistreerde gebruiker
  3. Vul in het veld "wachtwoord" het wachtwoord in van de geregistreerde gebruiker
  4. Klik op de knop "Aanmelden"
- **Verwachte resultaat:** in de rechterbovenhoek wordt de naam van de ingelogde gebruiker weergegeven alsook het aantal artikelen die reeds in het winkelmandje van de gebruiker liggen

Test Cases: opmerkingen

- Soms kan het ook handig zijn om op voorhand de data die je nodig hebt voor de test te specificeren
- Groepeer gerelateerde testen (bv.: “link” de test cases die bij eenzelfde test scenario horen)
- Voorzie niet enkel testen voor “positieve” scenario’s (happy flow), maar ook voor “negatieve” scenario’s en onverwachte flows
  - Bv.: Voer een foutief email-adres/wachtwoord in, Gebruiker is wachtwoord vergeten, …

Van Requirements naar Test Cases

- Voorbereiding:
  1. Bij de start van het project, schrijft de product owner/klant/… een document met de vereisten (Requirements) en acceptatie criteria
  2. Testers reviewen deze documenten en beginnen met het opstellen van de test cases
  3. Bij onduidelijkheden worden de vereisten verder aangevuld

Van Requirements naar Test Cases

- Voorbeeld:
  - Regelmatige klanten moeten producten die ze reeds eerder besteld hebben eenvoudig opnieuw kunnen bestellen, zonder dat ze deze telkens opnieuw moeten opzoeken
    - **ACC 1:** de optie "Bestelgeschiedenis" is zichtbaar op de account-pagina
    - **ACC 2:** wanneer de klant op "Bestelgeschiedenis" klikt, krijgt hij/zij een overzicht met de producten die hij/zij in het verleden heeft aangekocht
    - **ACC 3:** de gebruiker kan eerder gekochte items aan zijn/haar winkelmandje toevoegen
  - Visualiseer de workflow (maak eventueel een flow-diagram)

Van Requirements naar Test Cases

- Creëer het “Happy Path”-scenario
  - Pad zonder fouten of alternatieve wegen
  - Voorbeeld:
    1. Koop producten via de zoek-functie
    2. Klik op “Bestelgeschiedenis” van de account-pagina
    3. Verifieer of de eerder bestelde producten getoond worden
    4. Voeg producten van de lijst met reeds bestelde producten toe aan het winkelmandje
    5. Vervolledig de aankoop van de eerder bestelde producten

Van Requirements naar Test Cases

- Standaard items:
  - Verifieer of alle "standaard objecten" (bv.: menu's, headers, footers, logo's, …) zichtbaar zijn op alle nieuwe schermen en pagina's

    → **worden vaak over het hoofd gezien!**

- Negatieve Test Cases
  - Welke problemen kunnen mogelijks optreden?
  - Voorbeeld:
    - Een product is niet op voorraad: wordt dit getoond op de pagina "Bestelgeschiedenis"?
    - De prijs van het product is gewijzigd t.o.v. de eerdere bestelling, wordt dit weergegeven op de pagina?
    - Welke foutboodschappen worden getoond indien een systeemfout optreedt? Schrijf test cases die verifiëren dat elke foutboodschap op het gepaste moment getoond wordt

Van Requirements naar Test Cases

- Grenzen:
  - Is er een minimum en/of maximum voor aantallen/grenzen?
  - Voorbeeld:
    - Hoeveel producten worden getoond in het overzicht van de bestelgeschiedenis? Bv.: alle producten kunnen getoond worden, maar wat indien een gebruiker meer dan 5000 producten besteld heeft?
    - Wat moet er getoond worden indien de gebruiker in het verleden nog geen bestellingen geplaatst heeft?

Bronnen

- https://www.guru99.com/software-testing-introduction-importance.html
- https://www.testdevlab.com/blog/2018/07/importance-of-software-testing/
- https://blog.testlodge.com/what-is-a-test-case-in-software-testing/