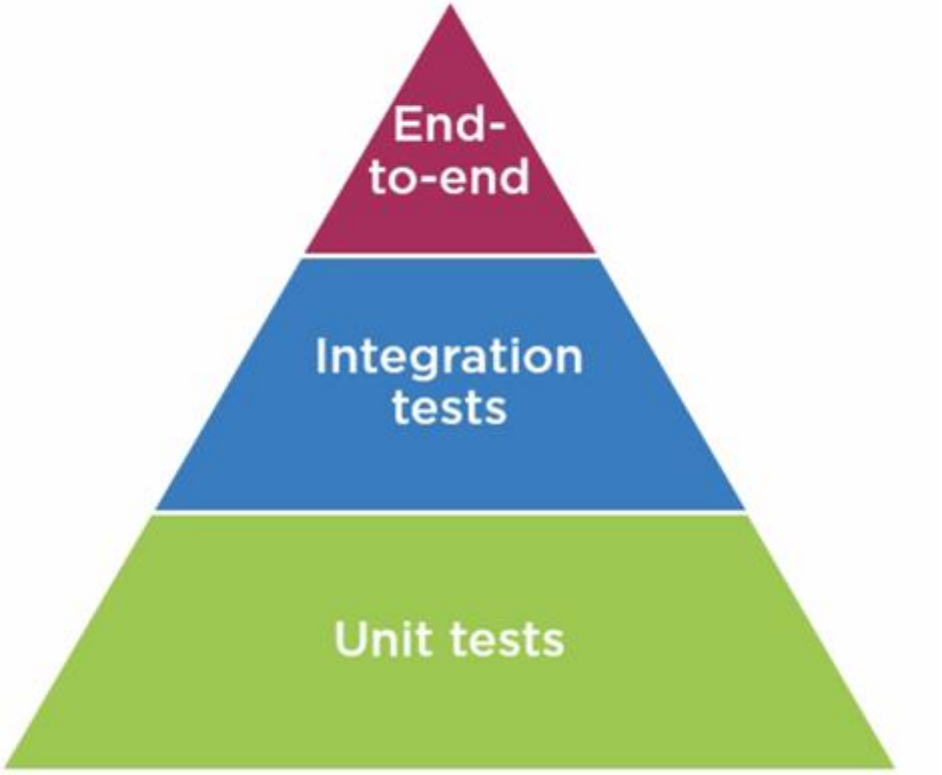
Testen van Software

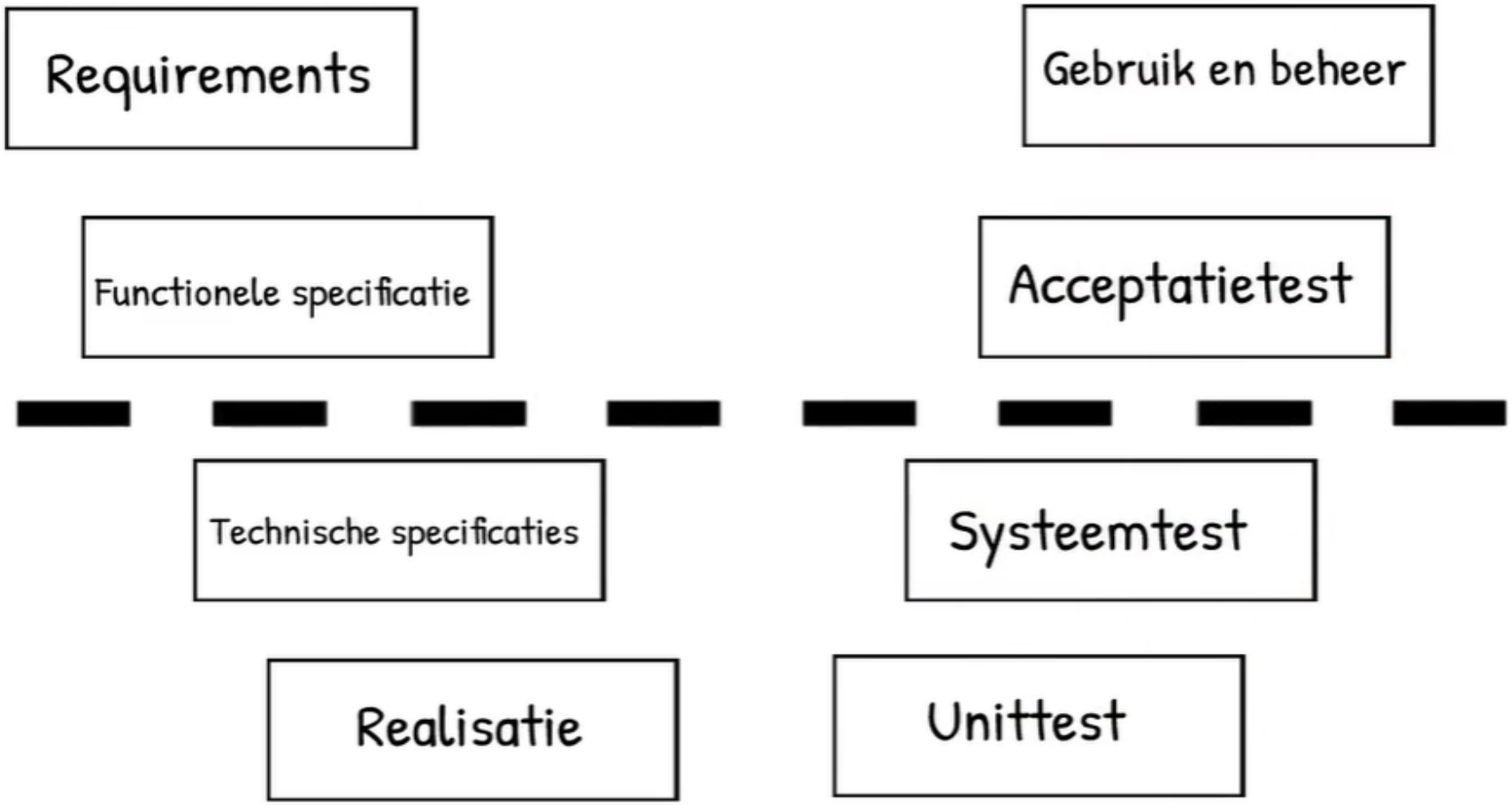
Inhoud Software Testen

- Software testen: wat moet er getest worden?
- Soorten testen (unit – integration – system – acceptance - requirements)
- Test plan
- Test cases
- Code-first versus Test-first
- TDD en BDD
- Unit Testen

Software testen – C# developers



Soorten testen



Categorieën van testen

2 categorieën van testen

- White-box testen:

focus op intern functioneren van de software **(technische en functionele vereisten)**

De implementatie is gekend door de tester

- Black-box testen.

focus op **gedrag** (**functionele** en **niet-functionele** vereisten).

Vanuit perspectief van externe of eindgebruiker

De implementatie is niet gekend door de tester

Soorten testen

- **Ontwikkeltesten**
  Deze testen worden uitgevoerd door de ontwikkelaars

  verder **onderverdeeld** in:

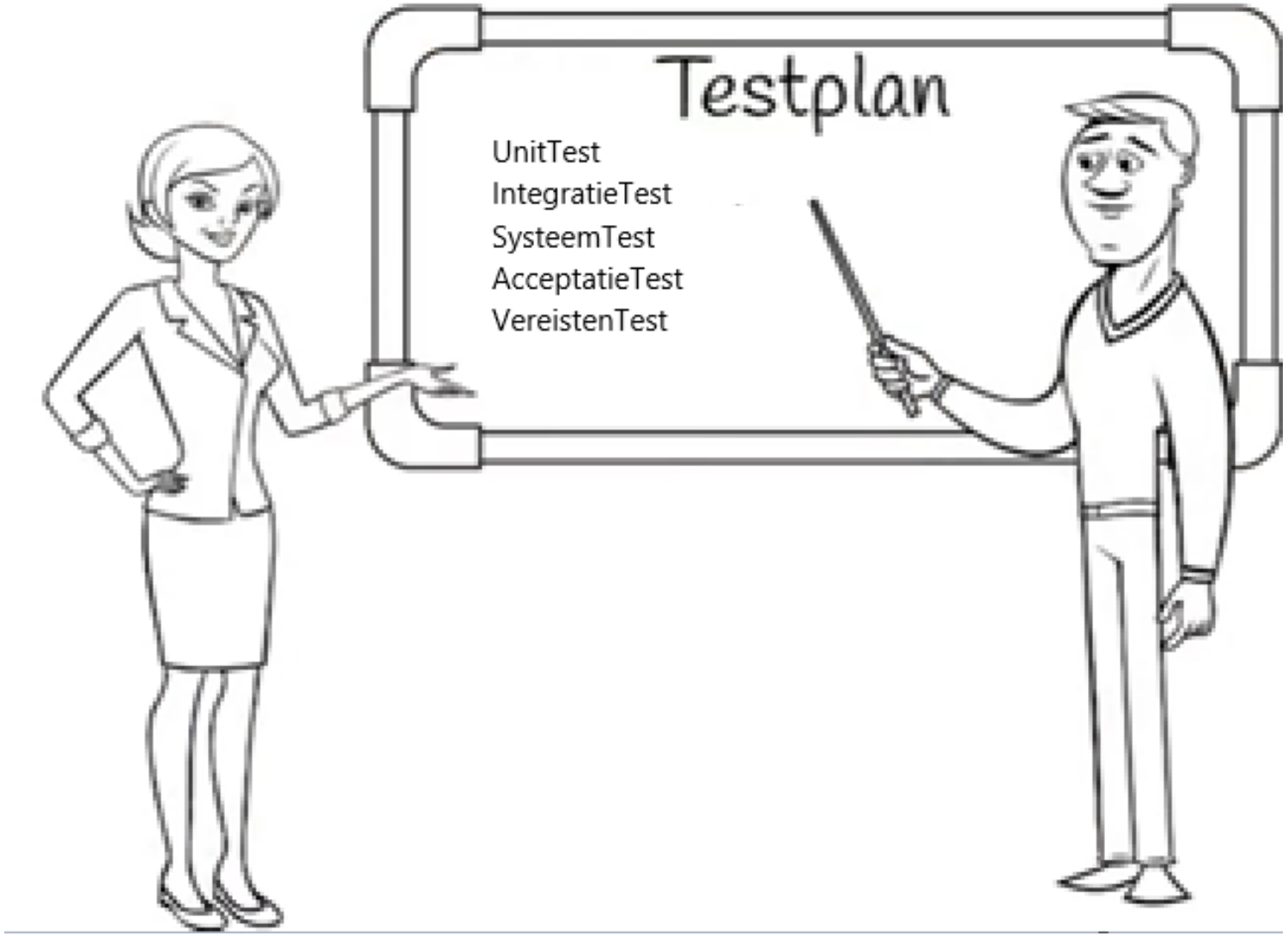
  - **Unit test** (UT)
  - **Unitintegratietest** (UIT)
- **Systeemtesten**
  testen of het **ontwikkelde systeem of delen daarvan** aan de in de **functionele- en technische specificaties gestelde eisen voldoen**.

  uitgevoerd in een (goed beheersbare) testomgeving.
- **Acceptatietesten**
  Door de toekomstige gebruiker(s) uitgevoerde test of het systeem aan **de functionele en niet-functionele eisen** voldoet. Uitgevoer in “als-ware-het-productie” omgeving.

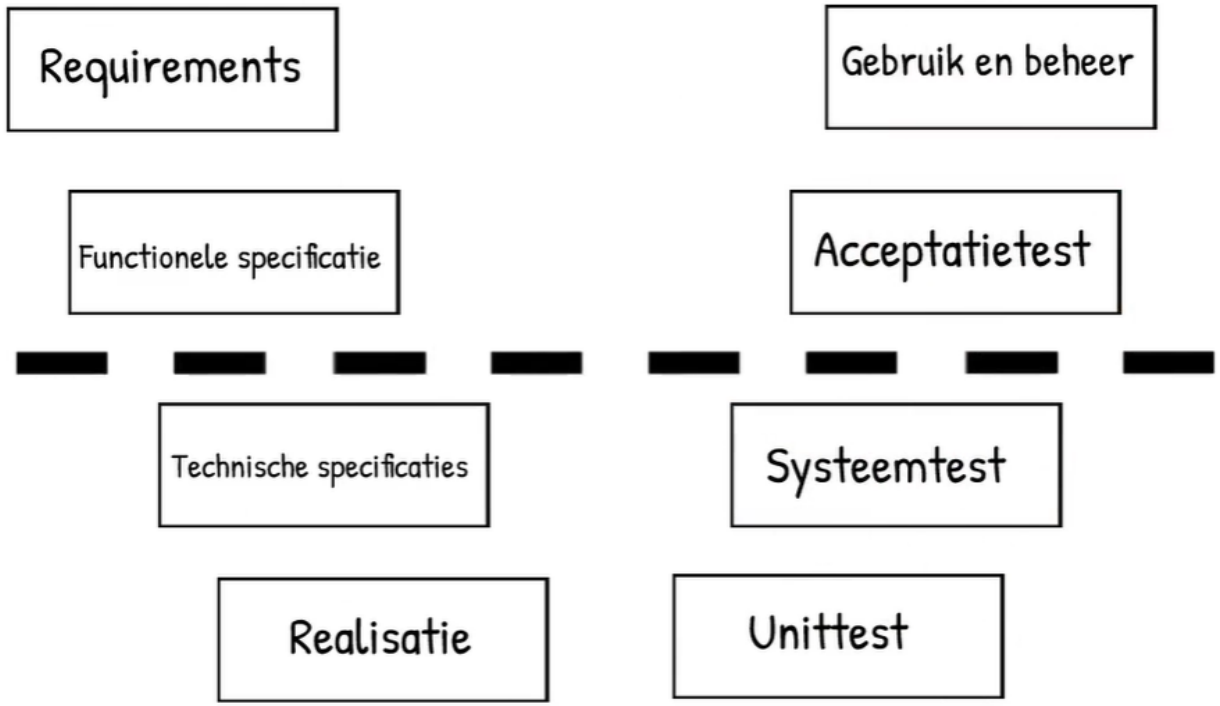
  is verder onderverdeeld in:

  - **Functionele acceptatietest** (FAT): testen of het systeem aan de **functionele eisen** voldoet.
  - **Gebruikersacceptatietest** (GAT): testen of het systeem aan de **wensen/eisen van de gebruiker** voldoet.
  - **Productieacceptatietest** (PAT): testen of het systeem aan de **productieeisen** voldoet

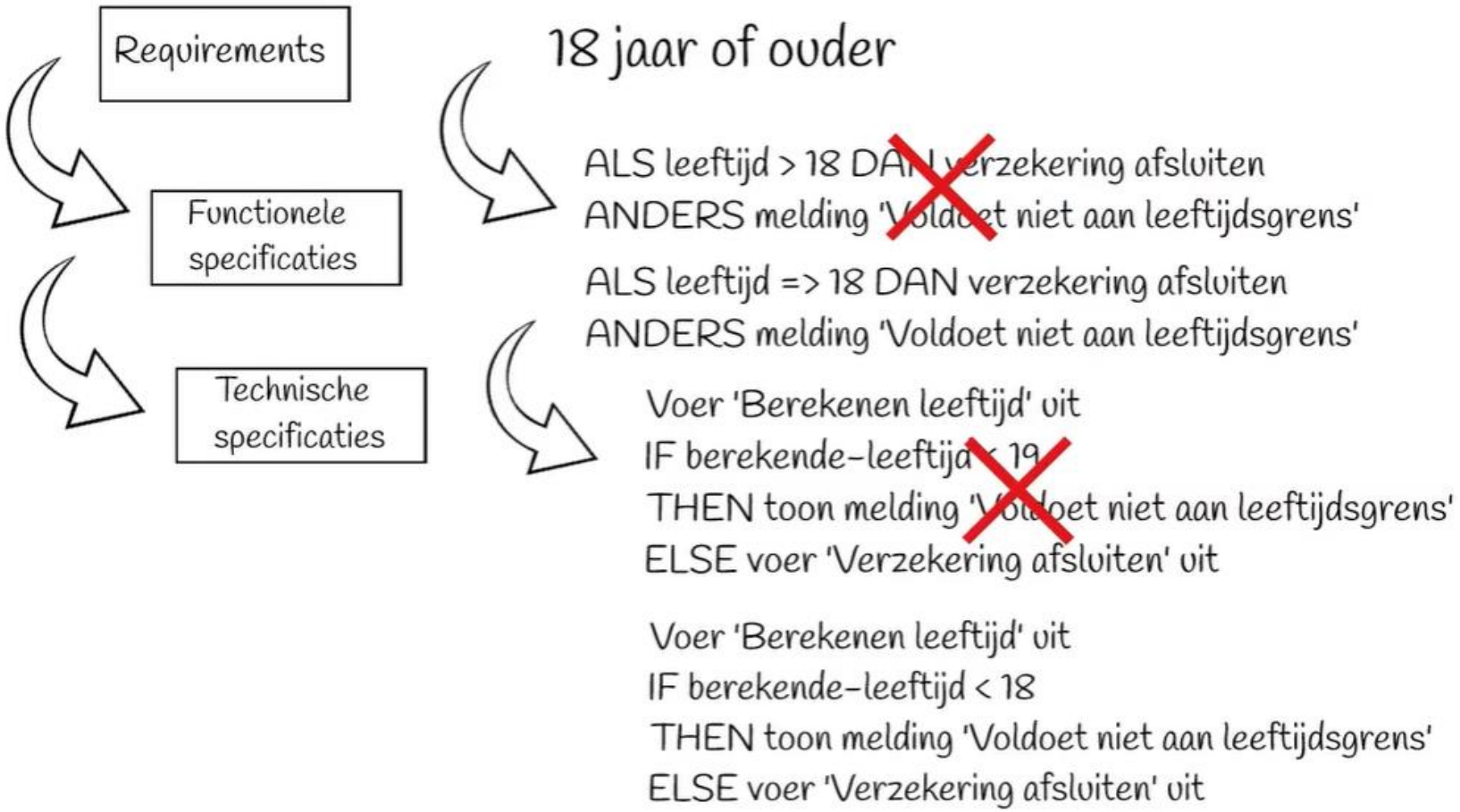
Software testen



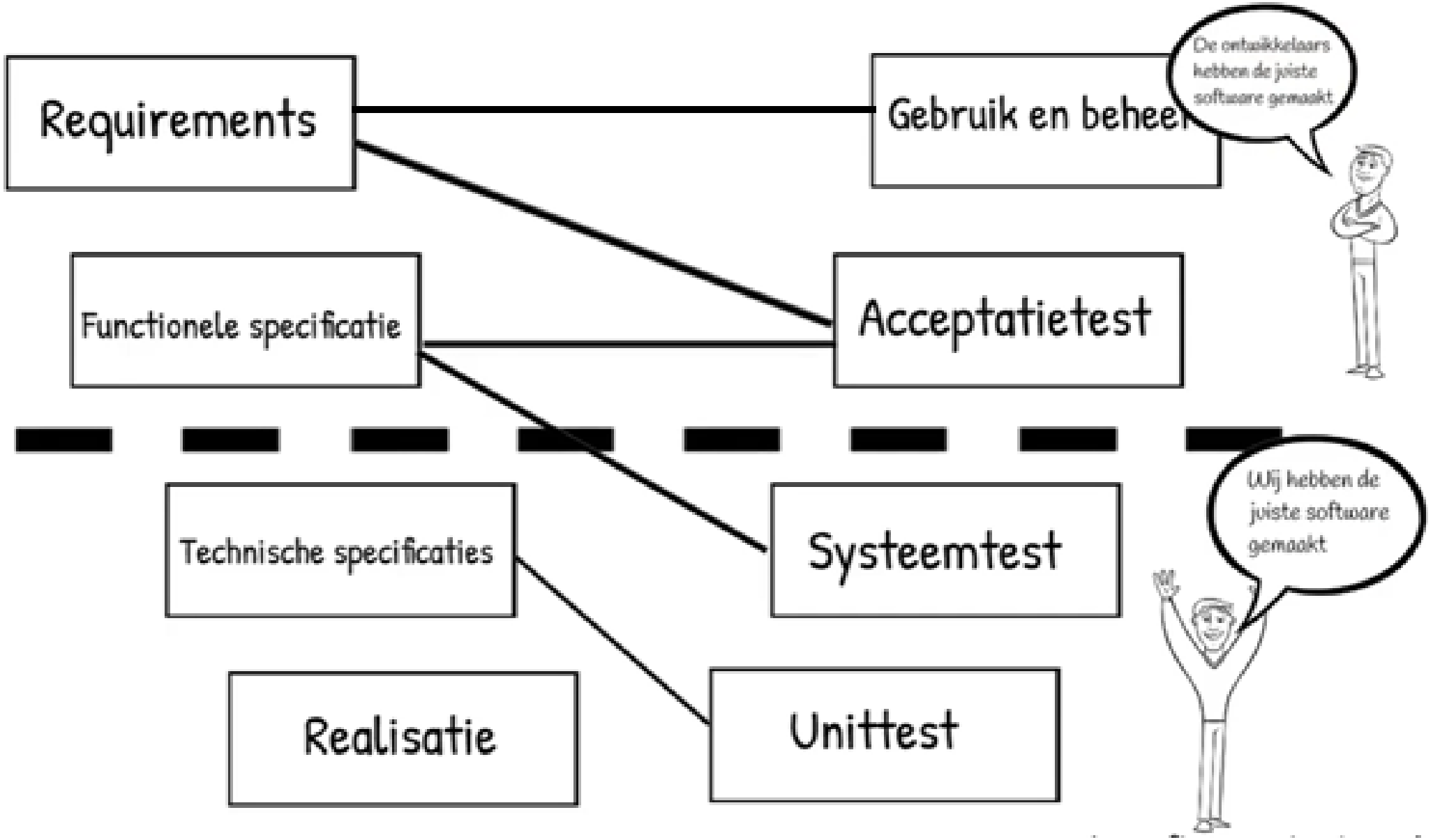
Software testen



Software testen – fouten bij ontwikkeling code



Software testen



Software testen – testplan

Voorbeeld van testplan

Voor het opstellen van een testplan zijn er sjablonen te vinden op internet:

https://docplayer.nl/11880213-Sjabloon-testplan-op-basis-van-sysqa-teststrategieaanpak-organisatie.html

Software testen – test cases

- **Positieve testcases (“Happy path testing”)**

  uitvoeren van scenario’s zoals deze normaal door een eindgebruiker zal worden gebruikt.

  Bij het uitvoeren van deze scenario’s worden alleen correcte en geldige gegevens gebruikt.

- **Negatieve testcases (“Error path testing“)**

  bedoeld om mogelijke tekortkomingen aan het licht te brengen die ernstige gevolgen hebben voor het gebruik van de applicatie

  Bv bepalen van de omstandigheden waaronder de applicatie kan crashen.

  Controleren of er voldoende validaties zijn in de software

Software testen – voorbeelden test cases

1. Stel we hebben een webapplicatie waar we blogs mee kunnen plaatsen. Voor de applicatie gelden onder andere de volgende vereisten(requirements):
    - Het password moet letters en cijfers bevatten.
    - Er mogen per gebruiker maximaal 5 blogs per dag worden geplaatst.
    - In een blog mogen alleen .jpg foto's worden gebruikt
    - Een blogpost mag niet langer zijn dan 300 worden.

Vraag: Geef een voorbeeld van positieve en een negatieve testcase

Software testen – vb positieve test cases

- Creëer een password met letters en cijfers
- Creëer 4 blogs op een dag
- Creëer 5 blogs op een dag
- Creëer 1 blog met een .jpg foto in de blog
- Creëer 1 blog met 299 woorden
- Creëer 1 blog met 300 woorden

Software testen – vb negatieve test cases

- Creëer een password met alleen letters
- Creëer een password met alleen cijfers
- Creëer 6 blog op een (1) dag
- Creëer een blog met daarin een .png, tif en andere niet .jpg bestand types
- Creëer een blog met daarin 301 woorden
- Creëer een blog met daarin meer dan 301 woorden

Code-first versus Test-first

- **Code-first**

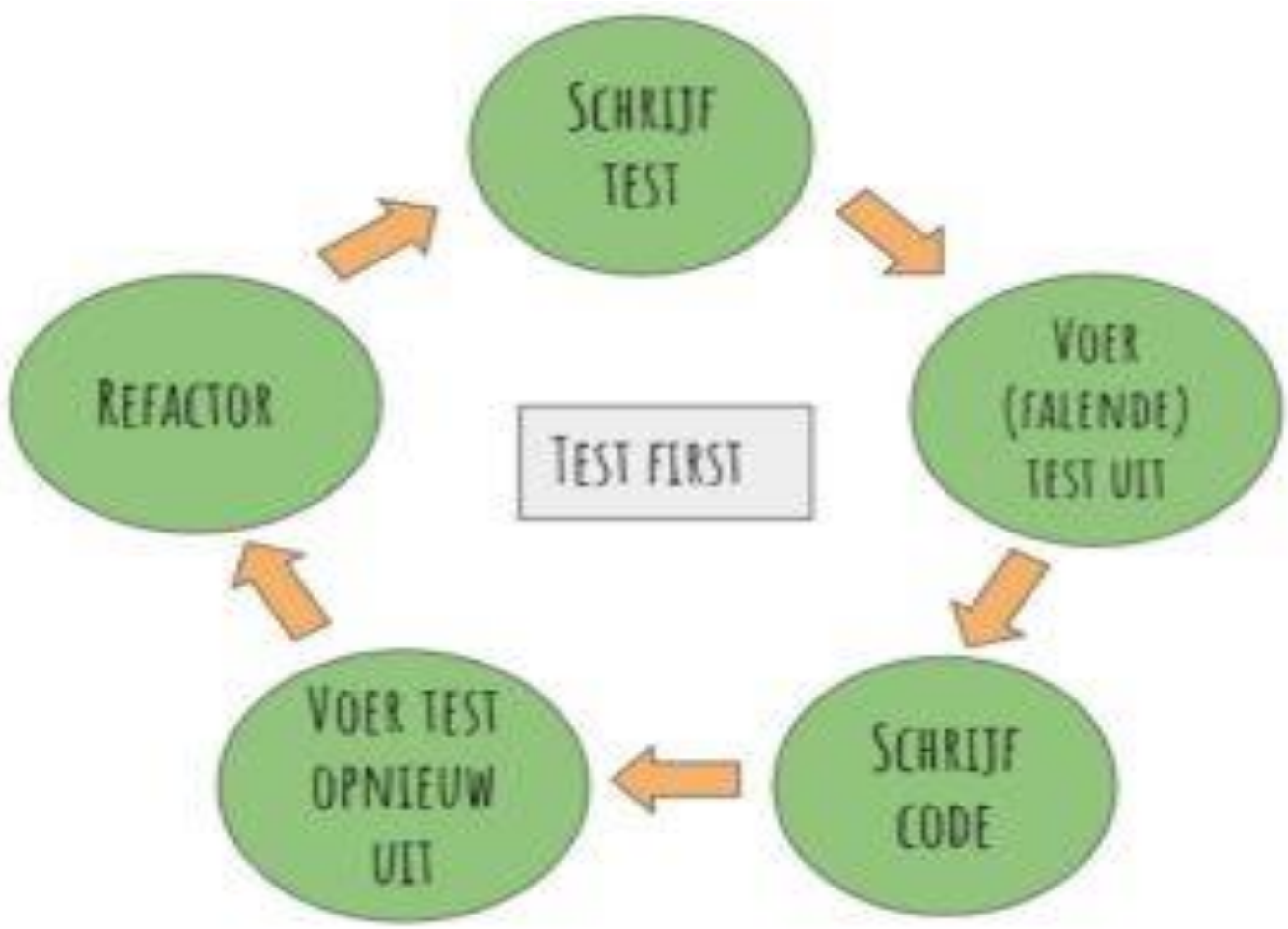
  eerst code schrijven, daarna de testen

Nadeel: het testen wordt pas laat in het proces wordt uitgevoerd en blijkt het ook moeilijk te zijn om een goede testdekking te realiseren.

- **Test-first** (TDD, BDD)

eerst testen schrijven, daarna de code

TDD – Test Driven Development



TDD – Test Driven Development

De ontwikkelaar volgt de volgende 5 stappen:

1. **schrijf een unit test die een deel van de software test**
2. **voer de test uit, deze moet mislukken omdat in de code de functionaliteit nog ontbreekt**
3. **schrijf “net genoeg” code, om de test te laten slagen**
4. **voer de unit test opnieuw uit**
5. **als de test slaagt, ga dan verder met de volgende test, en anders herschrijf / wijzig de software om de test te laten slagen**

Het resultaat =volledig geautomatiseerde reeks unit tests

Primaire Doel van TDD

Op voorhand bepalen van concrete en gedetailleerde verwachtingen over het gedrag van de software

Voordelen TDD – Test Driven Development

- vanuit het perspectief van de eindgebruiker
- gebaseerd zijn op de gebruikers-requirements
- Problemen met de bruikbaarheid van de UI kunnen zo
- Overbodige functionaliteiten worden niet geïmplementeerd
- meer vertrouwen bij het ontwikkelteam en bij de klant.
- Ondanks de extra code nodig is voor de tests, zal de ontwikkelingstijd toch korter worden
- fouten worden in vroeg stadium gevonden
- onderdelen van de code worden los van elkaar getest, minder afhankelijkheden en dus minder complex

Nadelen TDD – Test Driven Development

De programmeur schrijft de tests als de code voor de applicatie.

- Indien de programmeur iets vergeet, wordt zowel de test als in de code over het hoofd gezien.
- Wanneer er een groot aantal tests succesvol is, kan dit de indruk geven dat de applicatie volledig getest is.
- Een valkuil is dat een integratie- en systeemtest niet of nauwelijks uitgevoerd worden.

BDD – Behaviour Driven Development

- **voordat** er wordt onwikkeld, wordt eerst het **gedrag** van de software beschreven.

- Het gedrag is beschreven **vanuit het perspectief van de stakeholders** (business, ontwikkelaars, testers, ...)

Het primaire doel van BDD:

= **communicatie** verbeteren, **iedereen begrijpt wat er ontwikkeld moet worden** voordat er gestart wordt met de ontwikkeling.

- De **business/analysten** moeten in staat zijn om de **gewenste functionaliteit eenduidig te beschrijven**
- De **ontwikkelaars** moeten **de functionaliteit begrijpen die ze moeten ontwikkelen**.

BDD – Behaviour Driven Development

User Stories voor beschrijving van gedrag, bv:

ALS een <rol>,
wil ik <een doel bereiken>
zodat <klantwaarde>

Ook worden concrete voorbeelden uitgewerkt in de vorm van scenario's, bv:

Scenario 1: Title

Given [context]
And [some more context]
When [event]
Then [outcome]
And [another outcome]

Voordelen BDD – Behaviour Driven Development

- Goede samenwerking
- Gericht op toegevoegde waarde
- Lagere kosten
- Traceerbaarheid

Referenties

https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019

https://www.youtube.com/watch?v=hsfVPPYoc9o

https://jakeydocs.readthedocs.io/en/latest/mvc/controllers/testing.html

https://raaaimund.github.io/tech/2019/05/07/aspnet-core-unit-testing-moq/

https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit

https://wakeupandcode.com/unit-testing-in-asp-net-core/