

Testen

Contents

1	Testen	2
1.1	Wat is software testing?	2
1.2	Waarom testen we?	2
1.3	Application Lifecycle.....	3
1.4	Exercise : hoe begin je eraan?.....	5
1.5	Agile Testing Quadrant.....	5
2	Formeel test plan.....	6
2.1	Wat is een formeel test plan.....	6
2.2	Voorbeeld	6
2.3	Test Cases	7
2.4	Exercise : maak 1 Testcase	8
2.5	Given / When / Then	8
2.6	Organisatie van de test-cases	8
3	Test Management Tool.....	9
3.1	Wat is een Test Management Tool?	9
3.2	TestCollab	9
3.3	Exercise : Testcase in TestCollab	9
3.4	Test uitvoeren	11
3.5	Test Executions.....	11
3.6	Test management	13
3.7	Reusable Steps	14
4	Defect Management Tool.....	14
5	Opdracht	15

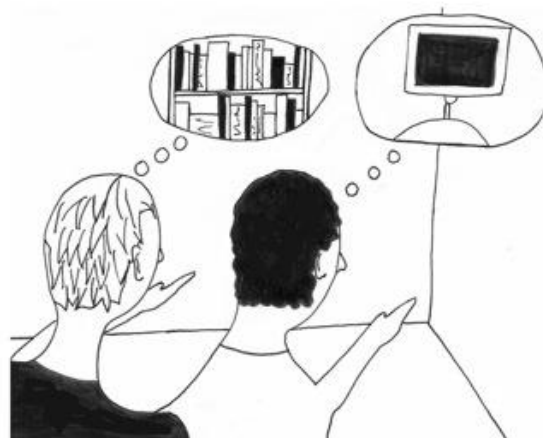
1 Testen

1.1 Wat is software testing?

Software testing involves the **execution** of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the **requirements** that guided its design and development,
- responds correctly to **all kinds of inputs**,
- performs its functions within an acceptable **time**,
- is sufficiently **usable**,
- can be **installed** and **run** in its intended **environments**
- achieves the general **result** its stakeholders desire.

https://en.wikipedia.org/wiki/Software_testing



"Na de brainstorm waren alle betrokken leden van het projectteam 'nieuw archief' het eens over de wenselijke uitkomst."

1.2 Waarom testen we?

A **primary purpose of testing** is to detect software failures so that defects may be discovered and corrected.

Heeft de software voldoende **kwaliteit** om...

- Te releasen naar productie?
- Te releasen naar een bepaalde omgeving?
- Te kunnen starten met een andere category van testen?
- Het factuur te betalen?
- Met een volgend project te starten?

Wat doen we als het antwoord nee is?

- Bug fixen
- Opnieuw testen!

Wanneer is de kwaliteit **voldoende**?

- 100% bugfree?
- Contract?
- Wat zeggen de stakeholders?

Quality?

- QA = Quality Assurance
- QC = Quality Control

1.3 Application Lifecycle

ALM = Application Lifecycle Management

Let op: Application Lifecycle != Software Development Lifecycle

ALM vs. Software Development Life Cycle [\[edit \]](#)

ALM is a broader perspective than the [Software Development Life Cycle](#) (SDLC), which is limited to the phases of [software development](#) such as requirements, design, coding, testing, configuration, project management, and change management. ALM continues after development until the application is no longer used, and may span many SDLCs.

https://en.wikipedia.org/wiki/Application_lifecycle_management

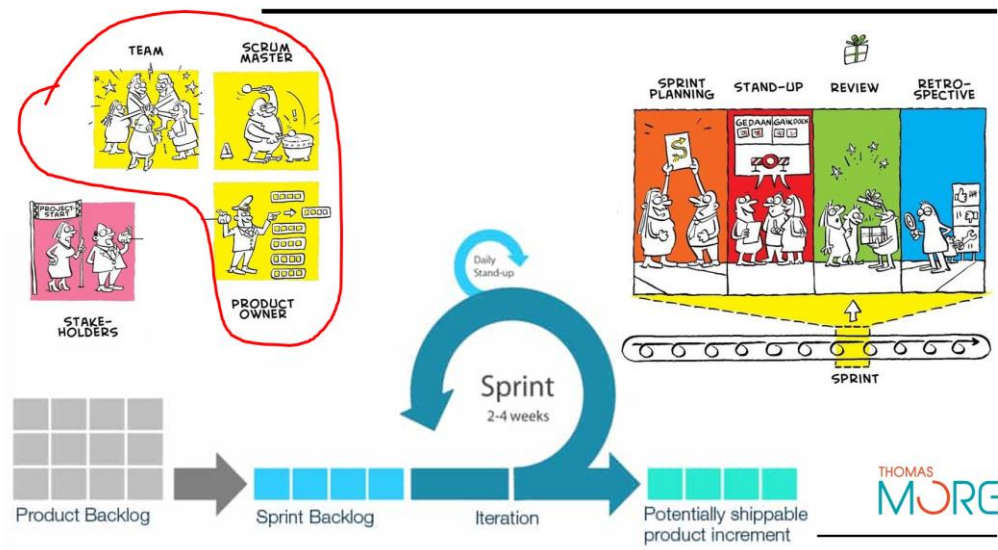
Software Development Lifecycle = Project

Vanuit standpunt van het **werk dat gedaan moet worden**

Voorbeelden van **projecten**:

- Develop nieuwe applicatie
- Develop nieuwe features voor bestaande applicatie
- Revamp bestaande applicatie
- Technologische upgrade van bestaande applicatie

Project Management (bvb Waterfall of Scrum)



Application Lifecycle: vanuit standpunt van een applicatie:

- De eerste versie van de applicatie wordt gemaakt
 - Concept
 - Analyse
 - Development + testing (bvb in sprints)
 - Release
- Operation
 - Monitoring
 - Maintenance
 - Help desk / service center
- Nieuwe features voor bestaande applicatie
 - Concept
 - Analyse
 - Development + testing (bvb in sprints)
 - Release
- End of Life
 - Opkuisen Productie omgeving
 - Opkuisen test omgevingen
 - Opkuisen administratie

1.4 Exercise: hoe begin je eraan?

We gaan een applicatie testen.

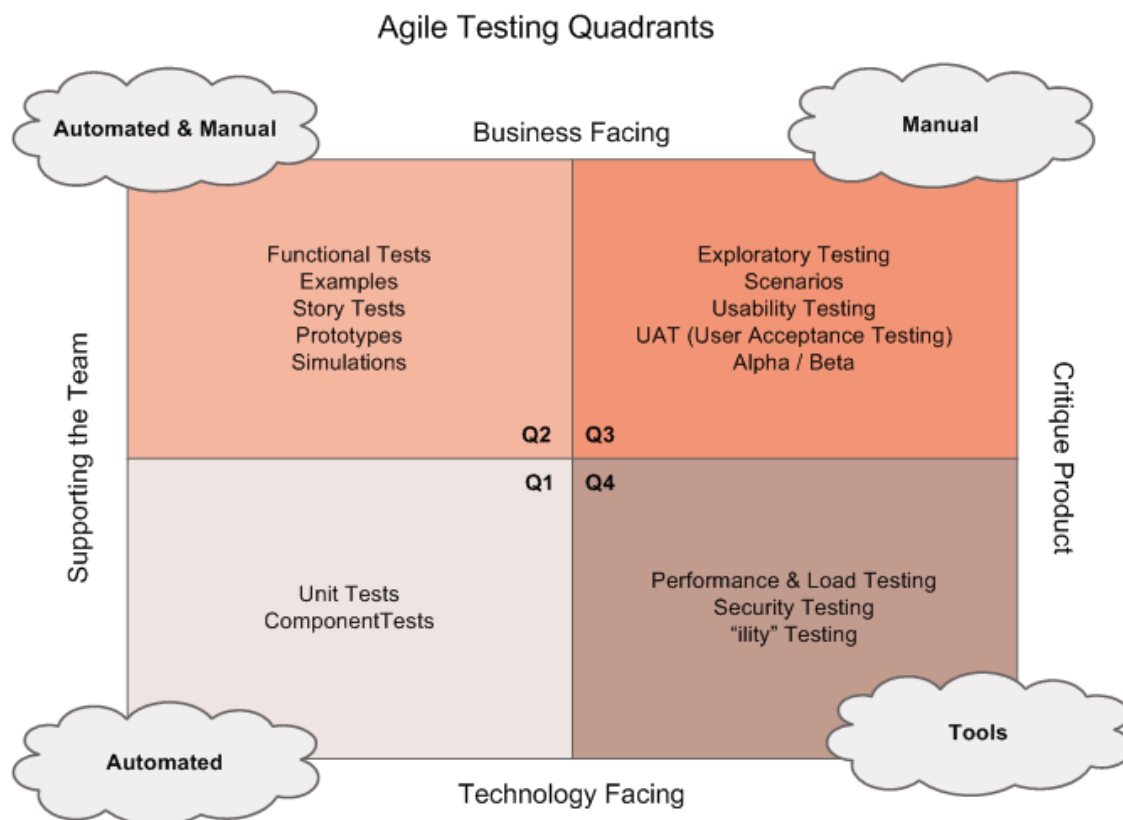
Beschrijving van de applicatie: menuCardStories.pdf.

Hoe begin je hieraan?

Probeer de applicatie al eens uit.

We bespreken dit in groep.

1.5 Agile Testing Quadrant



zie <https://agiletester.ca/>

- Business Facing / Technology Facing
- Supporting the team / Critique the Product
- Functional test: test van een stuk functionaliteit
- Exploratory testing: test cases worden niet op voorhand aangemaakt maar testers checken een systeem "on the fly". Testing as a "thinking" activity.
- Usability testing: is het systeem gemakkelijk te gebruiken? Is het intuïtief voor een gebruiker?
- UAT / Acceptance Testing: voldoet het systeem aan de business requirements? Is het goed genoeg om naar Productie te gaan? Deze test wordt dikwijls uitgevoerd door de Stakeholders en hierop volgt dan een officiële GO.
- Beta testing: een beperkte set van gebruikers in PROD omgeving

- Unit Test: automatische test van een klein onafhankelijk stukje software in isolatie
- Regression Tests: set van tests die je elke keer opnieuw wil uitvoeren

2 Formeel test plan

2.1 Wat is een formeel test plan

Je documenteert de tests die je uitvoert voor een applicatie **op een standaard manier**.

Voordelen:

- Het is duidelijk wat er verwacht wordt van de applicatie
- Duidelijk wat er gedaan moet worden
- Mogelijk om een planning te maken
- Mogelijk om het werk te verdelen
- Communicatie over de uitgevoerde tests
- Rapportering
- Volledigheid van de testen

Wat is:

- **Test case** = 1 afgebakende handeling die een duidelijk zichtbaar **expected result** heeft
- **Test suite** = een verzameling test cases die bij elkaar horen en meestal samen uitgevoerd worden
- **Test scenario** = high-level, korte beschrijving van de functionaliteit die getest moet worden
- **Test plan** = verzameling van test cases, volgens een bepaalde organisatie

Voor een bepaalde situatie (release/bugfix) kan gekozen worden om een bepaald deel van het test plan uit te voeren.

2.2 Voorbeeld

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	<ol style="list-style-type: none"> 1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit 	Userid = guru99 Password = pass99	User should Login into an application	As Expected	Pass
TU02	Check Customer Login with invalid Data	<ol style="list-style-type: none"> 1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit 	Userid = guru99 Password = glass99	User should not Login into an application	As Expected	Pass

2.3 Test Cases

Wat wil je bijhouden ivm Test Case / Test Scenario?

- Test Case:
 - ID
 - Description / Summary
 - Test Steps
 - Expected result
- Meer interessante info voor een test case:
 - Prerequisites: wat heb je nodig om deze test uit te voeren
 - Met welke requirements/story is dit gelinkt?
 - In welke user-flow zit dit? (bvb: login of discovery of buying ...)
 - Author
 - Creation Date: wanneer is de test case aangemaakt
 - Automation
 - Execution:
 - let op: het is altijd de bedoeling dat een test case meerdere keren uitgevoerd kan worden
 - Executor: wie heeft de test uitgevoerd
 - Execution date
 - Welke versie van de software (welk build/welke commit)
 - Welke test-omgeving
 - Pass/Fail
 - Remarks

De bedoeling is dat

- de test gemakkelijk kan **uitgevoerd** worden
- als er een bug is dat het voor de developer duidelijk is hoe de **bug** te **reproducen** en wat er precies fout gaat
- je **conclusies** kan trekken ivm de kwaliteit / stabiliteit van de software

Dus **voor elke aparte test case** wil je weten:

- hoe begin je eraan
- wat heb je allemaal nodig
- alle urls etc
- welke stappen volg je om tot een bepaald punt te komen
- input: welke data geef je in
- wat is het resultaat dat je verwacht

Probleem:

- duplicatie....

Lees dit: <https://www.guru99.com/test-case.html>

2.4 Exercise: maak 1 Testcase

Werk 1 test case uit voor de menucard applicatie.
We bespreken dit in groep.

2.5 Given / When / Then

Dit is een formaat dat soms gebruikt wordt om testcases te formuleren.

Given: de pre-requisites, de begin situatie

When: wat doe je

Then: wat is het verwachte resultaat

2.6 Organisatie van de test-cases

Je kan je test-cases op verschillende manieren organiseren:

- Per sprint / per story
- Per functionele flow
- Per release

3 Test Management Tool

3.1 Wat is een Test Management Tool?

Test management tools are used to store information on how testing is to be done, plan testing activities and report the status of quality assurance activities.

https://en.wikipedia.org/wiki/Test_management_tool

Er bestaan verschillende tools voor Test Case Management.

De meeste zijn betalend.

Bvb: HP-QC, TestRail, Xray, Testpad, Testlink, TestCollab...

3.2 TestCollab

Wij gaan werken met Testcollab.

Testcollab is een tool die free is voor personal use (tot 3 users, tot 200 test cases):

- <https://testcollab.com/>
- Signup
- **Let op:** je gaat eerst in een **free trial period** van 30 dagen. Als deze periode afgelopen is zal de applicatie je komen vragen om een definitief plan te selecteren. Op dat moment **kies je het free plan**. Daarna kan je gewoon verderwerken. Je kan deze optie kiezen in de "Update Plan" pagina onder "Settings" op het main dashboard.

3.3 **Exercise:** Testcase in TestCollab

Let op: maak deze documentatie in het Engels.

In de meeste bedrijven is de voertaal voor alle documentatie Engels. In de meeste teams heeft al wel eens iemand meegewerkt die geen Nederlands kan...

Zet de test case die je gemaakt hebt in TestCollab.

- Maak een project in TestCollab
 - Ga naar je account dashboard
 - Kies in het menu: Projects
 - Kies Add
 - Geef een naam in. **Open Advanced Settings en kies YES voor "Separate text fields for steps and expected result"**
 - **Let op: dit kan je niet meer aanpassen nadat het project aangemaakt is** – als je deze optie niet kiest zijn de testcases minder overzichtelijk
- Je komt op de Overview pagina van je nieuwe project
- Ga naar de Test Cases voor dit project
- Add Suite – vul de velden in en Save

- Je komt nu automatisch in het “Add Test Case” venster
- Vul de velden in

Save en check hoe het resultaat eruit ziet!

- Tip: je kan de Preview button gebruiken om te zien hoe het resultaat eruit zal zijn
- Tip: definieer voor elke step een Expected Result
- Bedoeling is om de steps zo bondig mogelijk te beschrijven
- Tip: gebruik **Markdown** syntax om je tekst te formatteren:
 - <https://en.wikipedia.org/wiki/Markdown>
 - Markdown is hier in een beperkte vorm aanwezig – niet te veel formatteren
- Tip: gebruik het Help Center als iets niet lukt of als je info nodig hebt

Meer info:

- <http://support.testcollab.com/en/articles/2186205-adding-a-test-case>
- <http://support.testcollab.com/en/articles/2182079-keeping-separate-text-fields-for-steps-vs-single-field>

Maak nog een 2^e test case aan in dezelfde suite.

Mijn eerste testcase ziet er zo uit:

The screenshot shows the TestCollab web application interface. The top navigation bar includes 'menucard', 'Overview', 'Test Cases', and 'Milestones'. The left sidebar has 'Select Suite' and 'Test Cases' sections. The main panel displays details for a test case titled 'menucard product dialog' (ID: 6). The test case description is 'menucard product dialog for anonymous user'. The 'Steps' section contains a table with 5 steps:

Step	Expected Result
#1 open menucard	you see a list of products - grouped in categories and subcategories. check layout
#2 click on a product	product window pops up with extra info for the chosen product
#3 click on the > button (right side of the product window)	info for next product in the list is shown
#4 click on the < button (right side of the product window)	info for previous product in the list is shown
#5 click outside the product window	product window disappears

The bottom of the panel shows an 'Execution History' table with columns: Execution Name, Result, Executed Date, and Executed By. Below the table are buttons for Edit, Duplicate, Automate, and Delete.

3.4 Test uitvoeren

Als je naar

3.5 Test Executions

Je hebt nu een test suite met 2 test cases.

Als je nu de applicatie wil (laten) testen, dan maak je een "Test Execution" aan.

Noot: terminologie in TestCollab:

- **Test Suite** = verzameling van **Test Cases**
- **Test Execution** = uitvoering van een of meerdere Test Suites/Test Cases
- **Test Plan** = kan meerdere Test Executions bevatten

- **Execution Template** = bepaalde selectie van Test Suites / Test Cases die je meerdere keren wil laten uitvoeren

Zie ook <http://support.testcollab.com/en/articles/2190782-executing-test-case>

Noot: wij gaan gewoon met **Test Executions** werken (zonder Test Plan)

- Ga in menu naar Test Executions
- Kies "Create execution"
- Vul titel in
- Next
- Selecteer de persoon die de tests moet uitvoeren
- Selecteer onderaan de Test Suite(s) die uitgevoerd moeten worden
- Save

De Tester krijgt nu een mail.

- Kies de Test Execution
- Klik op "Start Now"
- Voer de Test Cases uit
 - Kies het resultaat: Pass of Fail (of Skip)
 - Je kan ook voor de individuele stappen Pass of Fail aanduiden

- Je kan hier ook opmerkingen toevoegen

Je krijgt een rapportje met de status van de uitgevoerde Test Suites / Test Case voor deze execution.

Bekijk even welke informatie je hier allemaal krijgt.

Als je voor een Test Case extra info toegevoegd hebt, of je hebt een individuele step als “Fail” aangegeven, dan kan dit terugvinden door deze Test case te selecteren in Test Case Management (maar niet rechtstreeks vanuit de Execution).






Fail

step 4 not ok

Test Case Revision Number: 4

Started: 2021 February 03 14:59:32 Ended: 2021 February 03 14:59:54

Time Taken: 36 sec

#	Step	Expected result	Result	Notes
1	open menucard	you see a list of products - grouped in categories and subcategories. check layout		
2	click on a product	product window pops up with extra info for the chosen product		
3	click on the > button (right side of the product window)	info for next product in the list is shown		
4	click on the < button (right side of the product window)	info for previous product in the list is shown		
5	click outside the product window	product window disappears		

Posted on 2021 February 03

3.6 Test management

Organiseer je Test Cases in Test Suites.

Je kan Test Suites ook nesten.

Noot: In de “Test Cases” pagina krijg je ook een knop “Execute” bij een Test Suite. Die genereert automatisch een “Test Execution” voor deze Suite.

Het is dus belangrijk om je Test Suites goed te organiseren.

- Op welk niveau maak je je Test Cases? Voor bovenstaande Test Case “Menucard product Dialog” had ik misschien ook 5 verschillende Test Cases kunnen maken. Zou dat beter zijn?
- Het moet zo gemakkelijk mogelijk zijn om de tests uit te voeren

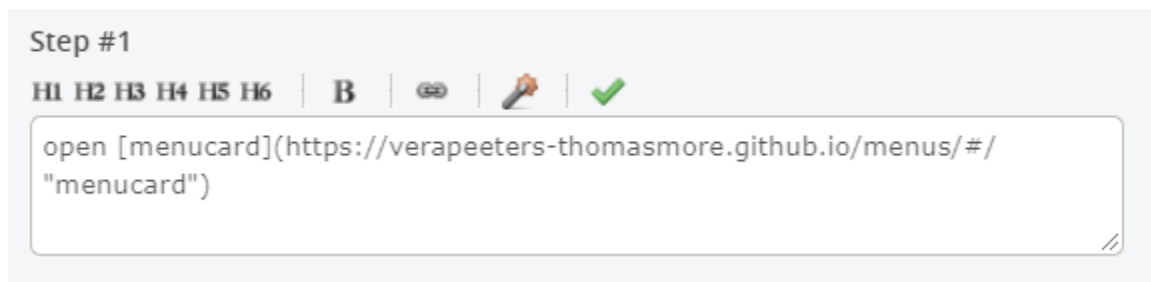
- Het moet zo gemakkelijk mogelijk zijn om een **bug te rapporteren!!!!** De manier waarop de Test Suites / Test Cases en de steps georganiseerd zijn kan hier een grote invloed hebben

Bekijk ook het Test Case Summary Report.

3.7 Reusable Steps

In verschillende Test Case heb ik nu een zelfde step gebruikt:

Steps*



Als de url van mijn applicatie verandert moet ik dit aanpassen in alle steps waar deze url gebruikt wordt....

Naarmate het test plan uitgebreider wordt zullen er meer en meer steps terugkomen. We kunnen in TestCollab reusable Steps definiëren.

- Ga naar "Test Cases"
- Kies Reusable Steps
- Add Reusable Step
- Vul de velden in
- Save

Deze Step kan je nu gebruiken in je Test Cases.

Noot: op dit moment (feb 2021) zit er in Test Collab nog een bug: als je een aanpassing doet in een reusable step dan wordt dat niet aangepast in de Test Executions van de Test Cases die die step gebruiken. Is de bedoeling dat dat gefixt wordt.

Zie ook <http://support.testcollab.com/en/articles/2186188-reusable-steps>

4 Defect Management Tool

Een Test Case Management tool wordt dikwijls gebruikt in combinatie met een Bug Tracking Tool (bvb Jira, YouTrack, Bugzilla, Mantis, Testrail, ...)

Noot: Het is wel handig als er integratie voorzien is tussen de verschillende tools die je gebruikt.

Ook integratie met een deploy/pipeline systeem is handig.

Dat is buiten de scope voor deze cursus.

5 Opdracht

- Tool: testCollab
- Alle administratie ivm software development wordt in het engels gedaan. Ik trek geen punten af voor taalfouten.
- Maak een test plan
 - Minstens **10 test cases**
 - Minstens **3 test suites**
 - Complexiteit test cases moet vergelijkbaar zijn met “menucard product dialog” (niet veel eenvoudiger, maar ook niet veel complexer)
- Applicatie die je test: **MenuCard frontend**
 - <https://verapeeters-thomasmore.github.io/menus/#/>
 - <https://verapeeters-thomasmore.github.io/menubackend/>
 - Zie menuCardStories.pdf
 - Noot: Je moet menubackend niet testen maar zal het wel nodig hebben om functionaliteit van frontend te testen
- Voer het volledige test plan uit
- Rapportering:
 - Genereer een rapport van het uitgevoerde test plan
 - Maak een lijst met duidelijke beschrijvingen van de gevonden issues
 - maak daarbij een **korte** samenvatting:
 - wat is de status van het project
 - wat zijn je aanbevelingen (om dit product te releasen in productie)
- Maak ook een kort verslag over je persoonlijk ervaringen ivm deze opdracht:
 - 2 dingen die je gemakkelijk vond bij deze opdracht (en waarom)
 - 2 dingen die je niet gemakkelijk vond (en waarom)
- Opdracht in Canvas “**Test plan**”