

Samenvatting testing

Martijn Meeldijk



20 januari

ucll

# Why testing?

Software that does not work correctly can lead to many problems, e.g.

* loss of money
* loss of time
* business reputation
* injury or death

# Causes of Software Defects (invullen)

* An **ERROR** or a **MISTAKE** are a faulty action by a human
  + The developer misunderstood the specification or implemented it in the wrong way or just made a typo (e.g. <= instead of <), etc.
* This results in a **FAULT** or **DEFECT** in the source code
* When using or testing the system, we notice a **FAILURE** when the system doesn’t do what it is supposed to do.
* The actual behavior does not match the expected behavior

# Quality & the role of Testing (invullen)

* Testing will help **measure** the quality of software
* Testing can give **confidence** in the quality of the software if it finds few or no defects
* A properly designed test that passes, **reduces** the overall level of risk in a system
* Defect causal analysis leads to **improved** processes and improves the quality of future systems
* When testing does find defects, the quality of the software system **increases** when those defects are fixed

# Seven Testing principles

1. Testing shows the presence of defects
2. Exhaustive testing is impossible
3. Early testing
4. Defect clustering
   1. Defects come in groups (like rats)
5. Pesticide paradox
   1. Same tests over and over, no new defects found (like pesticide)
6. Testing is context dependent
7. Absence-of-errors fallacy
   1. Finding a large amount of errors does not ensure quality

# Waterfall model

* System Feasibility
* Requirements Analysis
* High Level Design
* Detailed Design
* Coding
* Integration
* Installation
* Operations and Maintenance

# A screenshot of a cell phone Description automatically generatedV-model

# A screenshot of a cell phone Description automatically generatedAgile Scrum

# Test levels

## Component testing

= unit testing

## Integration testing

= testing of interface between components or interactions between different parts of a system

## System testing

= Test the whole system or product

## System integration testing

Test integration between different systems: usually after system testing

## Acceptance testing

=goal is to establish confidence in (parts of) the system

# Test types

## Functional testing

=what the system does (black-box)

## Non-functional testing

=how the system works (performance, …)

## White-box testing

Systems internal structure

## Change-related testing

Based on changes made to a system

## Confirmation testing

Testing for fixes

## Regression testing

Repeat testing after modification, to discver defects caused by changes

# Black-box techniques

## Equivalence partitioning

= subset of domain in which attribute values for the test are equivalent

110kg -> heavyweight

200kg -> heavyweight, thus treated the same

## Boundary value analysis

Determine test case when boundary values are important

## Algorithm test

Test structures of a program (determine all paths)

## Decision table test

Dit slaan we mooi over

# Exploratory testing

To get insight in the system, by asking the software questions.

(heel korte uitleg)

# Static vs dynamic testing

## Static testing

* Manual examination
* Static analysis, without executing (can be automated)

## Dynamic testing

* Opposite of static testing
* By executing software