Programmeertechnieken & testen

Wat is testen?

Go to **www.menti.com** and use the code **19 37 58**

Wat is testing?

Mentimeter

Pause scroll

Slide is not active Activate

0

Waarom testen?

Go to **www.menti.com** and use the code **37 85 17**

Waarom testen?

Mentimeter

Slide is not active Activate

0

Wat is testen?

https://www.youtube.com/watch?time_continue=2&v=TDynSmrzpXw&feature=emb_logo

Developer testing

Testing doeleinde

- Testing to Critique
  - Iets testen dat klaar is
  - “Zijn de specificaties voldaan? Zijn er defecten?”
  - Tester-mindset vs developer-mindset

- Testing to Support
  - Team effort
  - Zo snel mogelijk parallel werken
  - Snel feedback geven

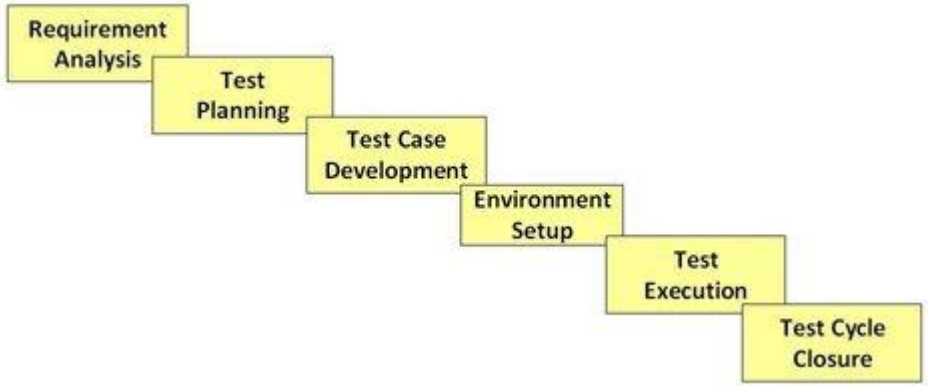
Testing styles

- Traditioneel testing
  - Waterfall model
  - STLC
  - QA engineer verantwoordelijkheid
- Agile testing
  - Agile development nodig
  - Reactive => Proactive
  - Team verantwoordelijkheid
    - No testing crunch
    - No handovers
    - Local testing expertise
    - Weinig tot geen breinloos werk

STLC

STLC

- Testing op het einde is **not the way to go**!
- STLC (Software testing life cycle) helpt ons om planmatig en efficiënt met testing om te gaan.
- STLC bestaat uit 6 fases
- Elke fase heeft zijn eigen entry en exit criteria



Requirements analysis

- Entry
  - Requirement document, acceptance criteria, applicatie architectuur beschikbaar
- Activiteit
  - Bestuderen van de requirements met testing in het achterhoofd
  - Wanneer iets niet duidelijk is, met stakeholders afstemmen
- Exit
  - RTM (requirements tracebility matrix) voorbereiding https://www.opencodez.com/software-testing/create-requirement-traceability-matrix-rtm-free-sample-download.htm
  - Test automation haalbaarheidsstudie

Test planning

- Entry
  - Requirement document, RTM, Test automation feasibility document
- Activiteit
  - Test strategy samen stellen (op basis van budget en resources, meestal door senior QA manager)
  - Tools kiezen
- Exit
  - Test plan https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2007/07/sample-test-plan-template.pdf
  - Effort inschatting

Test case development

- Entry
  - Requirement document, RTM en test plan, automation analyse
- Activiteit
  - Schrijven test cases
  - Reviewen van test cases
  - Test data creëren wanneer nodig
- Exit
  - Test cases
    https://www.guru99.com/test-case.html
  - Test data

Test environment setup

- Entry
  - System design, architecture documents, environment set-up plan
- Activiteit
  - Opzetten van de omgeving om te testen
    - Software, hardware
  - Test data voorzien
    - Test servers
  - Smoke test
    - Testing critical functionalities
- Exit
  - Omgeving om te testen klaar + running app
  - Smoke test results

Test execution

- Entry
  - RTM, Test plan, test cases, test environment, test data set up, unit test report voor de te testen build
- Activiteit
  - Uitvoeren van testen op basis van testplan en test cases
  - Bugs rapporteren
  - Hertests bugs
- Exit
  - RTM met status
  - Test cases geupdate
  - Bug rapport

Test cycle closure

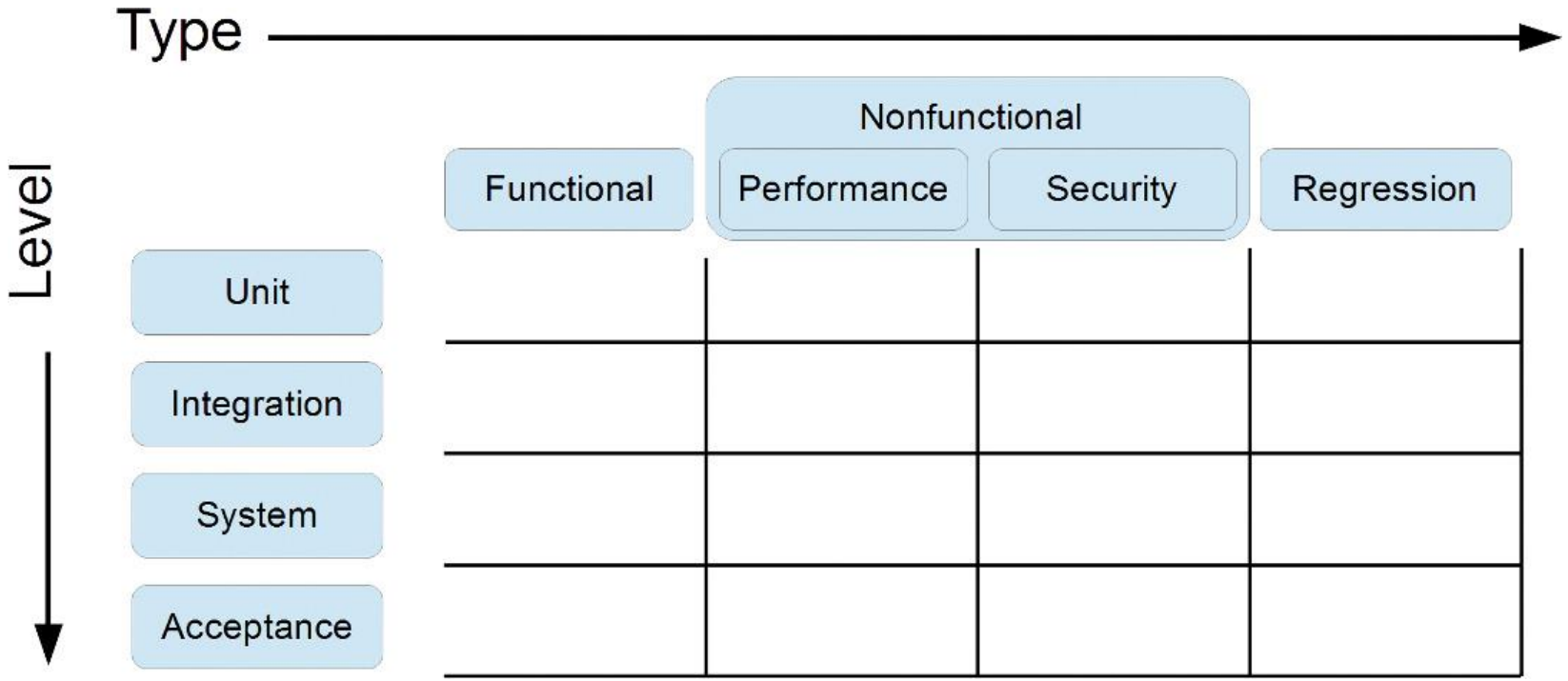
- Entry
  - Testing afgerond, test results beschikbaar, defect logs zijn beschikbaar
- Activiteit
  - Verzamelen van gegevens
  - Test closure rapport creëren
  - Test resultaat analyse om ernst en type bugs te definiëren
- Exit
  - Test closure report https://www.opencodez.com/software-testing/6-easy-steps-to-write-test-summary-report.htm
  - Test metrics

Test levels & types

Hoe testen?

- Er zijn meer dan 100 soorten test technieken (https://www.guru99.com/types-of-software-testing.html)
- Niet elk type test is steeds aan de orde
  - Veel hangt af van de scope en het type project
- Ruwweg kunnen we testen opdelen in 4 types
  - Functional testing
  - Non-functional testing
    - Performance
    - Security
  - Maintenance testing
- Daarnaast kan elke type test op 4 niveaus getest worden
  - Unit test
  - Integration test
  - System test
  - Acceptance

Test levels en types



http://www.informit.com/articles/article.aspx?p=2730111&seqNum=3

Unit tests

- Testen van een klein deel van de code
- Low-level
- Geschreven door developers
- Unit test heeft de scope van:
  - Methode
  - Class
  - Cluster van samenwerkende klasses

“. . . a reasonably large-scale code structure within an application, with well-defined API, that could potentially be swapped out for another implementation”

Integration test

•Tracy: Have you tested that the complex customer record is written correctly to the database?
•David: Sure! I wrote a unit test where I stubbed out the database. Piece of cake!
•Tracy: But the database contains both some triggers and constraints that could affect the persistence of the customer record. I don't think your unit test can account for that.
•David: Then it's your job to test it! You're responsible for the system tests.
•Tracy: I'm not sure whether the database is a "system." After all it's your way of implementing persistence. And besides, wouldn't you want to be certain that persisting the complex customer record won't be messed up by somebody else on the team? Sure, I can test this manually, but there are only so many times I can do it.
•David: You're right, I guess. I need a test that runs in an automated manner, like a unit test, but more advanced. It must talk to the database. Hmm … Let's call this an integration test! After all, we're integrating the system with the database.
•Tracy: …

Integration test

- Vaak gekoppeld aan code
- Ruimer dan een unit test
- Complexer dan unit test
- Taak van een developer
- Meerdere units samen testen

System tests

- Verifiëren van het volledige systeem
- Vaak black box testing
- Meestal uitgevoerd door QA engineers/testers.

Acceptance tests

- Eind gebruiker validatie
- Controleren of de requirements en de verwachtingen voldaan zijn

Functional testing

- Test het gedrag van een applicatie.
- Wordt er voldaan aan de functionele verwachtingen.
- Doet de applicatie wat het bedoeld is te doen?
- Doet de applicatie niet wat het niet bedoeld is te doen?

Non-functional testing

- Test de manier waarop iets gebeurd
- Geen features/functionele requirements
- Vb.:
  - Performance
  - Endurance
  - Load
  - Volume
  - Scalability
  - Usability
  - Security

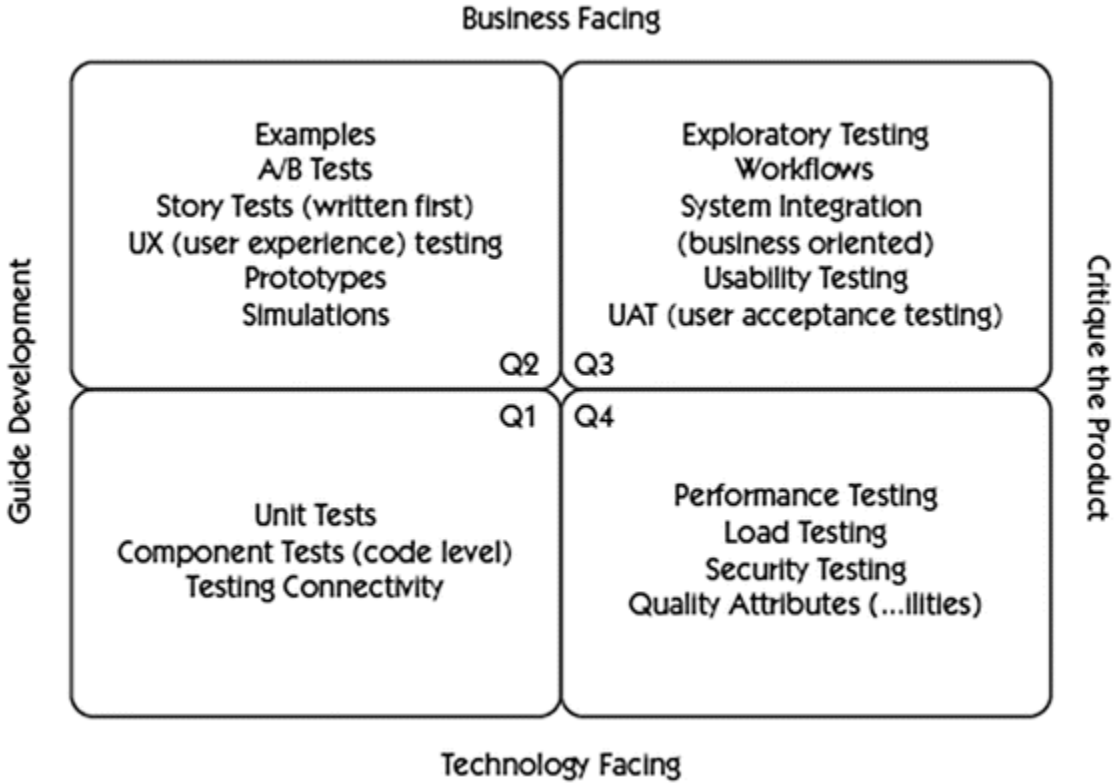
Maintenance testing

Software in ontwikkeling heeft soms refactoring* nodig. Om te voorkomen dat refactoring invloed heeft op de output van de code is er maintenance testing nodig. We testen dus geen nieuwe zaken maar gaan dubbel checken of enkele core features nog werken.

- Vb.:
  - Regressie testing
  - Maintenance testing

*** Refactoring** is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Its heart is a series of small behavior preserving transformations.

The agile testing quadrants



Andere test types

- Smoke testing
  - Triviale zaken testen
  - Enkel het meest belangrijke
- End-to-end testing
  - System testing on steroids
  - Meerdere systemen samen testen
- Characterization testing
  - Bij legacy projecten, testen opstellen op basis van code
- Positive & negative testing
  - Positive testing
    Met correcte data testen en verifiëren dat alles werkt zoals het hoort.
  - Negative testing
    Met incorrecte data testen en verifiëren dat de juiste fouten getoond worden.

Andere test types

- Small, medium & large tests
  - Om verwarring te voorkomen heeft Google een eigen systeem ontworpen voor termen zoals unit, system, integration tests.
  - Small test (~Unit test)
    - Klein en snel
    - Niet toegelaten om externe systemen aan te spreken
    - Binnen 60 seconden uitgevoerd
  - Medium test
    - Interactie testen tussen verschillende lagen
    - Binnen 300 seconden uitgevoerd
  - Large test
    - Geen limitaties

7 software testing principles

7 software testing principles

https://www.guru99.com/software-testing-seven-principles.html

Testing shows presence of defects

- Testing toont de aanwezigheid van defecten
- Dus **NIET** de afwezigheid van defecten.
- Analogie hoe bewijzen dat spoken niet bestaan 
- Testing vermindert de kans op niet-gevonden bugs.

Exhaustive testing is impossible

- Te uitgebreid testen is onmogelijk
- Het is steeds een afweging om de optimale hoeveelheid testen te voorzien/uit te voeren
- Vooral ervaring helpt hierbij

Early testing

- Start zo vroeg mogelijk met testen
- Hoe vroeger we testen hoe sneller bugs naar boven komen.
- Beter snel een fout oplossen dan verder bouwen op een fout waardoor we later een grote refactor moeten uitvoeren
- STLC kan hierbij helpen

Defect clustering

- Vaak komt het voor dat de meeste fouten zich slechts in een klein deel van de applicatie bevinden.
- Het pareto principe => 80/20 regel.
- 80% van de fouten/bugs bevinden zich in 20% van de modules/software.

Pesticide paradox

- Herhaaldelijk dezelfde pesticide gebruiken tegen insecten veroorzaakt een vorm van resistentie.
- In software steeds dezelfde cases testen zorgt er ook voor dat deze testen na verloop van tijd zullen blijven slagen
- Het is dus nodig om regelmatig testcases te reviewen en herschrijven
- Testers gaan ook steeds hun testmethodes moeten vernieuwen.

Testing is context dependent

- Er is niet 1 manier om te testen
- Testing hangt van verscheidene zaken af

Absence of errors - fallacy

- Het bedrieglijk zijn van afwezigheid van errors
- Soms kan software 99% bug free zijn en toch onstabiel zijn door het testen tegen de verkeerde requirements
- Naast het vinden van bugs is software testing ook het verifiëren van het correct werken van de business rules.