# Test in the Development Lifecycle

Gitte Ottosen
Gitte.Ottosen@Capgemini.dk
Twitter:godtesen

Capgemini

# A bit about me

**Gitte Ottosen**
Capgemini Danmark A/S
Gitte.ottosen@capgemini.dk
+45 52189711

**Education**
Corporal in the Royal Danish Airforce

**Certifications**
SCRUM master, ISEB foundation/practitioner, CAT trainer, Tmap Test Engineer, Tmap Test Manager, TPI Next foundation, SAFe SPC

**Experience**
- 24 years in the IT business
- 7 years in Capgemini Sogeti

**Focus**
Test management, test engineering, SCRUM, process improvement, LEAN, agile, context driven test, change management

**Agile Experience**
Customers: Systematic Software Engineering A/S, Mærsk Line IT, DONG, KMD, TDC

**Network**
Test20/Tecpoint, CAT trainer network
Fellow Sogeti Labs

# How do you want your car tested?

# Why Early Test

| SDLC phaces | Defect Introduction | Defect Detection |
|---|---|---|
| Requirement Specification/Analysis | 55 % | 5% |
| Design | 30 % | 10% |
| Construction and System Test | 15 % | 40% |
| Acceptance test, Production and Maintenance | 0 % | 45% |

*Source:*
*Boehm, Barry W Software Engineering Economics*
*Engelewood Cliffs, N.J: Prentice Hall, Hughes*
*DOD composite Software Error History*

# Why Early Test

If we develop 90% correct

| Requirement | Analysis | Design | Code |
|---|---|---|---|
| 90% correct | 90% correct | 90% correct | 90% correct |

| Accumulated effect whn 90% correct | | | |
|---|---|---|---|
| 90% correct | 81% correct | 72% correct | 65% correct |

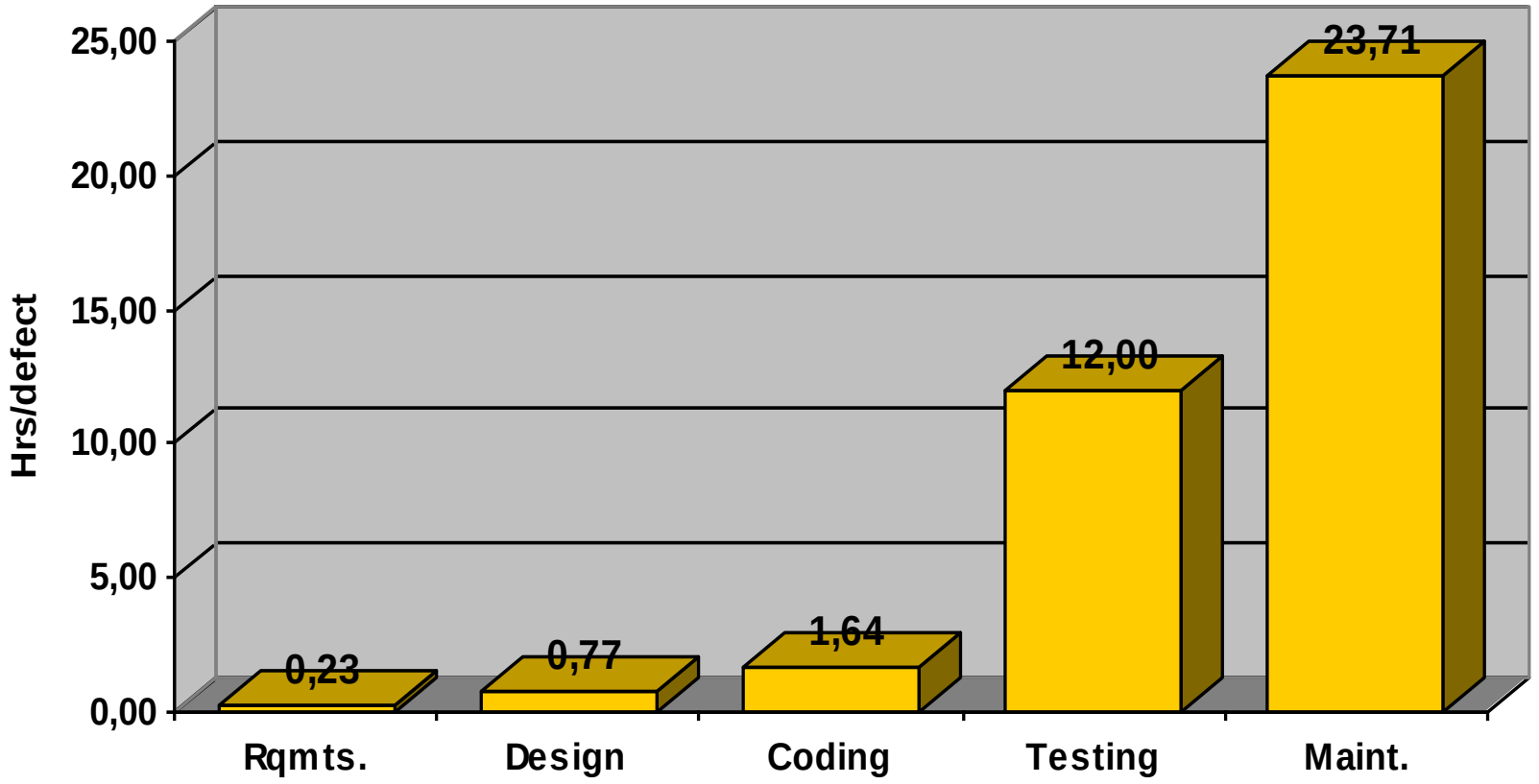
# Why early test

| Requirement | Analysis | Design | Code |
|---|---|---|---|
| 85% correct | 85% correct | 85% correct | 85% correct |

| Accumulated effect with 85% correct | | | |
|---|---|---|---|
| 85% correct | 72% correct | 61% correct | 52% correct |

*Source: Teradyne Software and Systems Test Inc. 1999*

# The Price for Fixing a Bug



# But What is Early Test?

**Review**

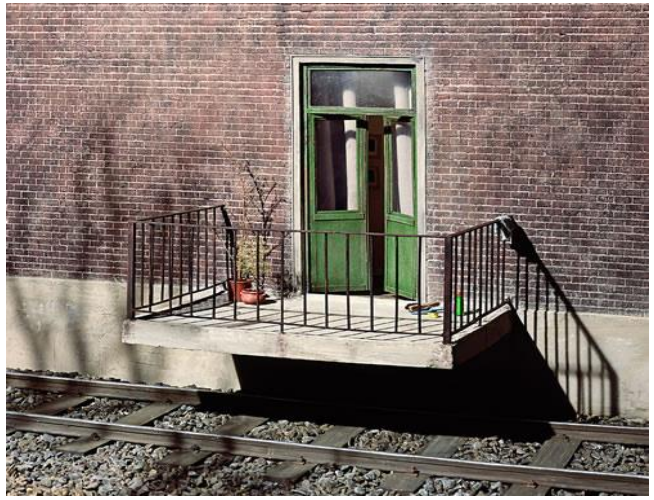
**Unit test**

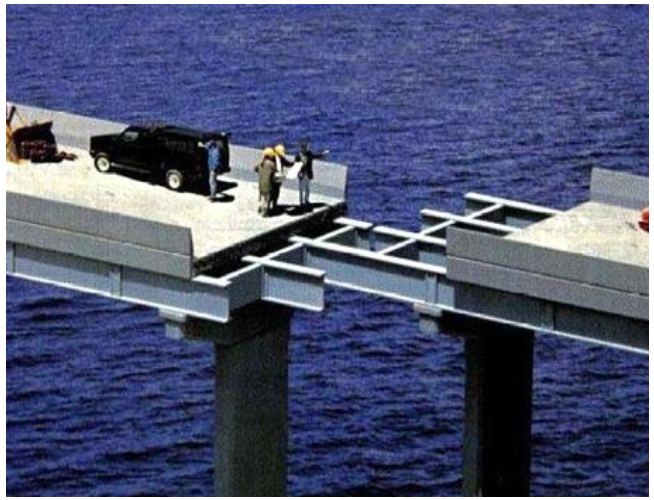
**Exploratory test of user stories**

**Automated regression test**

But it is just a bug....

# A couple of bugs





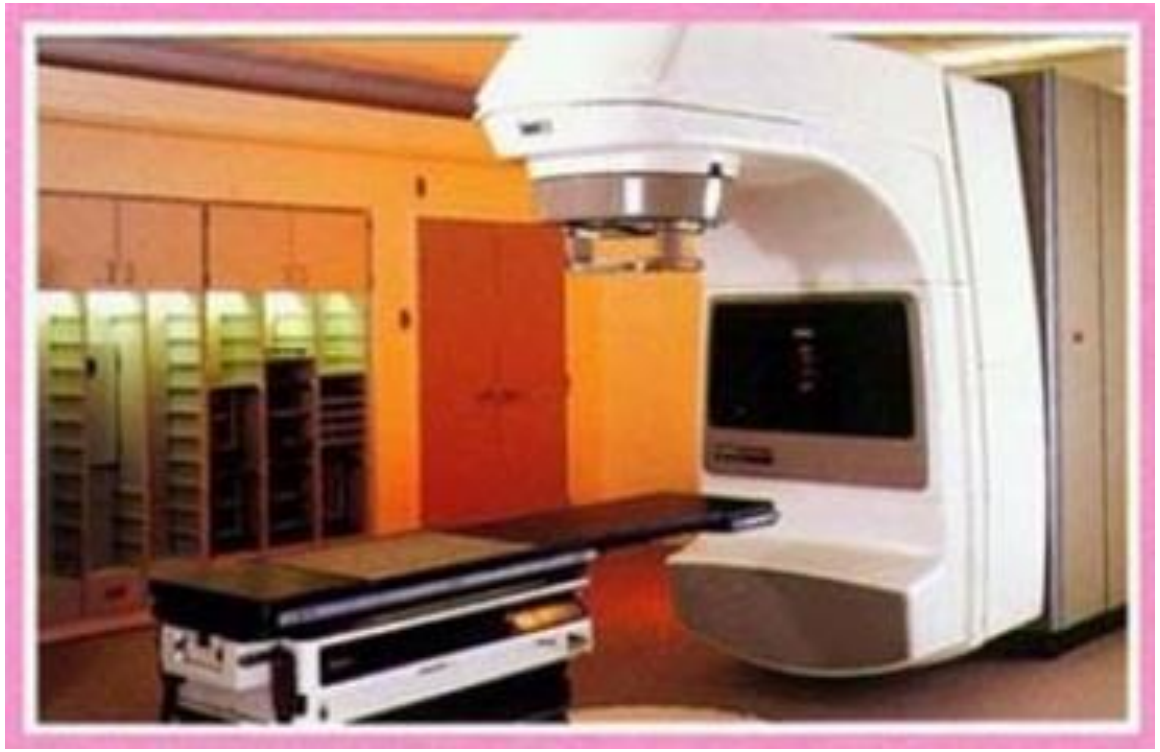
# F-22 Raptor


Sea of Japan
大阪
Osaka
Japan
North
Pacific
Ocean
HAWAII
Hawaii

# Ariane 5


failure due to an error in the software design caused by assertions having been turned off, which in turn caused inadequate protection from integer overflow.

# Therac-25



Massive overdose – 4 dead and 2 severly injoured

# Root Cause Therac 25

AECL did not have the software code independently **reviewed**.

AECL **did not consider the design** of the software during its assessment of how the machine might produce the desired results and what failure modes existed.

The system noticed that something was wrong and halted the X-ray beam, but **merely displayed the word "MALFUNCTION" followed by a number from 1 to 64**. The user manual did not explain or even address the error codes, so the operator pressed the P key to override the warning and proceed anyway.

AECL personnel, as well as machine operators, initially did not believe complaints. This was likely due to overconfidence.

AECL **had never tested the Therac-25 with the combination of software and hardware** until it was assembled at the hospital.

# The agile manifest

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

| | | |
|---|---|---|
| **Individuals and interactions** | ***over*** | **processes and tools** |
| **Working software** | ***over*** | **comprehensive doc.** |
| **Customer collaboration** | ***over*** | **contract negotiation** |
| **Responding to change** | ***over*** | **following a plan** |

That is, while there is value in the items on the right, we value the items on the left more.

**Reference**

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn
Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith
Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin
Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

# The agile manifest - Misunderstood

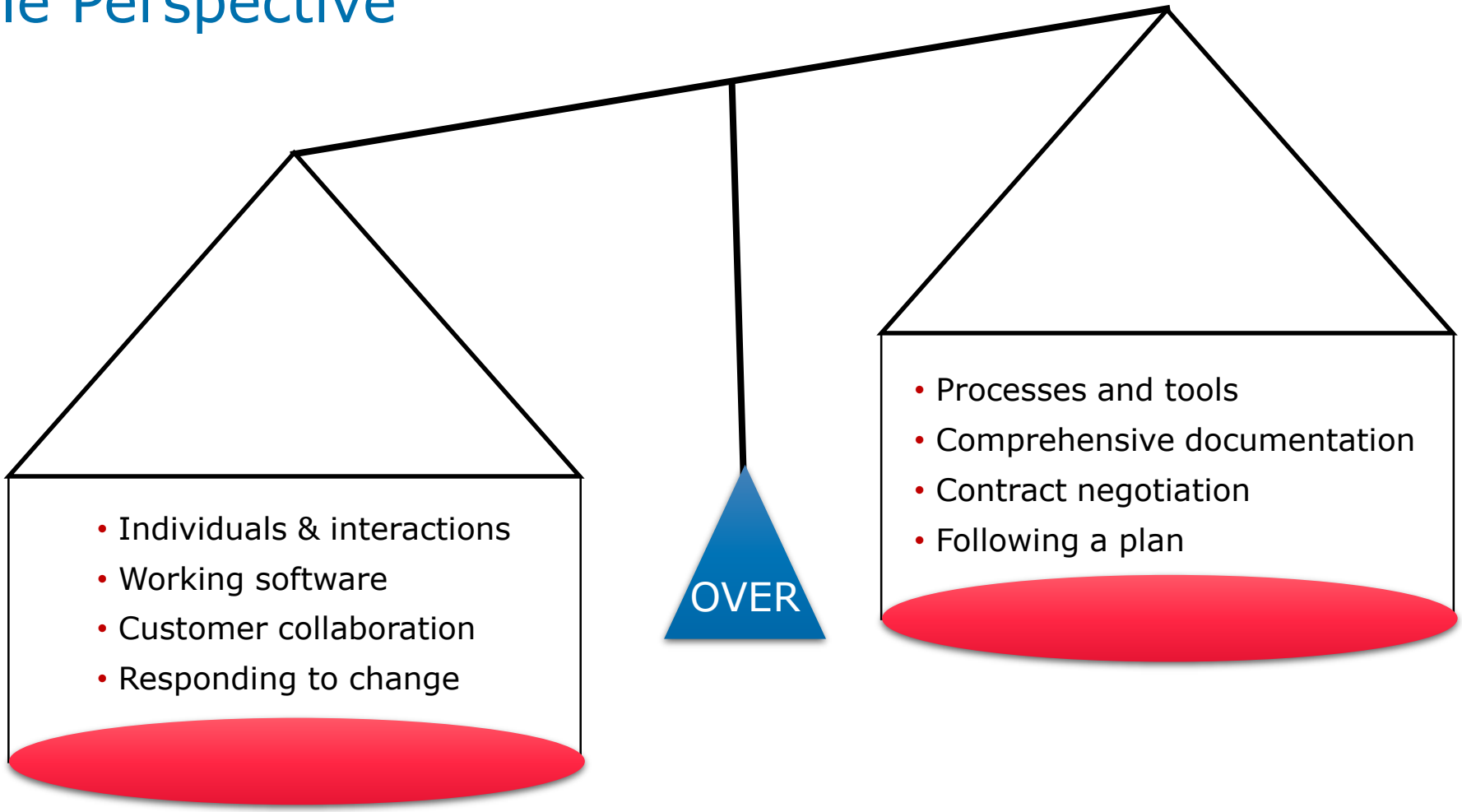
*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

| | | |
|---|---|---|
| **Individuals and interactions** | ***over*** | **processes and tools** |
| **Working software** | ***over*** | **comprehensive doc.** |
| **Customer collaboration** | ***over*** | **contract negotiation** |
| **Responding to change** | ***over*** | **following a plan** |

That is, while there is value in the items on the right, we value the items on the left more.

**Reference**
Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn
Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith
Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin
Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

# The Agile Perspective



# The 12 Agile Principles

1.Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2.Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3.Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4.Business people and developers must work together daily throughout the project.

5.Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6.The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7.Working software is the primary measure of progress. Agile processes promote sustainable development.

8.The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

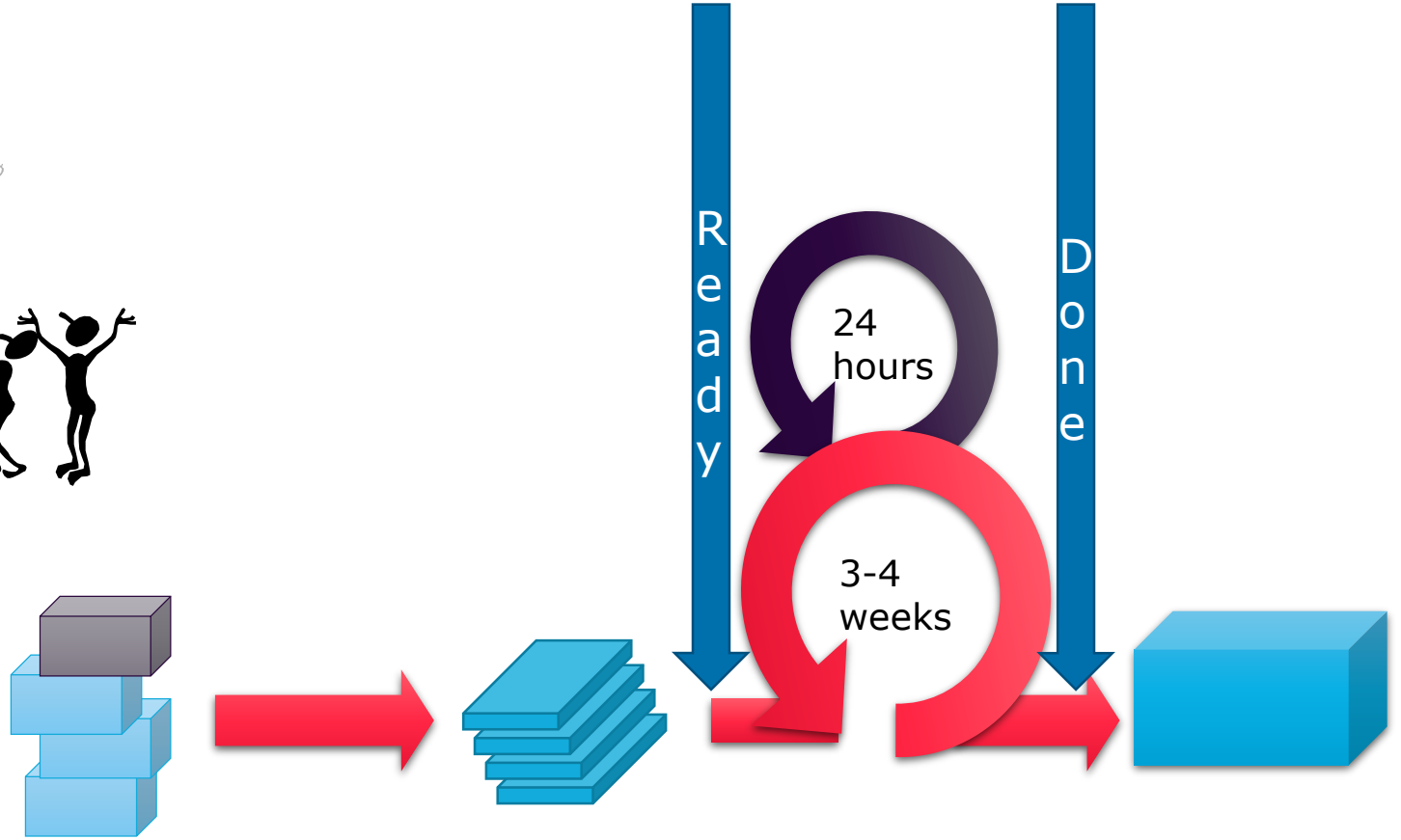
9.Continuous attention to technical excellence and good design enhances agility.

10.Simplicity--the art of maximizing the amount of work not done--is essential.

11.The best architectures, requirements, and designs emerge from self-organizing teams.

12.At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

# Scrum

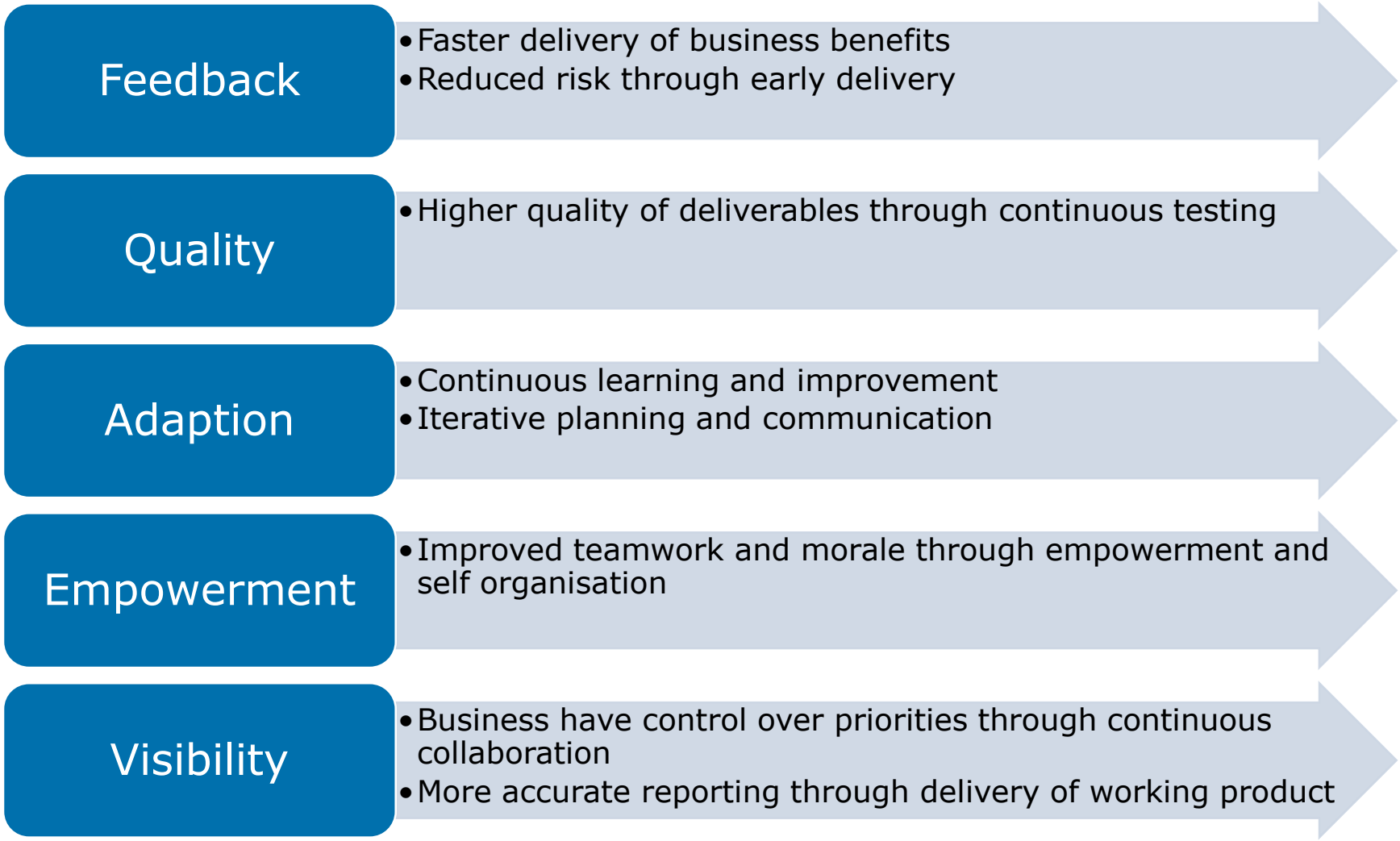



Product Backlog
Prioritized by customer

Sprint Backlog
Broken down by
SCRUM team.

Testet product ready
for release

# What Changes with Agile

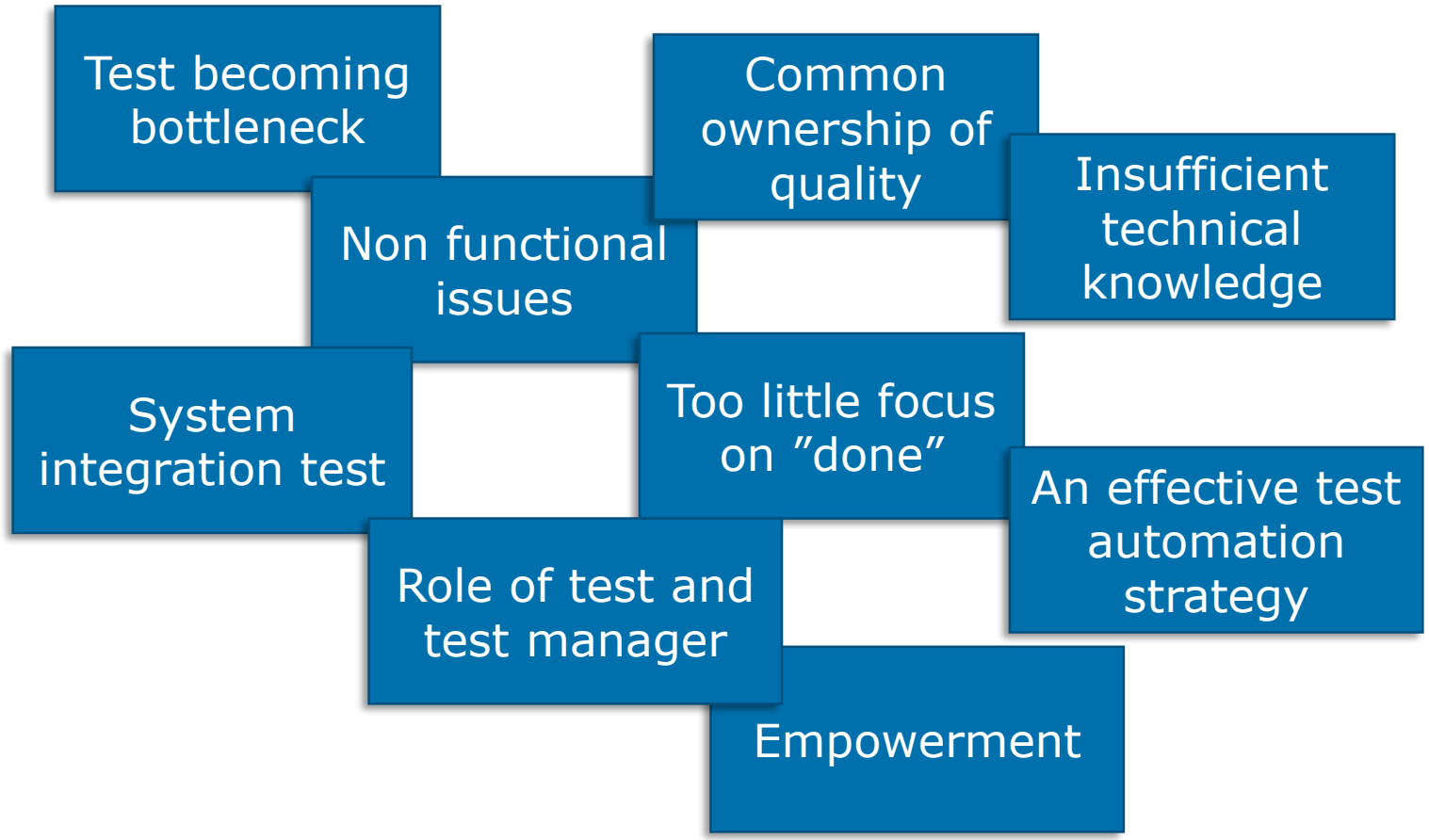


# Traditional versus Agile Projects

| | Plan driven | Agile |
|---|---|---|
| Change | Manage & control it | Change is inevitable – embrace and expect it |
| Planning/test design | Comprehensive upfront plans/test design | Plan/design as you go |
| Documentation | Can be heavy | Minimised - Only as much as necessary |
| Handoffs | Formal entry/exit criteria | Team Collaboration |
| Test Automation | System level built by tool specialists, created after code is 'done' | All levels, built by anyone, an integral part of the project |

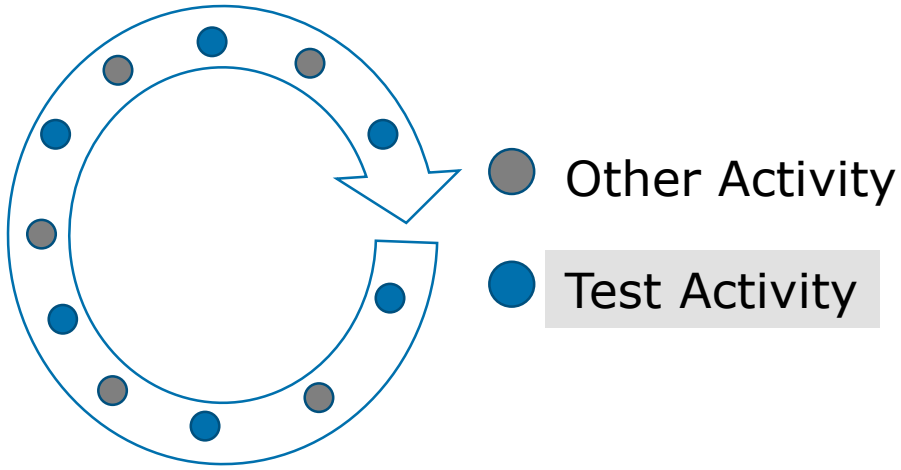
Source: Elizabeth Hendrickson

# Challenges

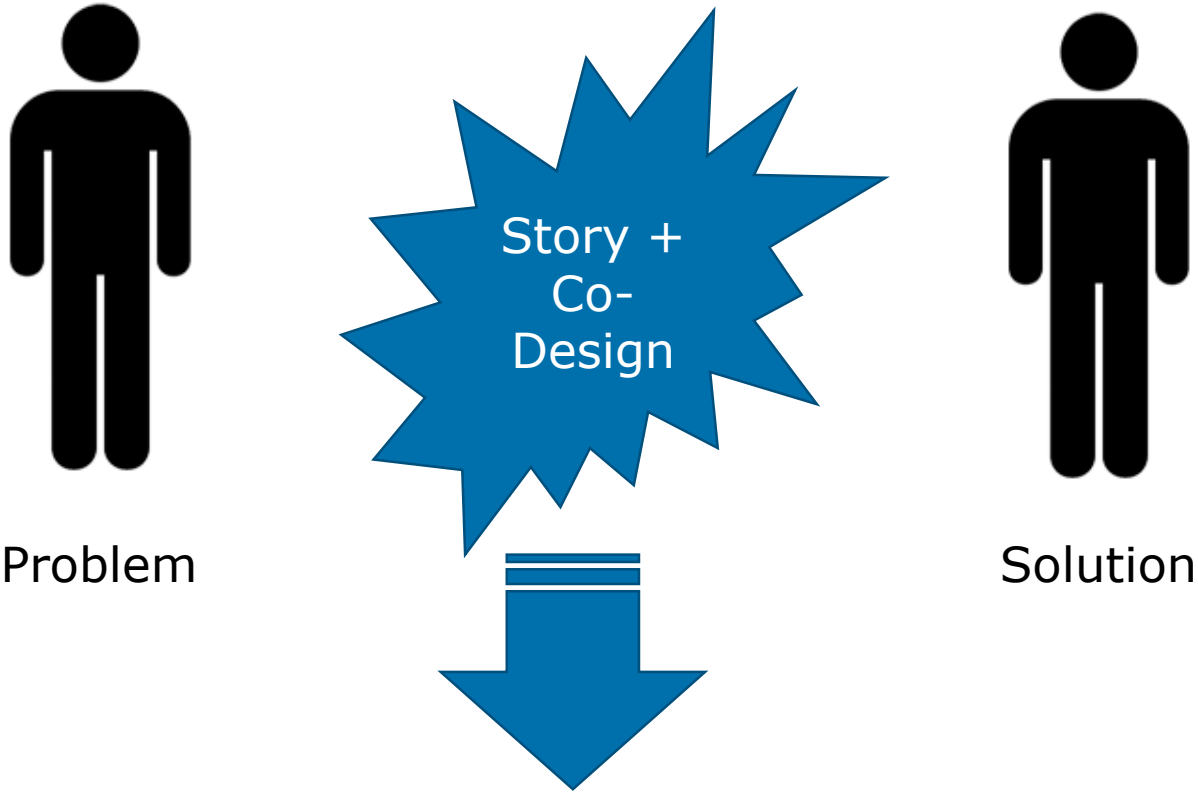

Test becoming bottleneck
Common ownership of quality
Non functional issues
Insufficient technical knowledge
System integration test
Too little focus on ”done”
An effective test automation strategy
Role of test and test manager
Empowerment

# Test in Agile Projects

- Test is done continuously through the iteration, it is NOT a finishing activity
- All team members take part in the test activities – quality is a shared responsibility.



# Define Acceptance Criteria




**Acceptance Criteria**

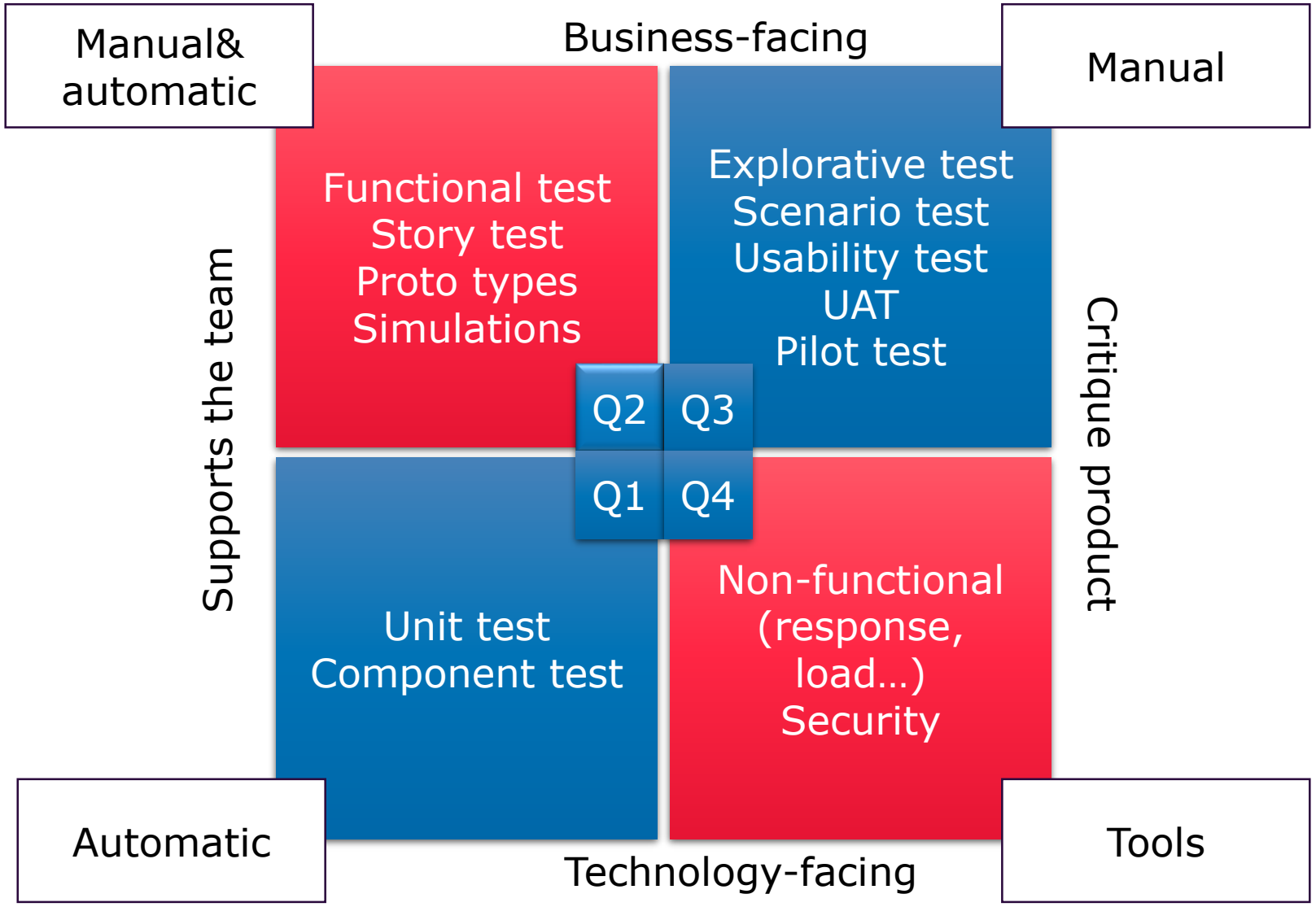
- I can find all users
- I can sort the result according to price
- …

**Details**

- Cookies will be used to store…
- E-mail must be validated….
- …

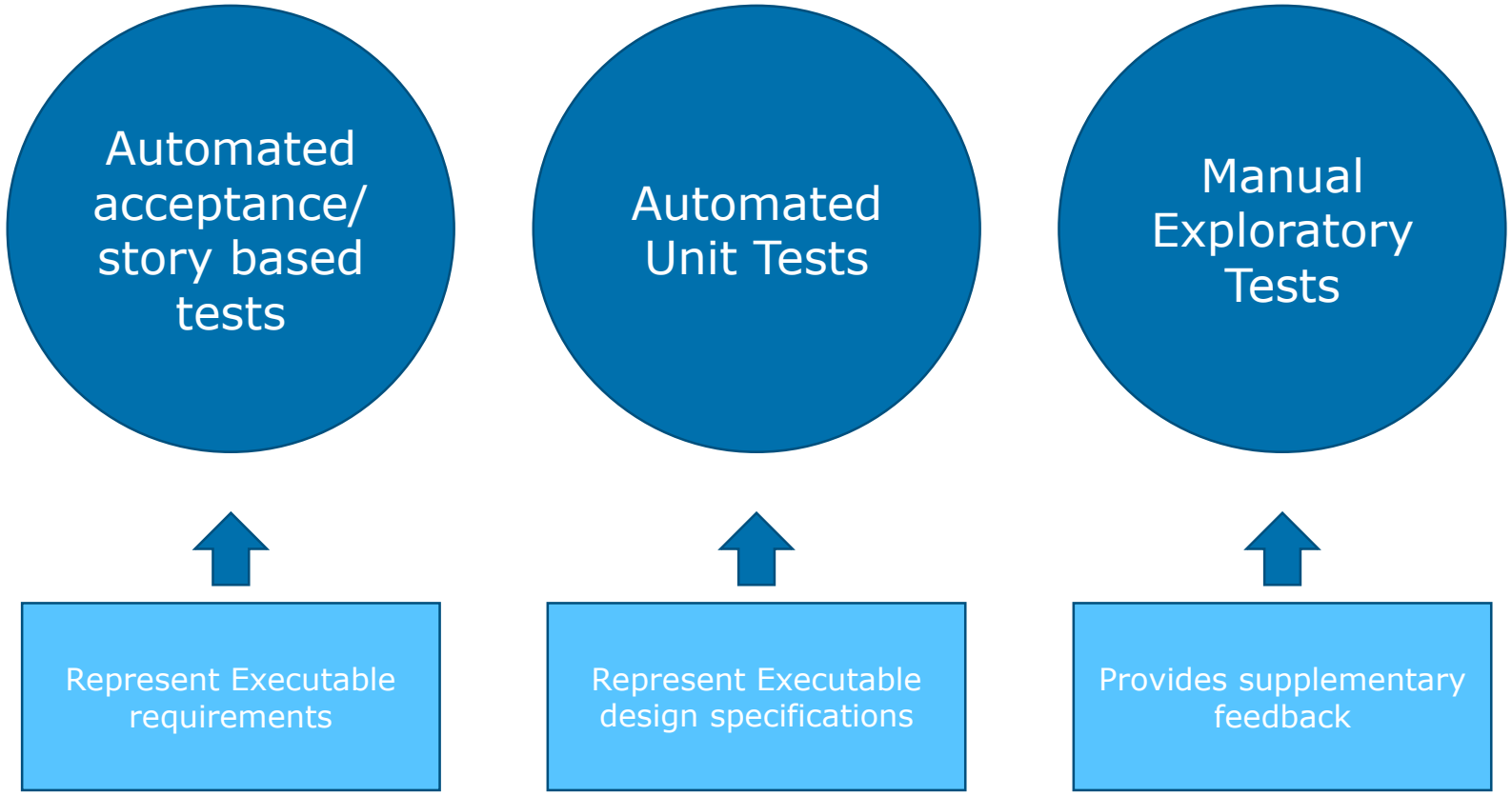
# SMART Requirement

- Specific
- Measurable
- Acceptable
- Relevant
- Timespecific

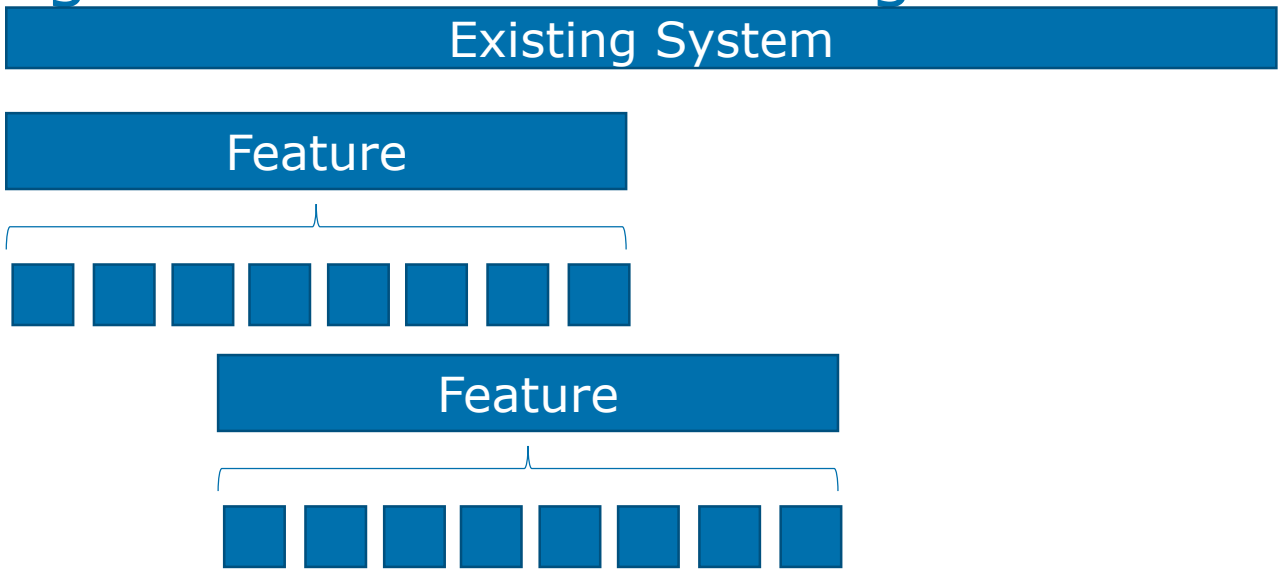


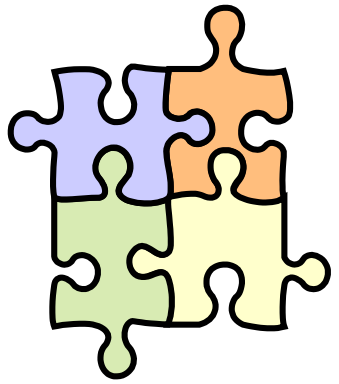

Source: Brian Marick

# Testing Within a Sprint



# Is It Enough? – Remember the Big Picture

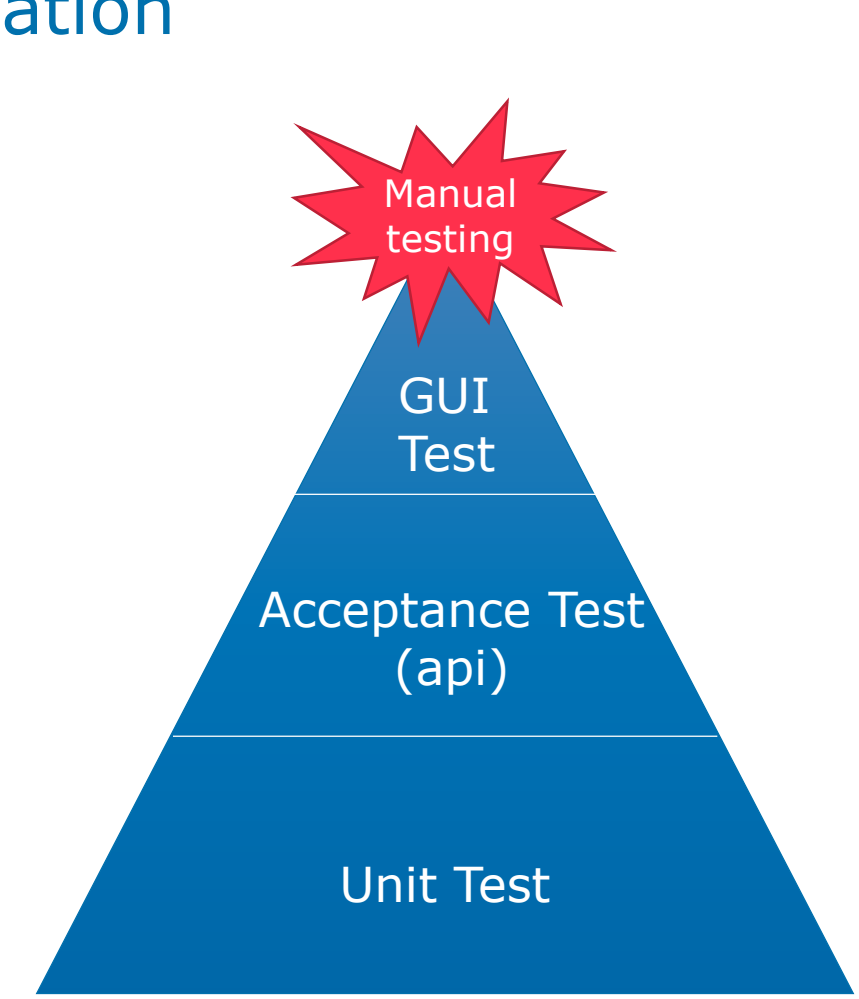


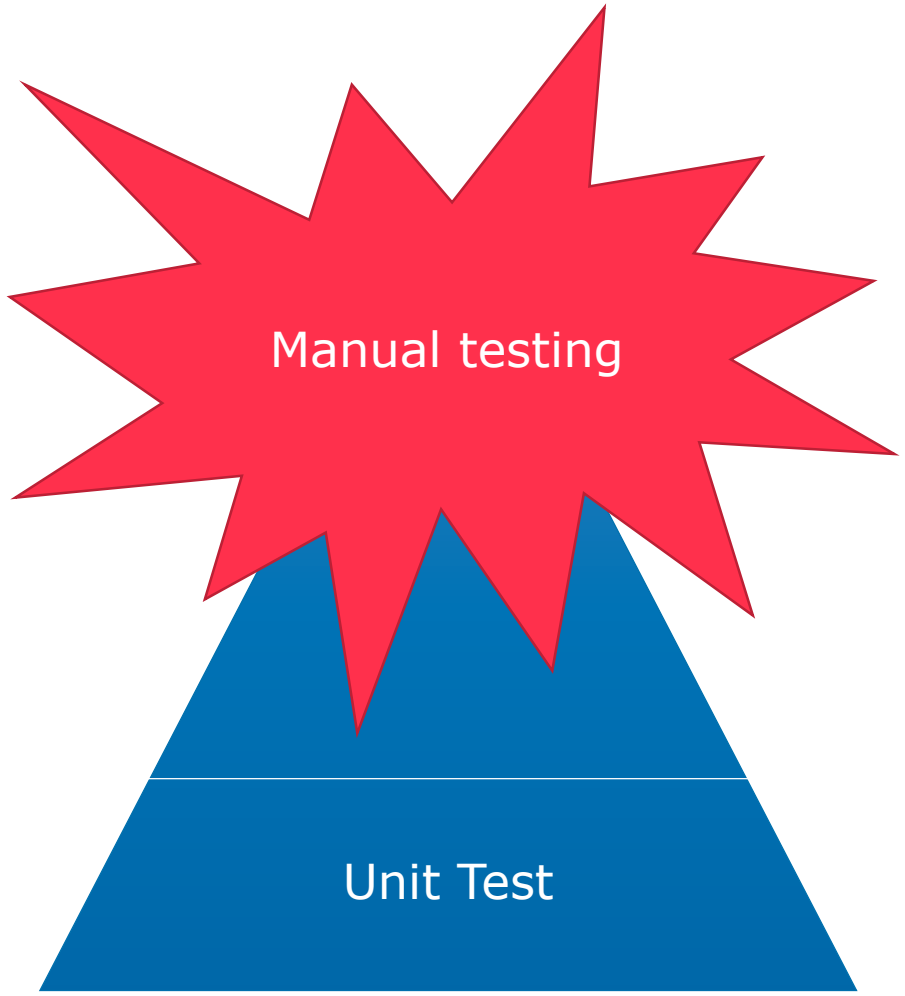



**Think about the testing quarants**

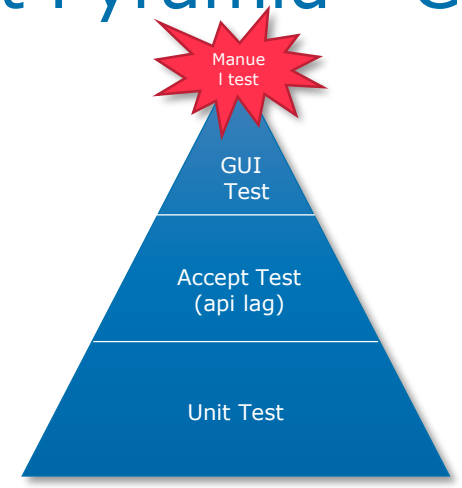
| | | | |
|---|---|---|---|
| Unit test<br>Component test | Functional test<br>Story test<br>Proto types<br>Simulations | Explorative test<br>Scenario test<br>Usability test<br>UAT<br>Pilot test | Non-functional<br>(response,<br>load…)<br>Security |

# Automation

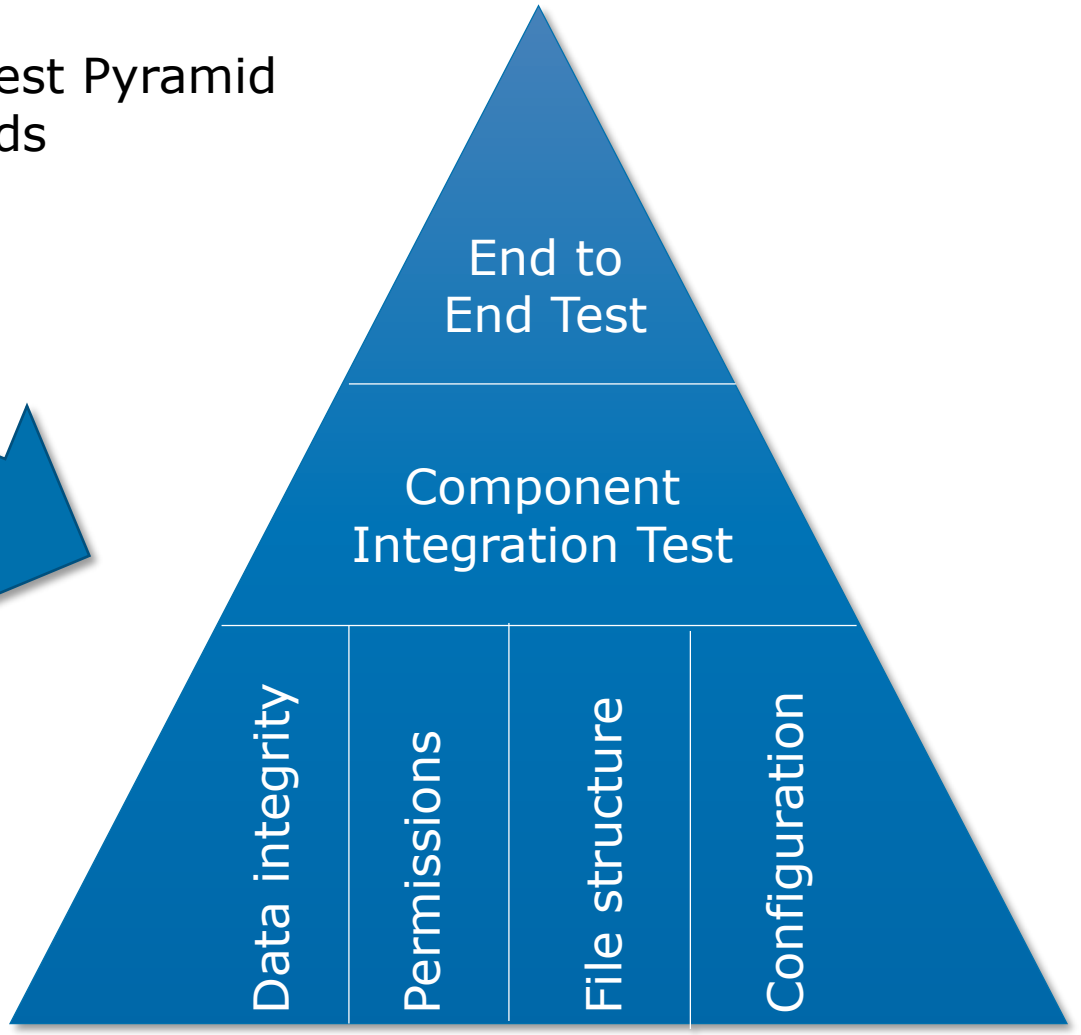

Manual testing
GUI Test
Acceptance Test (api)
Unit Test

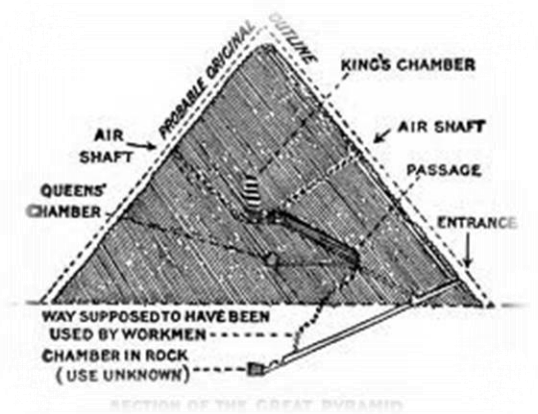

Manual testing
Unit Test

# The Test Pyramid - Context is Everything…

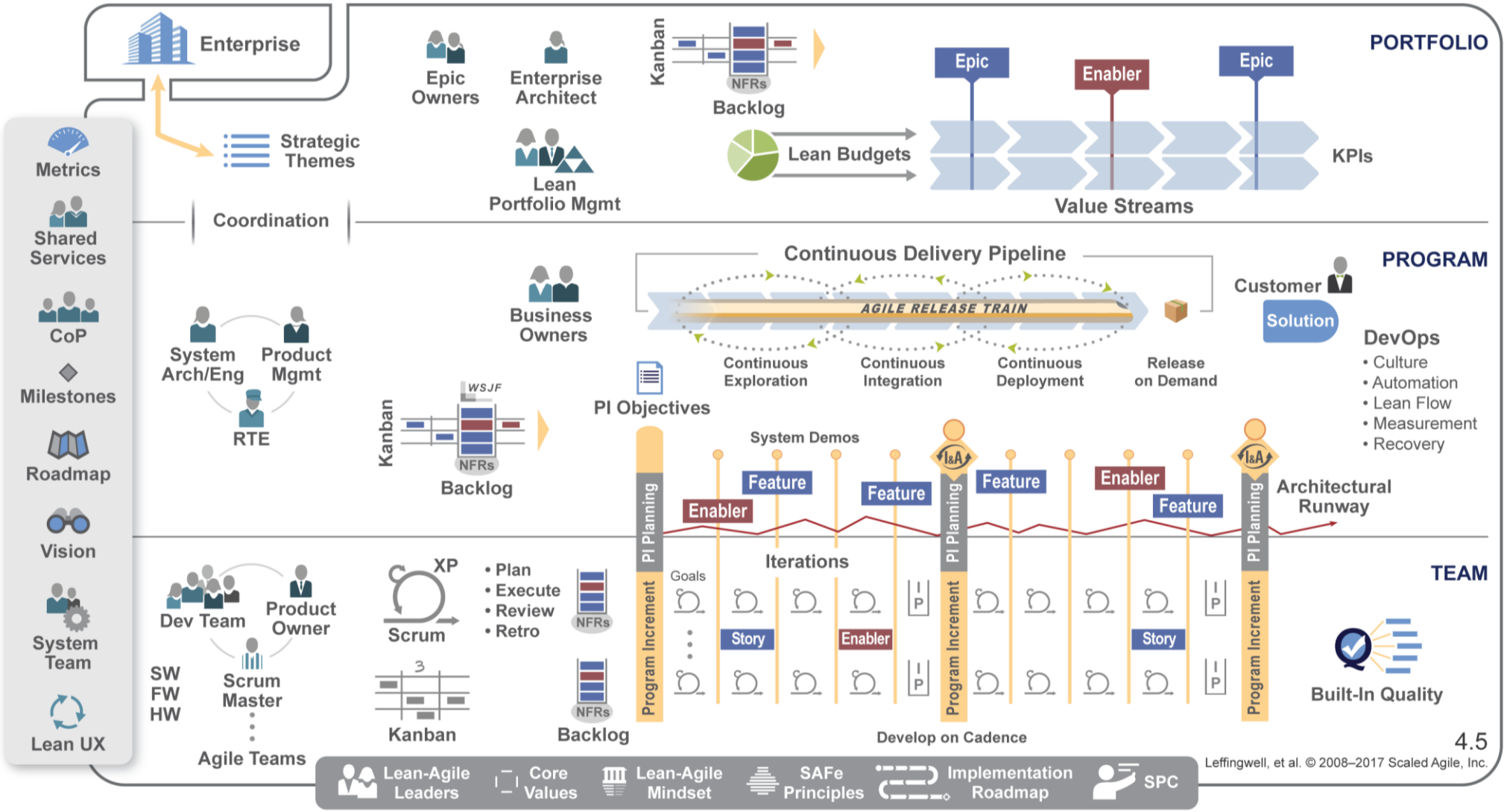



Adapt the Test Pyramid
to your needs



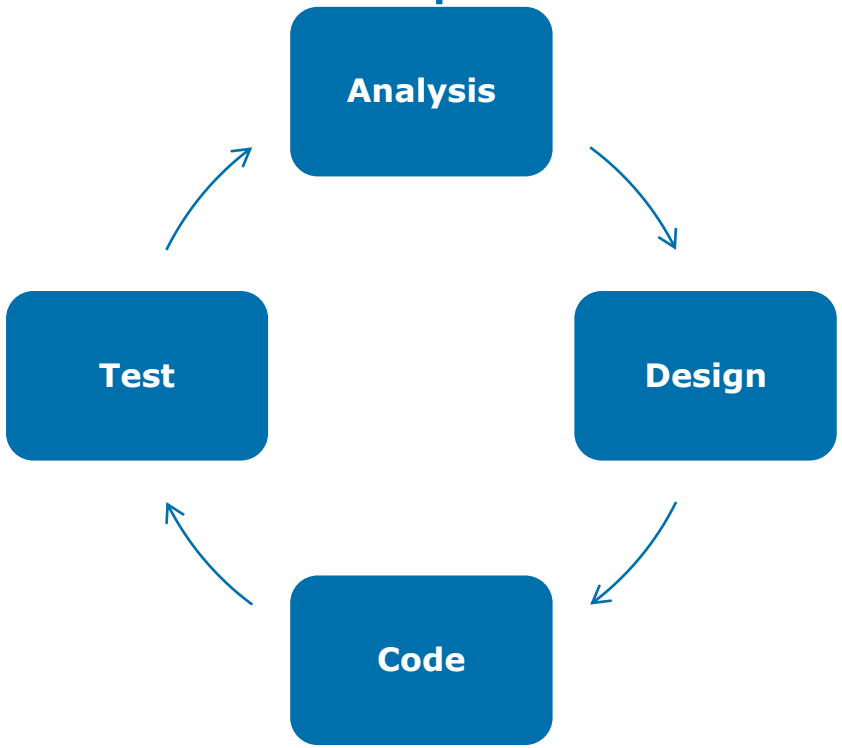



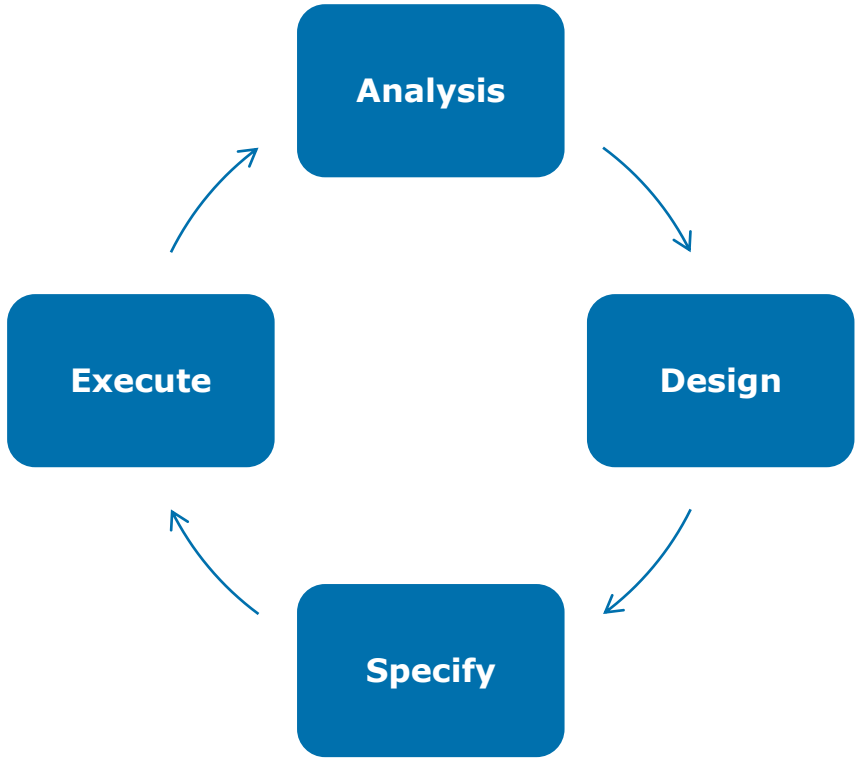

Source: Lisa Crispin



# What is Structured Testing

# A Cycle for Development... And Testing



# AND THAT GOES FOR ALL TEST LEVELS

Unit integration test

Unit test

System integration test

System test

Acceptance test

Non functional test types

Review

# A Pairwise Example

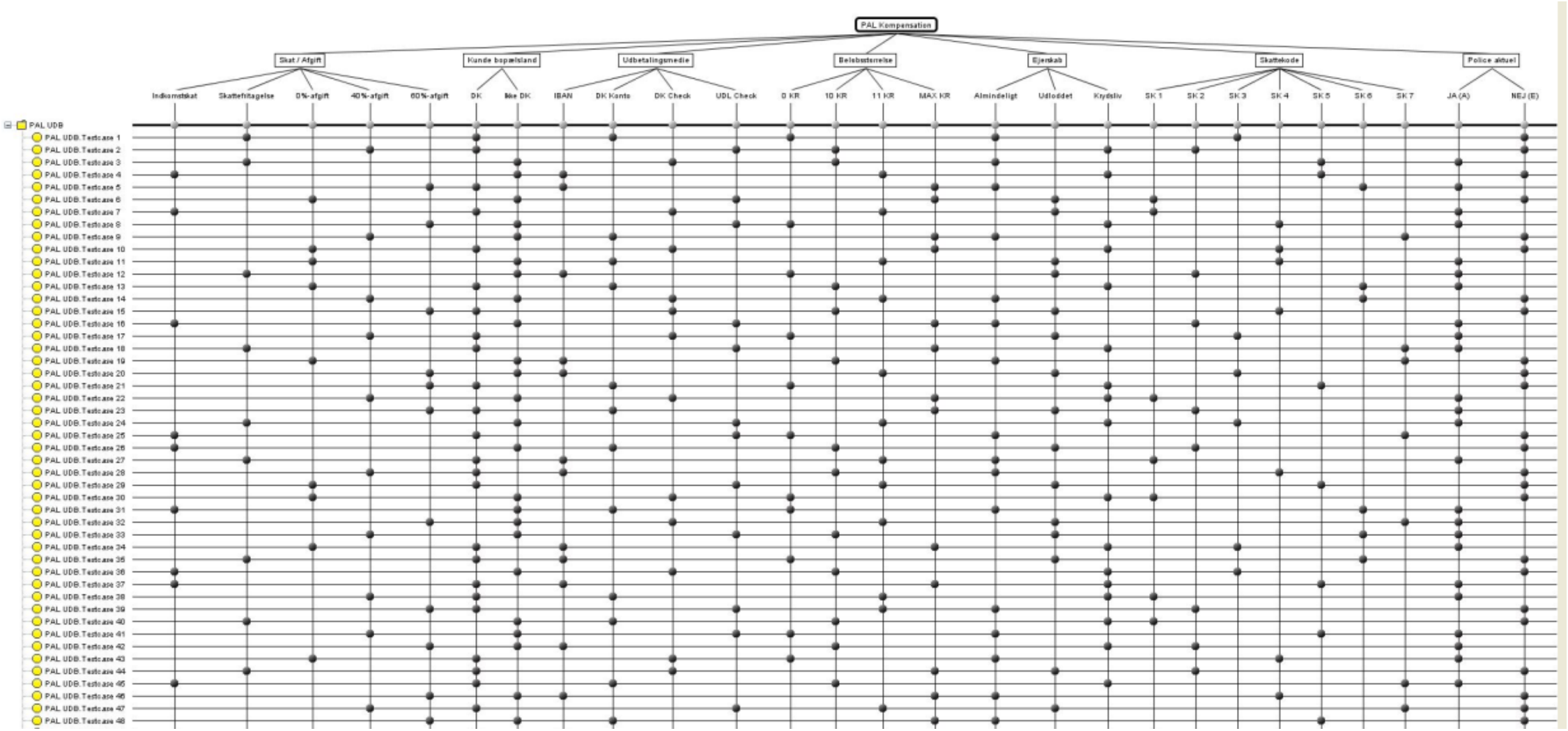
- Tax / Tax
  - Income Tax, Tax exemption, 0% tax, 40% tax, 60% tax
- Customer's country of residence
  - Denmark, Not Denmark
- Payment media
  - IBAN, DK account, DK-Check, UDL-Check
- Amount
  - 0 kr., 10 kr., 11 kr., Max kr.
- Ownership
  - Commonly, Distributed, Krydsliv
- Tax code
  - SK1, SK2, SK3, SK4, SK5, SK6, SK7
- Actual policy
  - Yes (A), No (E)

Number of combination

5 * 2 * 4 * 4 * 3 * 7 * 2 = 6.720

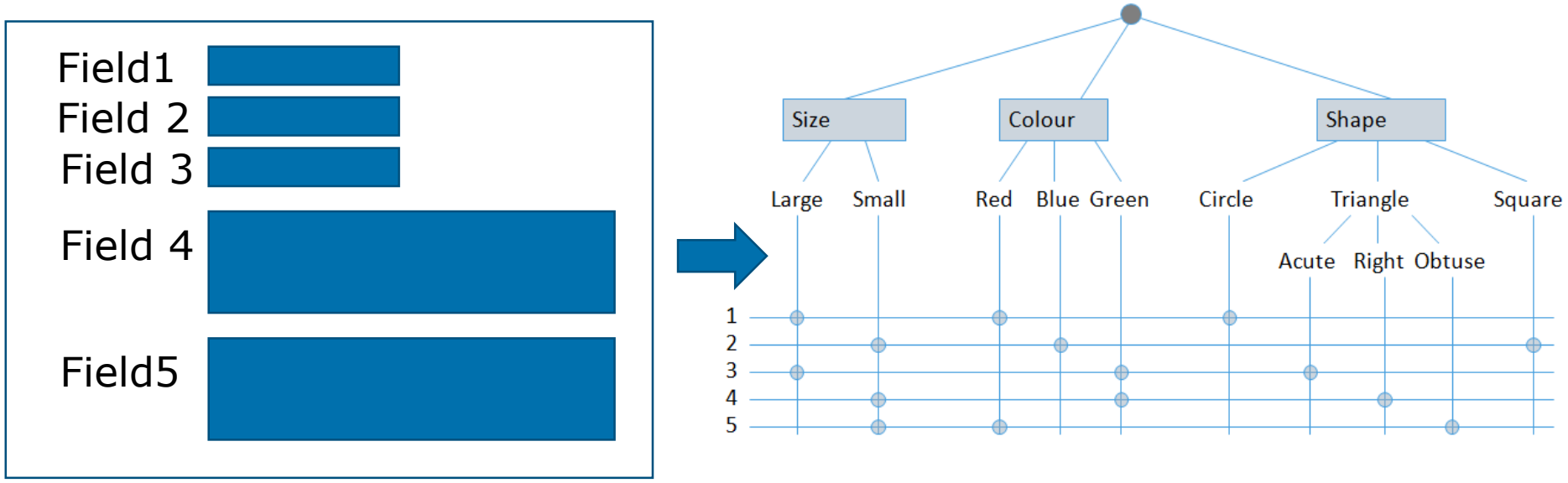
With pair wise
38 test cases (0,6%)

With triple-wise
178 test cases (2,6%)

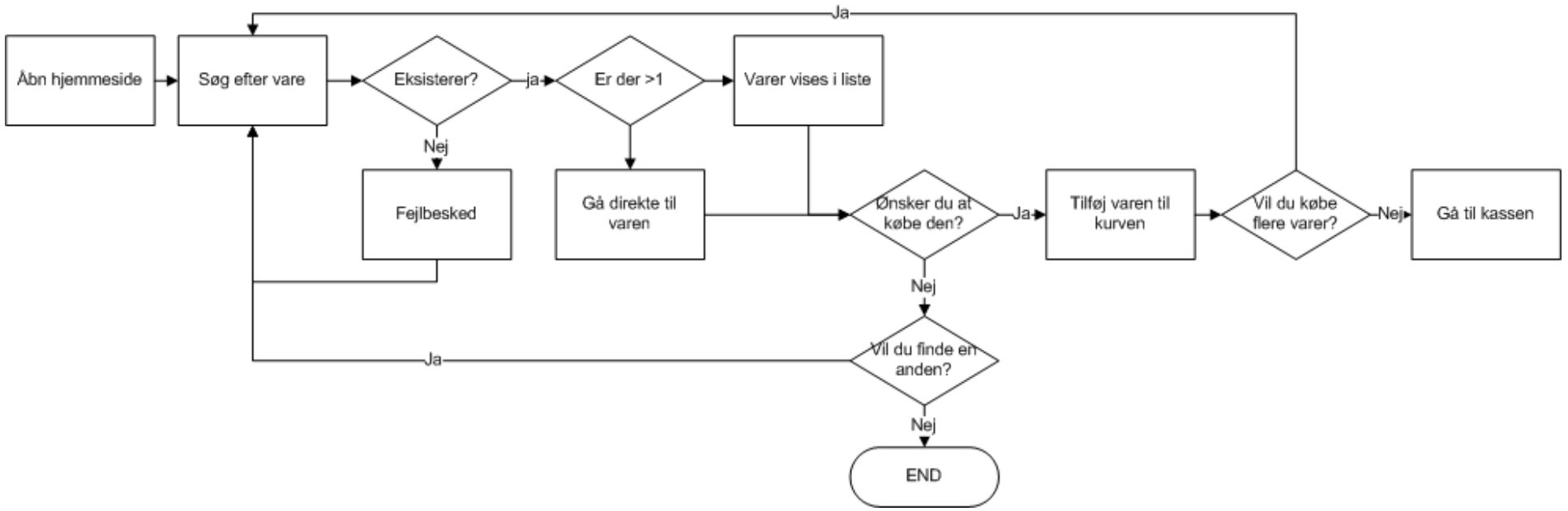


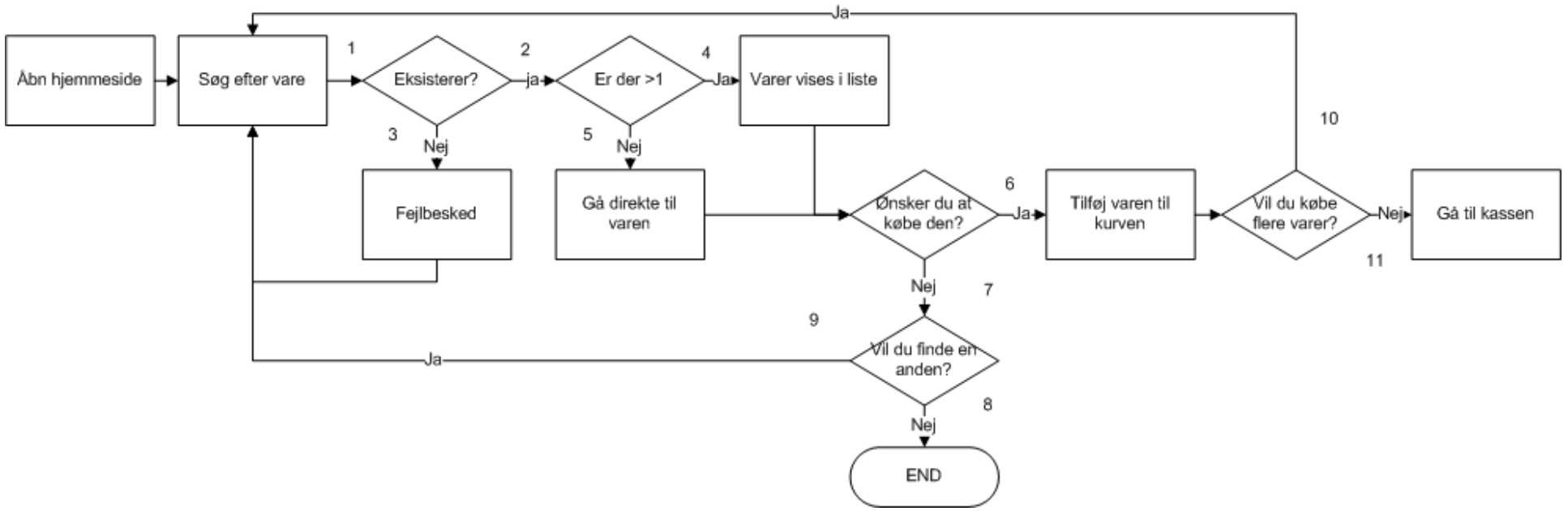

**Classification tree and triple-wise**

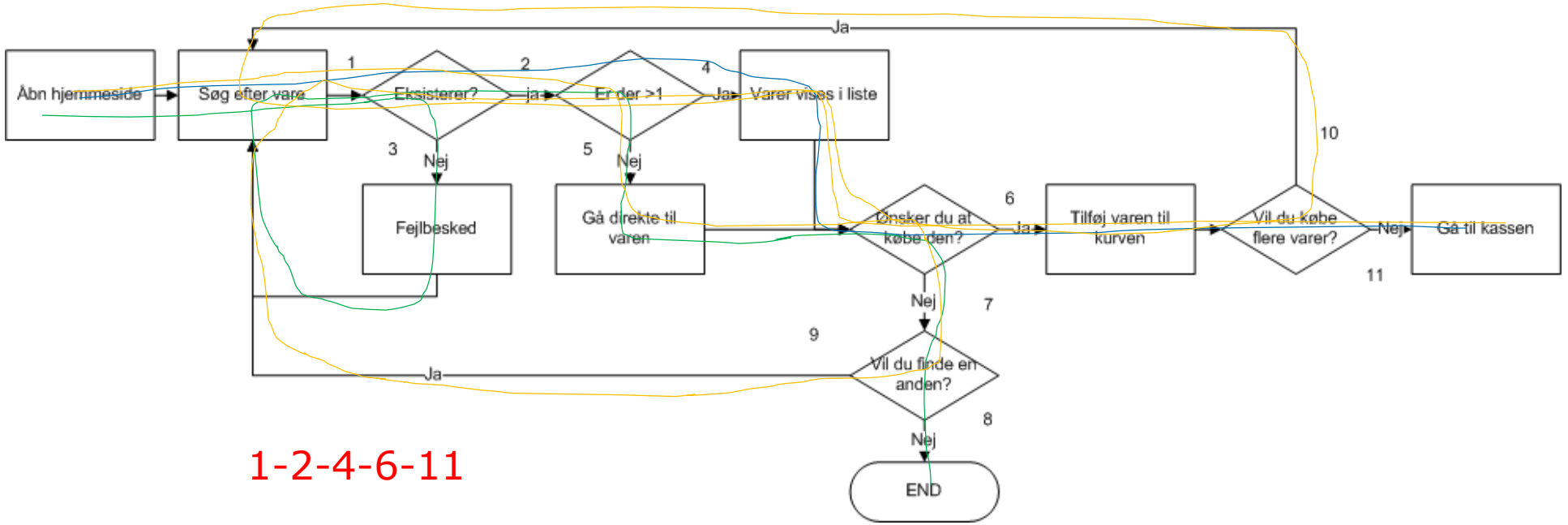
# Equivalence Partitioning and Classification Tree



# Workflows with the Users Glasses




Åbn hjemmeside
Søg efter vare
Eksisterer?
ja
Er der >1
Ja
Varer vises i liste
Nej
Fejlbesked
Nej
Gå direkte til varen
Ønsker du at købe den?
Ja
Tilføj varen til kurven
Vil du købe flere varer?
Nej
Gå til kassen
Nej
Vil du finde en anden?
Ja
Nej
END
Ja
1
2
3
4
5
6
7
8
9
10
11



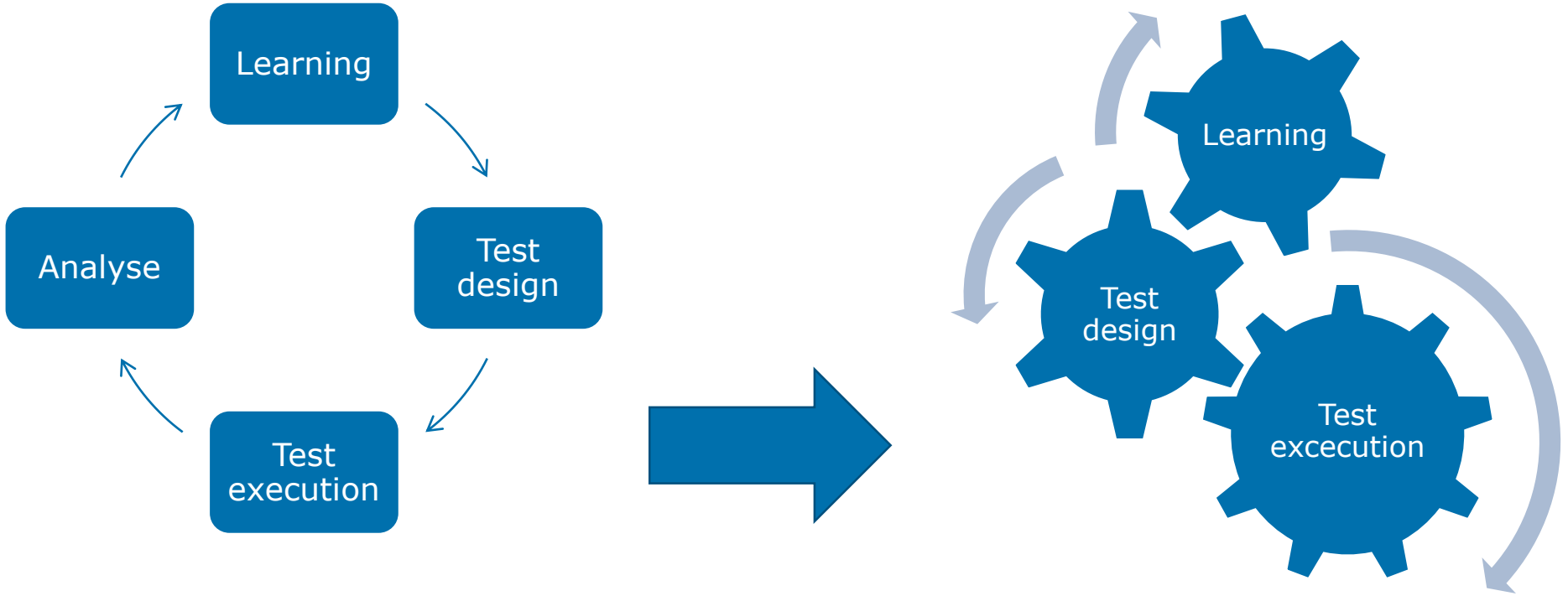

1-2-4-6-11

1-3-1-2-5-7-8

1-2-4-7-9-1-2-5-6-10-1-24-6-11

# Exploratory Testing and Agile Testing - ET

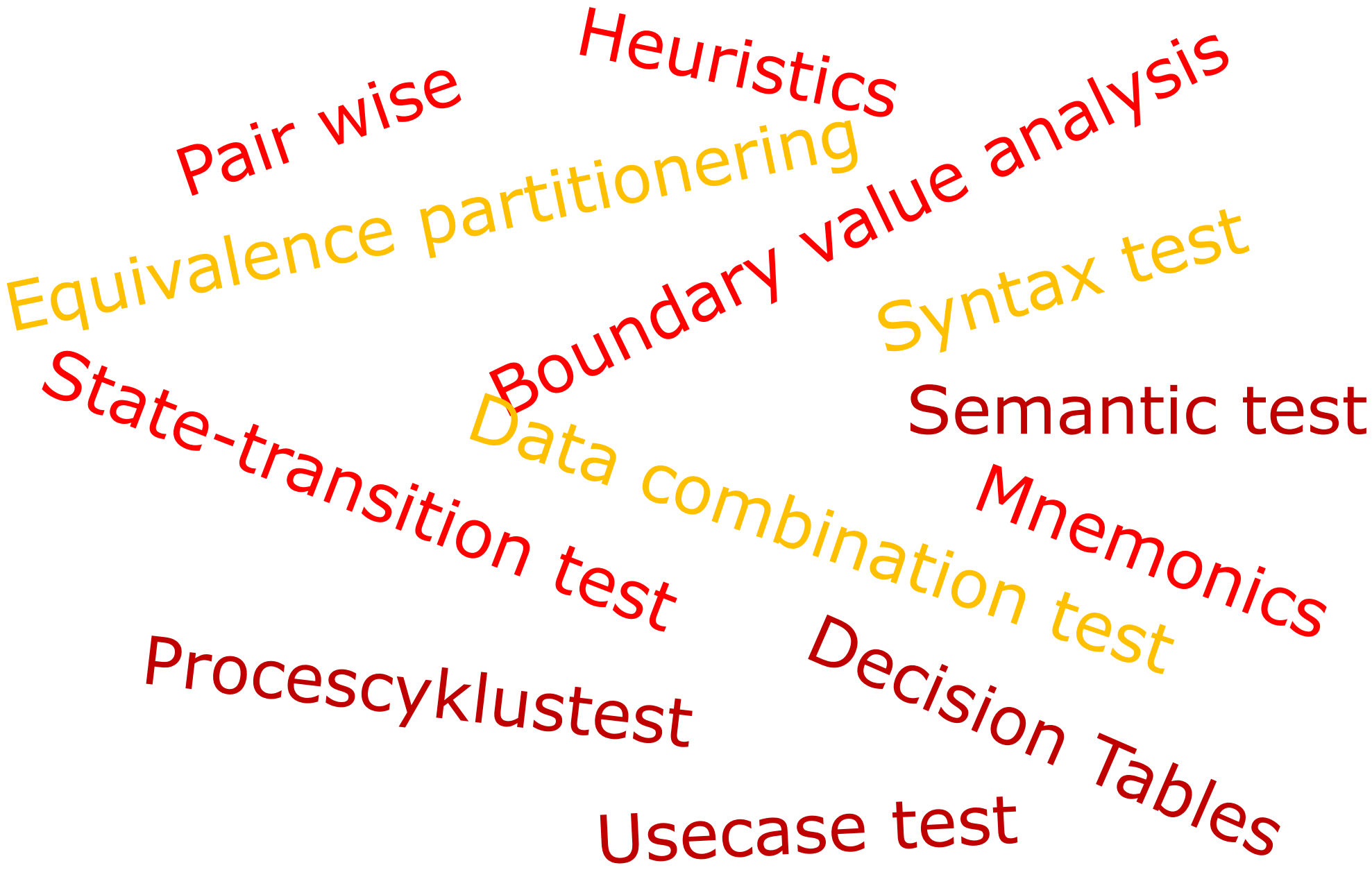
Exploratory testing is simultaneous learning, test design, and test execution.




Cycle

Continous

# Exploratory Testing and Agile Testing -Test charter


Pair wise
Heuristics
Equivalence partitionering
Boundary value analysis
Syntax test
Semantic test
State-transition test
Data combination test
Mnemonics
Procescyklustest
Decision Tables
Usecase test

# All in All – Who Tests?

Everybody!!

- Business. With the focus that what is developed can be used ”in the real world”.
- The tester. With the focus that specification and requirements are implemented – with focus on bughunting.
- The Developer. With the focus that he/she has build the software right

EVERYBODY REVIEWS – that is also testing... Just static