Introduction to Testing

Test

PBA Softwareudvikling/BSc Software Development

Ronnie Dalsgaard roda@cphbusiness.dk

Tine Marbjerg tm@cphbusiness.dk

Spring 2018

Today's Topics

- **Practical info**
  - Git — Semester plan
  - TimeEdit — Basic schedule
  - Moodle — General info (not updated on exam info)
  - UMS — SMS service

- **General introduction to the test course**
  - Course objectives (curriculum)
  - Semester plan
  - Literature
  - Study points

- **Introduction to testing** (Lit: Black chap. 1 + 2)
  - Fundamentals of testing
  - Testing in software life cycle

What do you about testing?

General topics	Technology

What is the Point of Testing?

- ❑ ?
- ❑ ?
- ❑ ?
- ❑ ?
- ❑ ?

…

Curriculums in General–

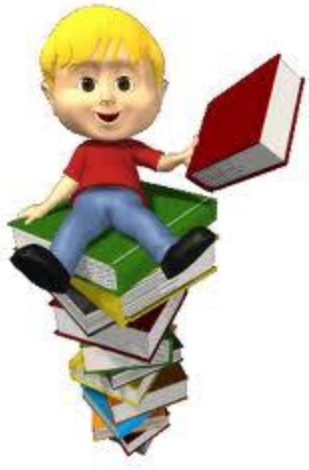
Learning objectives have different levels

EXAMPLE

- Knowledge
  - **Know** about control structures as non sequential control mechanism
- Skills
  - **Design** a "for loop" from scratch to solve a particular problem
- Competences
  - **Evaluate** the quality (e.g. efficiency) of a "for loop" vs. other control structures and use appropriately

PBA Soft Curriculum is on Moodle

Test Course Objectives – Knowledge Level



- Central test strategies and tests models and their role in the system engineering process
- Test as integral part of a development project
- Different test types

Course Objectives –
Skill Level

- Ensure traceability between system requirements and testing at all levels
- Apply black-box and white-box testing techniques
- Apply various criteria for the degree of test coverage
- Use techniques for verification and validation
- Use techniques and tools for automated testing
- Build systems to manage testing and incident management

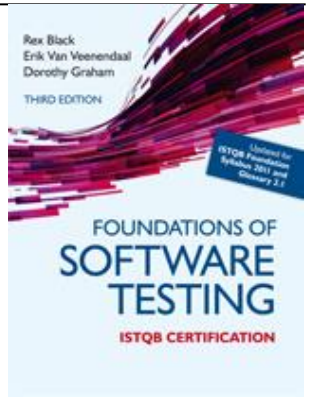
Course Objectives –
Competence Level

- Define, plan and execute testing in a development project that matches the project's quality requirements
- Plan and manage the implementation of internal and external testing of software systems.
- Design for testability

Summing Up Test Competences

- Craftsman skills
  - Like programmers need design principles/patterns
- Test as driver of requirements
- Test as integrated activity in development projects
- Efficiency – we need automation

Course Literature (& Certification options)



- Foundations of Software Testing by Black et al.
  - Book is basis for ISTQB test certifications
    - Certifications/ISTQB
  - Online version of book contents

- ISTQB Curriculum - Learning Objectives based on:
    - K1: remember; K2: understand; K3: apply; K4: analyze

- Online resources for agile and tool oriented topics

Exam

- June (probably 6th -8th)
- 25 minutes individual oral exam
- No preparation time
- You pick question which has both theoretical and practical perspective
- Questions relate to study point assignments

5 questions to you ☺

1. What is a bug?
2. Difference between testing and QA?
3. What is beta testing?
4. What is boundary testing?
5. What is data driven testing?

Difference between testing and QA

- Quality assurance involves the entire software development process and testing involves operation of a system or application to evaluate the results under certain conditions. **QA** is oriented to **prevention** and **Testing** is oriented to **detection**.

- TESTING means 'quality control' of a **product**

- QUALITY ASSURANCE measures the quality of **processes** used to create a quality product.

Data Driven Testing

- Testing in which the action of a test case is parameterized by externally defined data values, maintained as a file or spreadsheet.
- This is a common technique used in Automated Testing

Today's Learning Objectives

- Fundamentals of the test discipline & motivation for testing (Black chap 1)
- Different approaches to testing (Black chap 2)
- Knowledge of basic test terminology

Foundations of Software Testing

Chapter 1
Fundamentals of Testing

Characteristics of Testing

- A *systematic process* which tries to find deviations between the *expected* and the *actual* behavior
- A *destructive* activity
- A *complex* activity
- Often an *underrated* activity
  - Time
  - Qualifications

“Bug”

Bug

Also called defect or fault

http://en.wikipedia.org/wiki/Portal:Software_Testing

Debugging

Once we have observed a failure, we can investigate to find the fault/bug that caused it (i.e. the **root cause**) and correct the fault

When do Defects Arise?

Black fig. 1.1

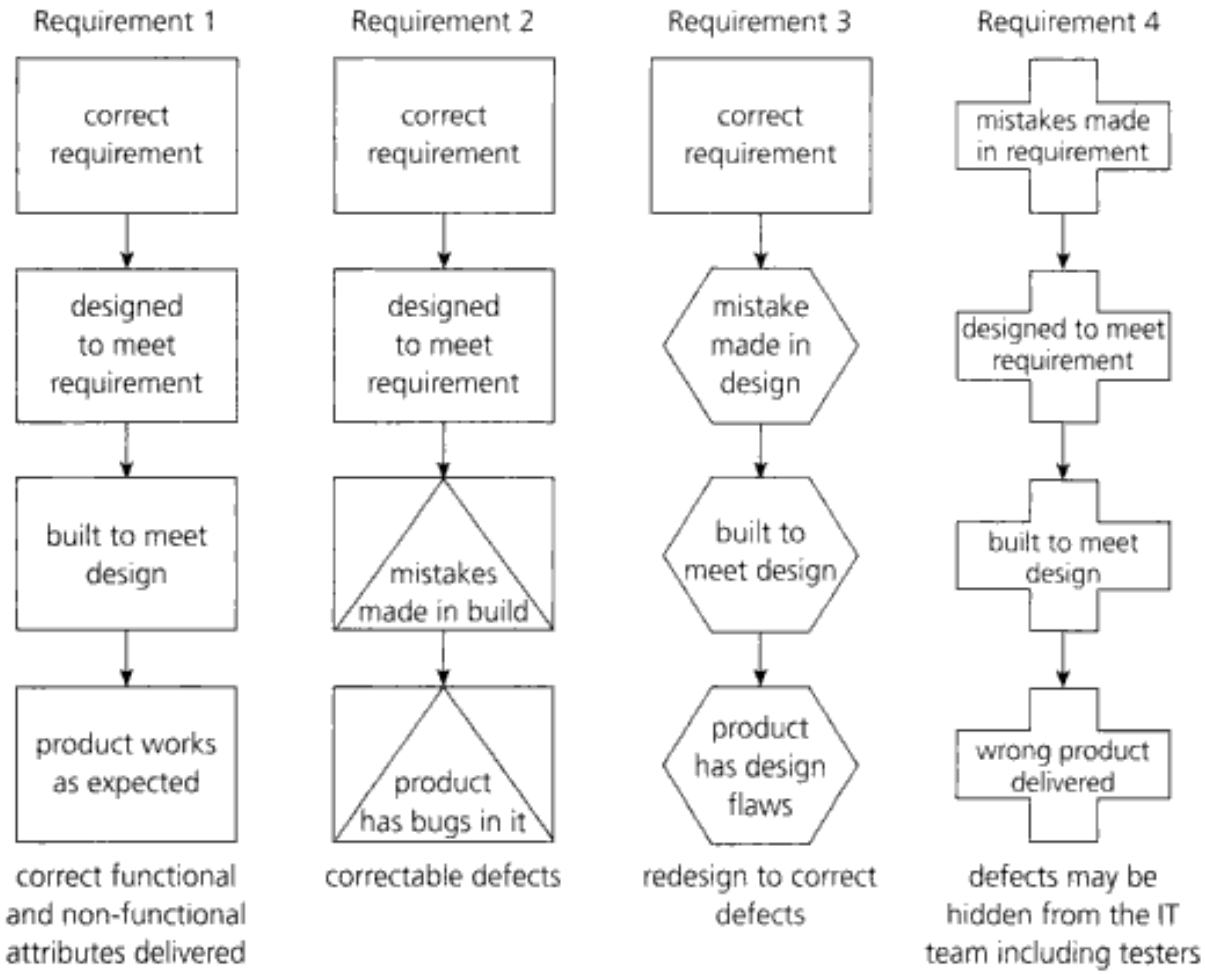


FIGURE 1.1 Types of error and defect

Cost of Defect/Change

- Single biggest cause of project failure is:
  Not getting the requirements right

Building the wrong thing is the most expensive mistake to make!!!

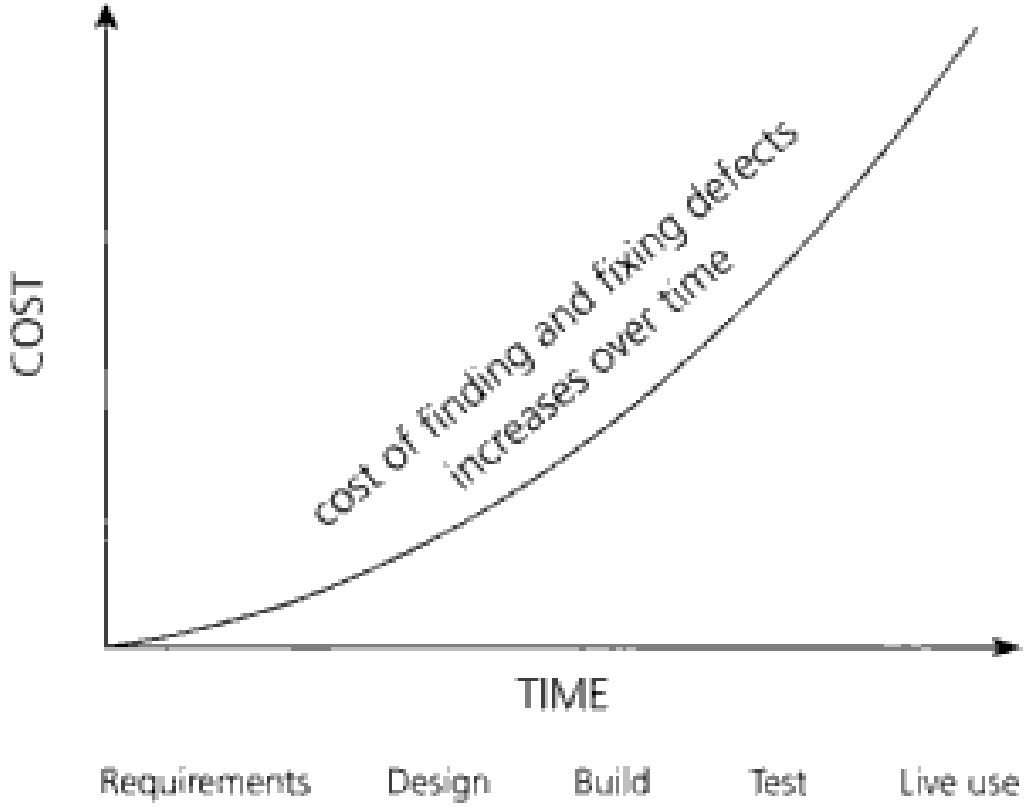


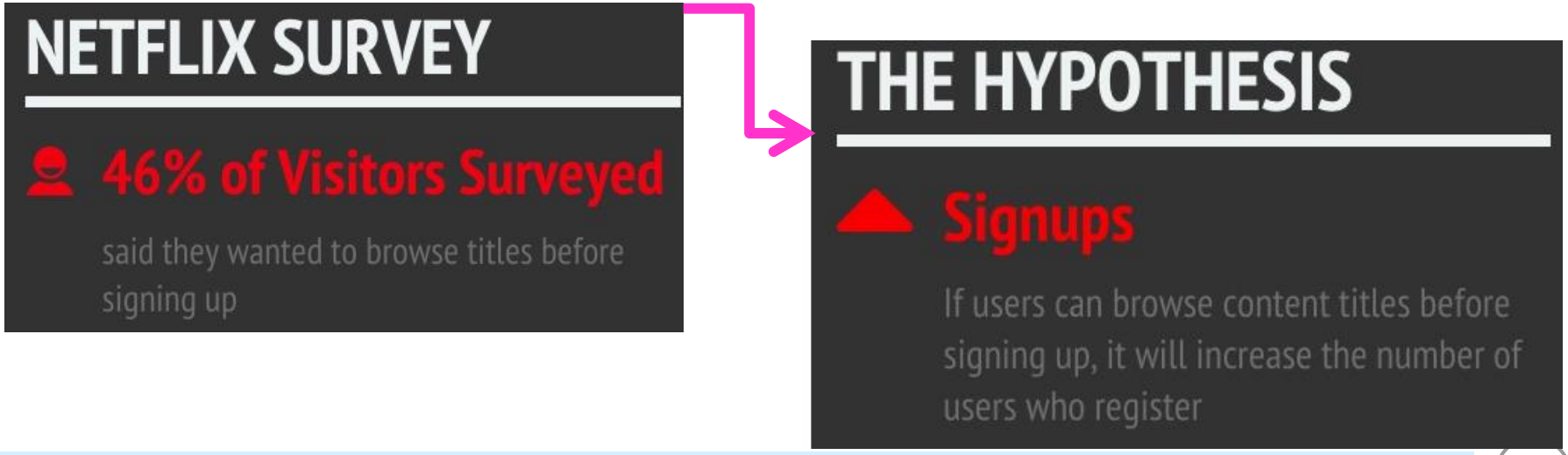
FIGURE 1.2 Cost of defects

Netflix Example 1/3

- *Your assumptions are your windows on the world. Scrub them off every once in awhile, or the light won't come in.*
  – Isaac Asimov

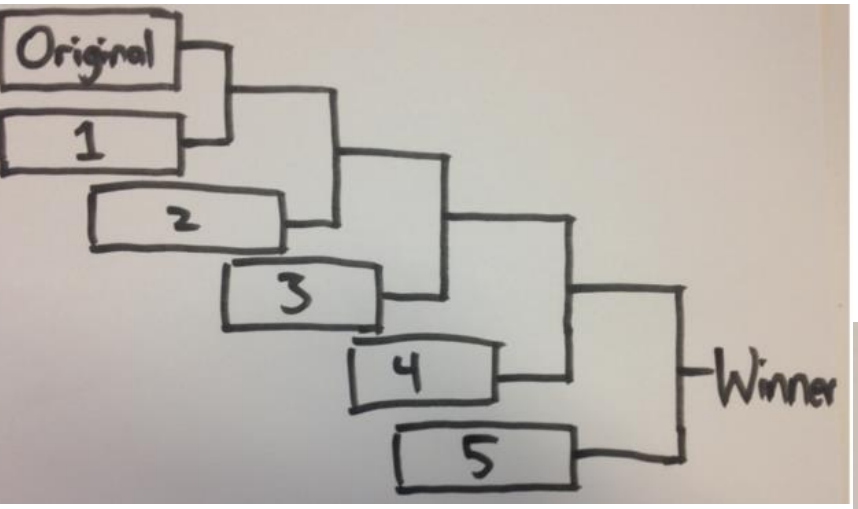
[The Test Results Netflix Never Expected]

"Come on Netflix. Why can't I browse titles before I sign up?"

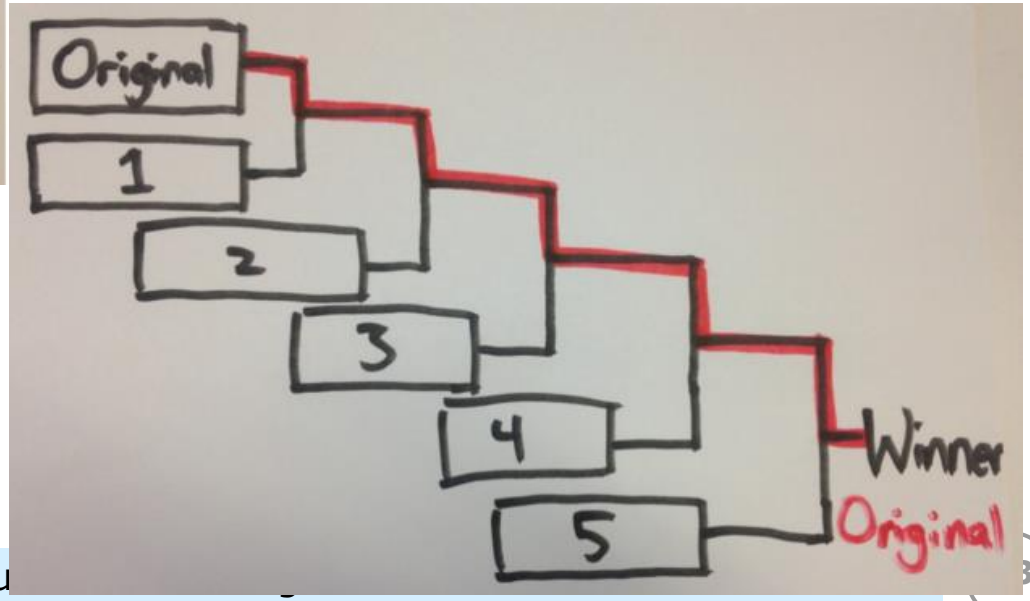


Netflix Example 2 /3

Netflix made experiments …



As they ran the experiments, they found that the original page (without title searching abilities) beat all the other variants



Netflix Example 3 /3

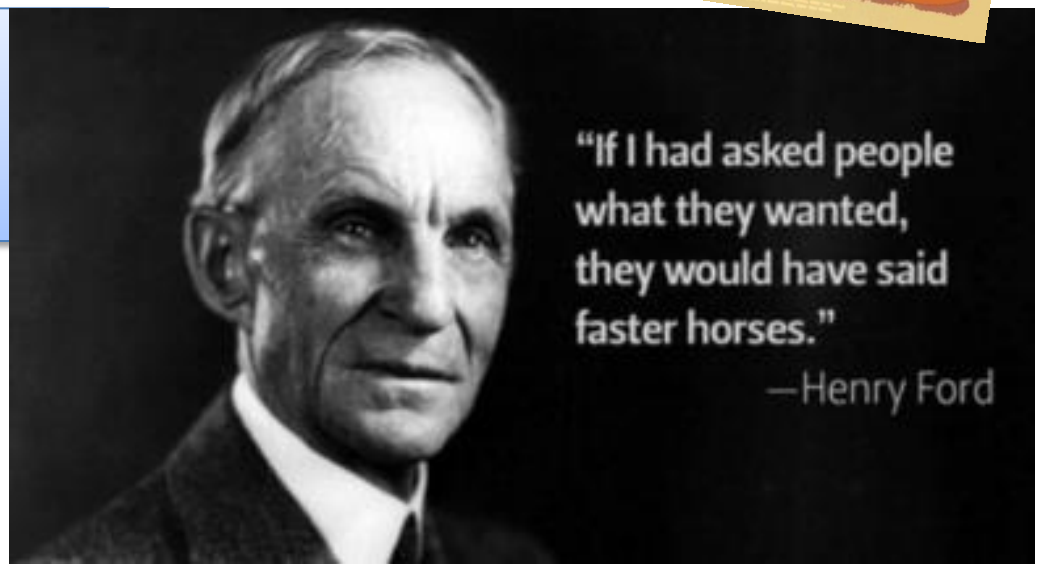
Conclusion 1

Don't Confuse the Meal With the Menu.

- ***Netflix is all about the experience***
- ***Simple choices***

Conclusion 2

Users Don't Always Know What They Want



"Your assumptions are your windows on the world. Scrub them off every once in awhile, or the light won't come in." – Isaac Asimov

Fundamental Test Process

Black sec 1.4

Test Activities are …

similar to activities in traditional system development

- Planning and Control
- Analysis and Design
- Implementation and Execution
- Evaluating Exit Criteria and Reporting (for each test level)
- Test Closure (at software release)

Seven Testing Principles

Black Table 1.1

7 testing principles– find explanation here or google them ☺

Discuss them in groups of 3-4 students (1st & 2nd semester students mingle)

Discuss

1. Testing shows the presence of defects
2. Exhaustive testing is impossible
3. Early testing
4. Defect clustering (80-20 rule)
5. Pesticide paradox
6. Testing is context dependent
7. Absence-of-errors fallacy

Psychology of Testing

- A tester needs the right mindset – this influences the success of testing.

- Following people traits are considered effective:
  - Curiosity
  - Professional pessimism (good understanding of "*nobody is perfect*")
  - A critical eye (slogan: "*if in doubt, it's a bug*")
  - Attention to detail
  - Experience
  - Good communication skills

Summing Up

- Why testing is necessary (Black sec 1.1)
  - Test supports higher quality
- What is testing (Black Sec. 1.2)
  - Test activities
  - Common objectives of testing
    - Defects, confidence, information to prevent bugs
- Fundamental test principles (Black Sec. 1.3)
- Psychology of Testing(Black Sec. 1.4)
  - People influence testing success

Foundations of Software Testing

Chapter 2
Testing throughout the software life cycle

Software Life Cycle

- Test is not isolated activity in a software development project
  - The chosen software development model influences test organization

- <u>Two</u> camps of system development philosophy
  - Lightweight and fast methodologies (agile)
    Time to market is central
  - Disciplined methodologies
    Quality and reliability are central

Waterfall [Royce 1970]

Black fig. 2.1

- Characteristics:
  - Development flows through distinct phases
  - No one has to deal with the consequences of their mistakes if different people in different phases
  - Testers have to make up lost time from previous phases
  - Time lag: it can take a long time for changes in requirements to filter through to the finished product

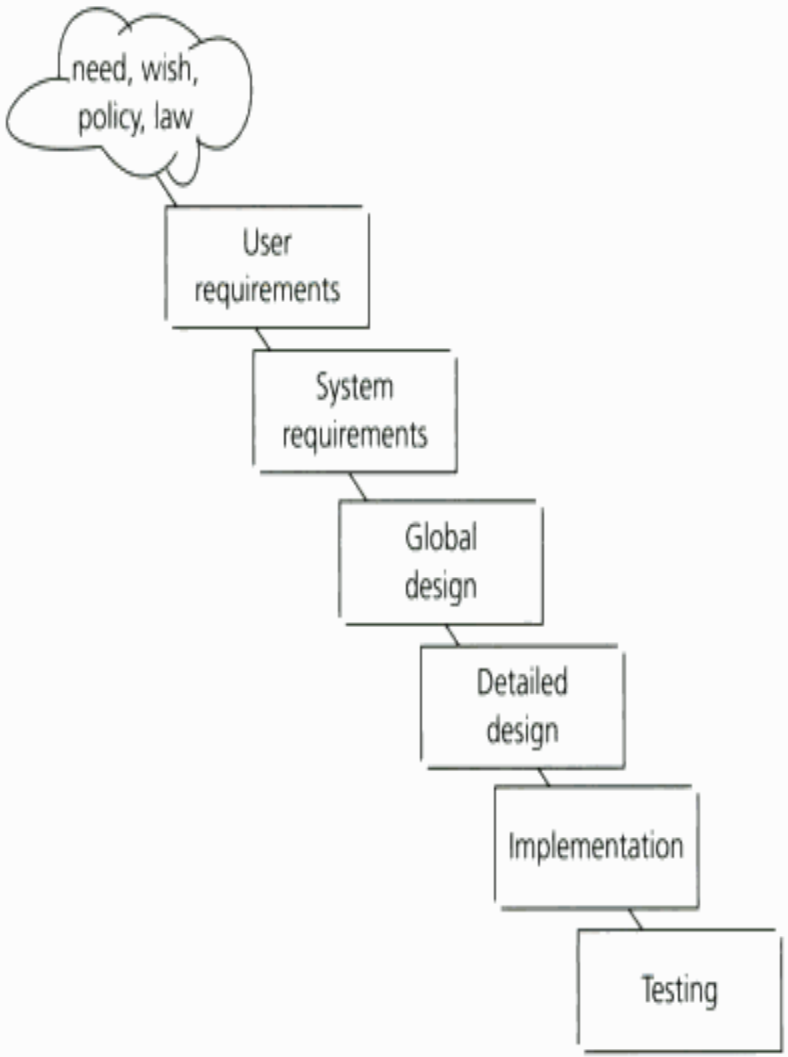


FIGURE 2.1 Waterfall model

V-model – Enhancement of Waterfall

Characteristics:
- Earlier Planning and Execution of Tests

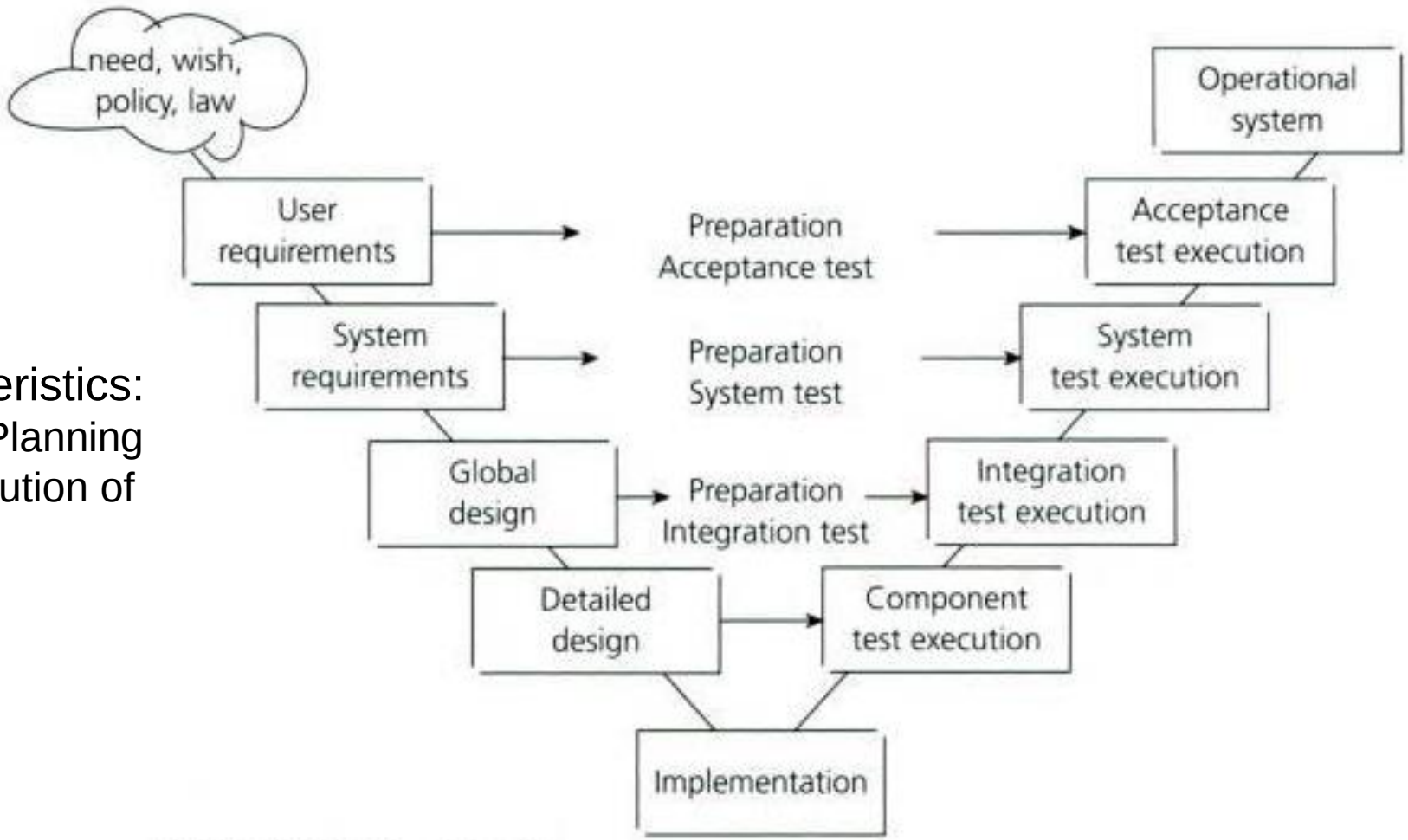
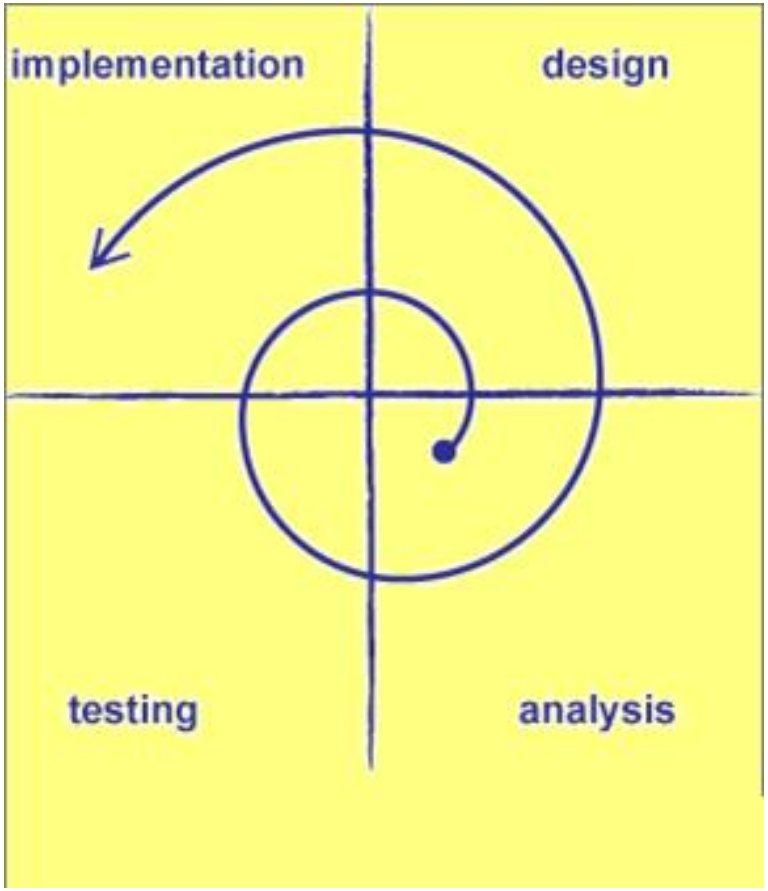


FIGURE 2.2 V-model

Iterative Development Model

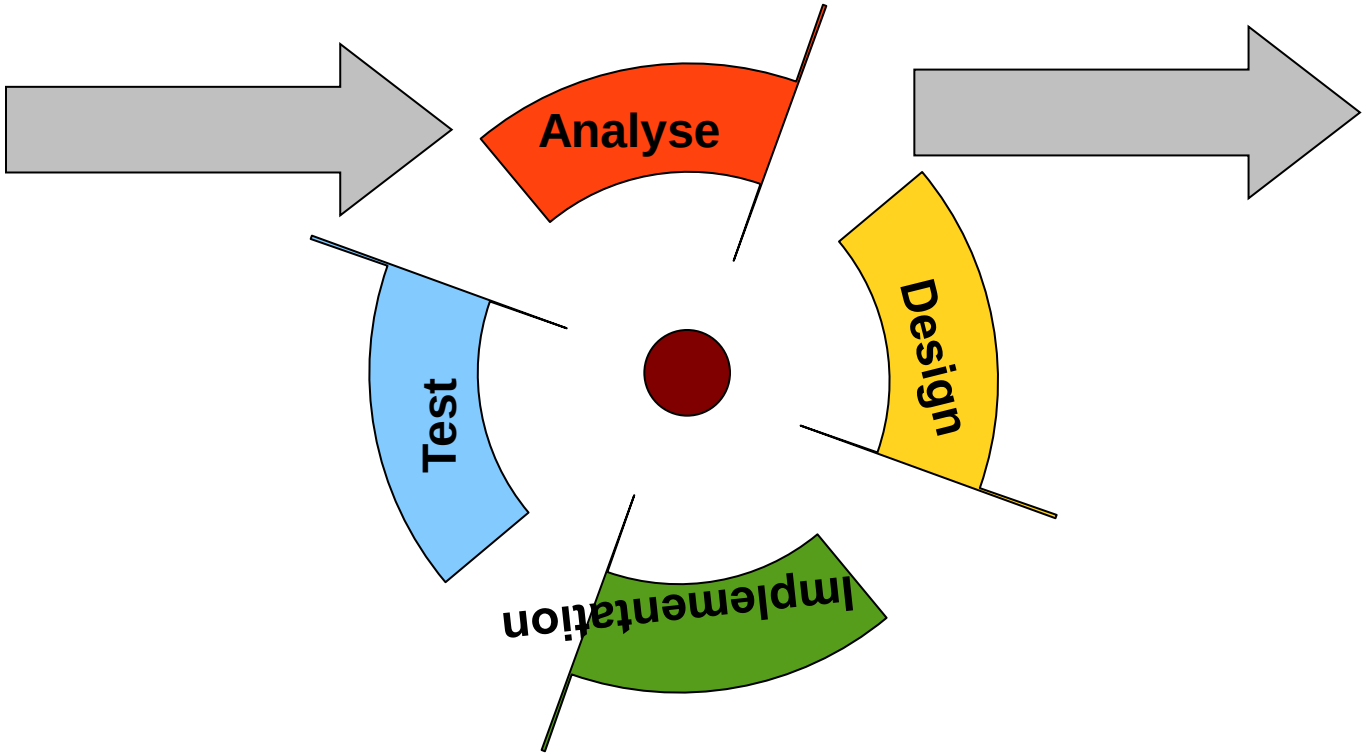
- Characteristics:
  - Many mini water cycles with incremental builds
  - Early feedback - customers can find out what they want by trying out a working system
  - Defects can be found at earlier stage in process
  - Increased regression testing
  - Examples: UP, XP



Agile Development

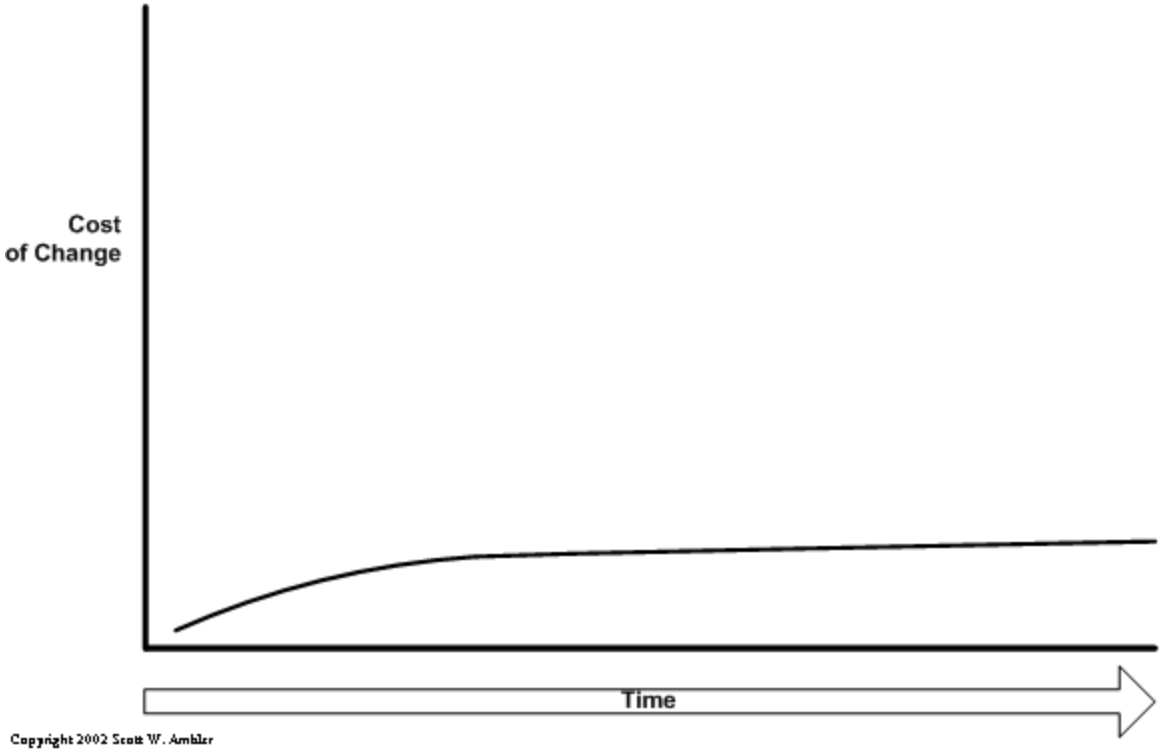
- A group of software methodologies based **on iterative incremental development**, where requirements and solutions evolve through **collaboration** between **self-organizing cross-functional teams**
- Often SCRUM for management + XP principles for development
- The assumption is ever-changing requirements
- Agile manifesto
- Principles Behind The Agile Manifesto
- Principles: e.g. pair programming, test-first, continuous integration and constant refactoring to ensure code quality

Test driven
Incremental Development



Agile Cost of Change Curve

- Kent Beck's cost of change curve



http://www.agilemodeling.com/essays/costOfChange.htm

Test Levels

- Unit (component) testing
- Integration testing
- System testing
- Acceptance testing

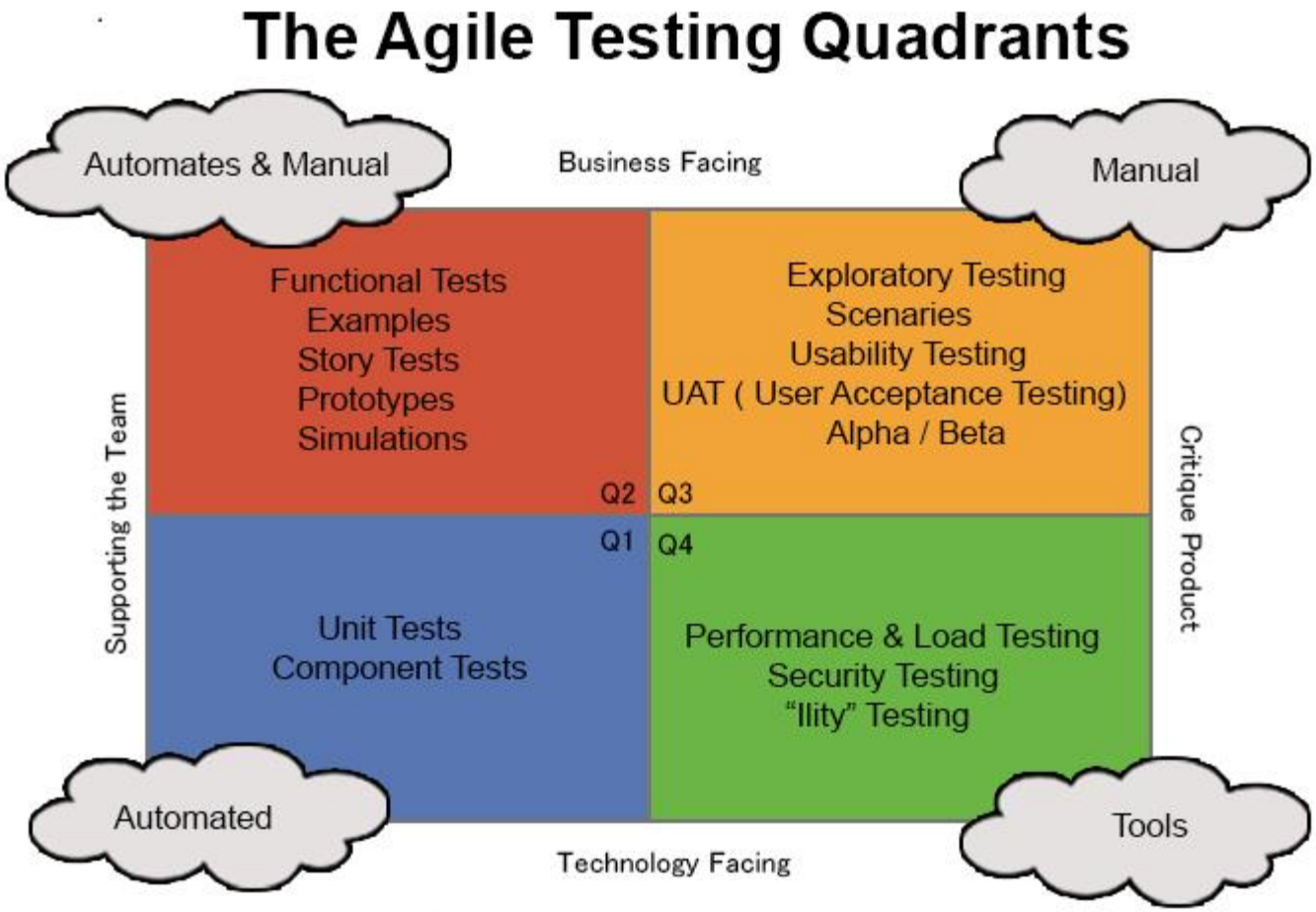
Non-functional Testing

Testing of software characteristics

Typically how well or fast something can be done

- **U**sability testing
- **R**eliability testing (robustness & recoverability)
  - Stress testing
- **P**erformance testing (speed, throughout, scalability)
  - Load testing (scalability)
- **S**upportability
  - Portability testing
  - Maintainability testing

Agile Testing Quadrants



Source: Lisa Crispin, Brian Marick

How do the models relate to each other?

Disciplined testing

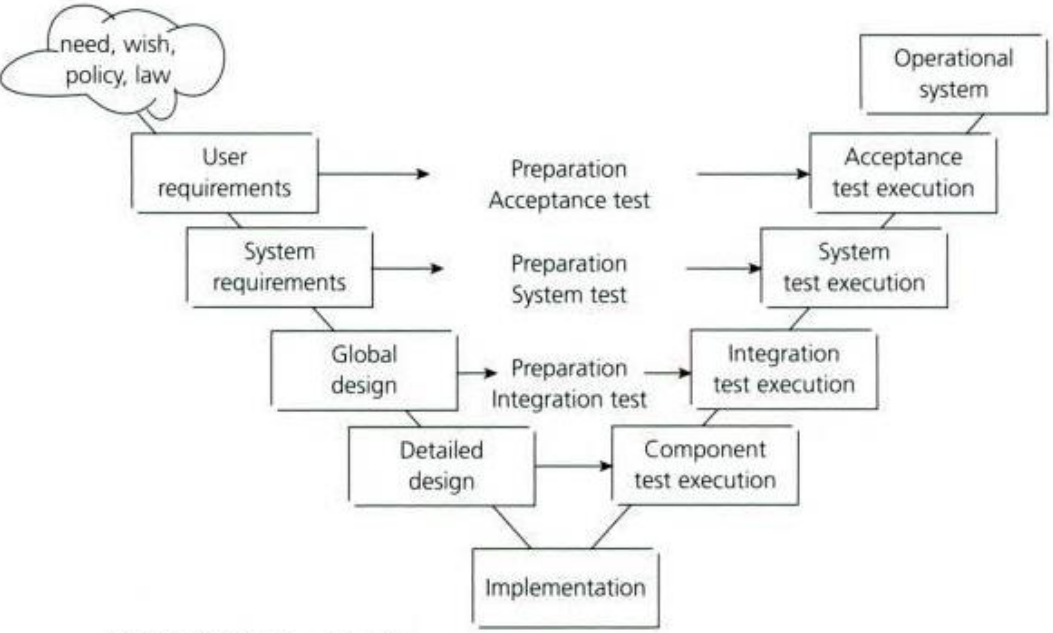
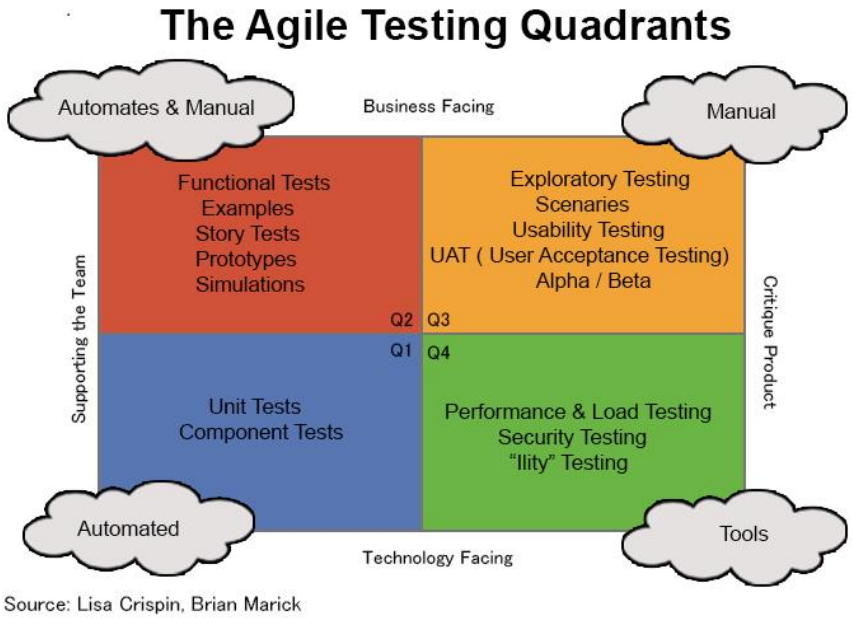


FIGURE 2.2 V-model

Agile testing



Source: Lisa Crispin, Brian Marick

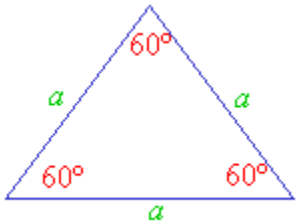
Exercise Time
home work!

Design of test cases

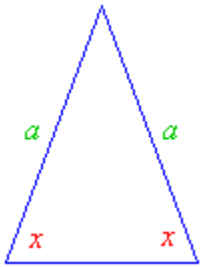
Designing Your First Test Cases

- Make a set of test cases (i.e. specific sets of data) that will adequately test this program:

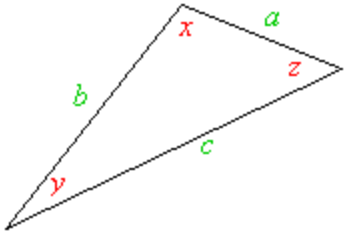
  The program reads three integer values from an input dialog. The three values represent the lengths of the sides of a triangle. The program displays a message that states whether the triangle is scalene (ingen ens sider), isosceles (ligebenet), or equilateral (ligesidet)



An **equilateral triangle** has all three sides of equal length.



An **isosceles triangle** has two sides of equal length.



A **scalene triangle** has no sides of equal length.

Home Work

- Step 1: Design test cases (on paper) for both successful and unsuccessful scenarios

- Step 2: Design and implement the Triangle program in a programming language (e.g. Java or C#)
  - you don't have to write unit tests, but you may do so ☺
  - No need for nice GUI, just console app is fine

- Bring the code next time
- Have an IDE installed on your computer
- Read Black chap. 3