# Testing – Overview

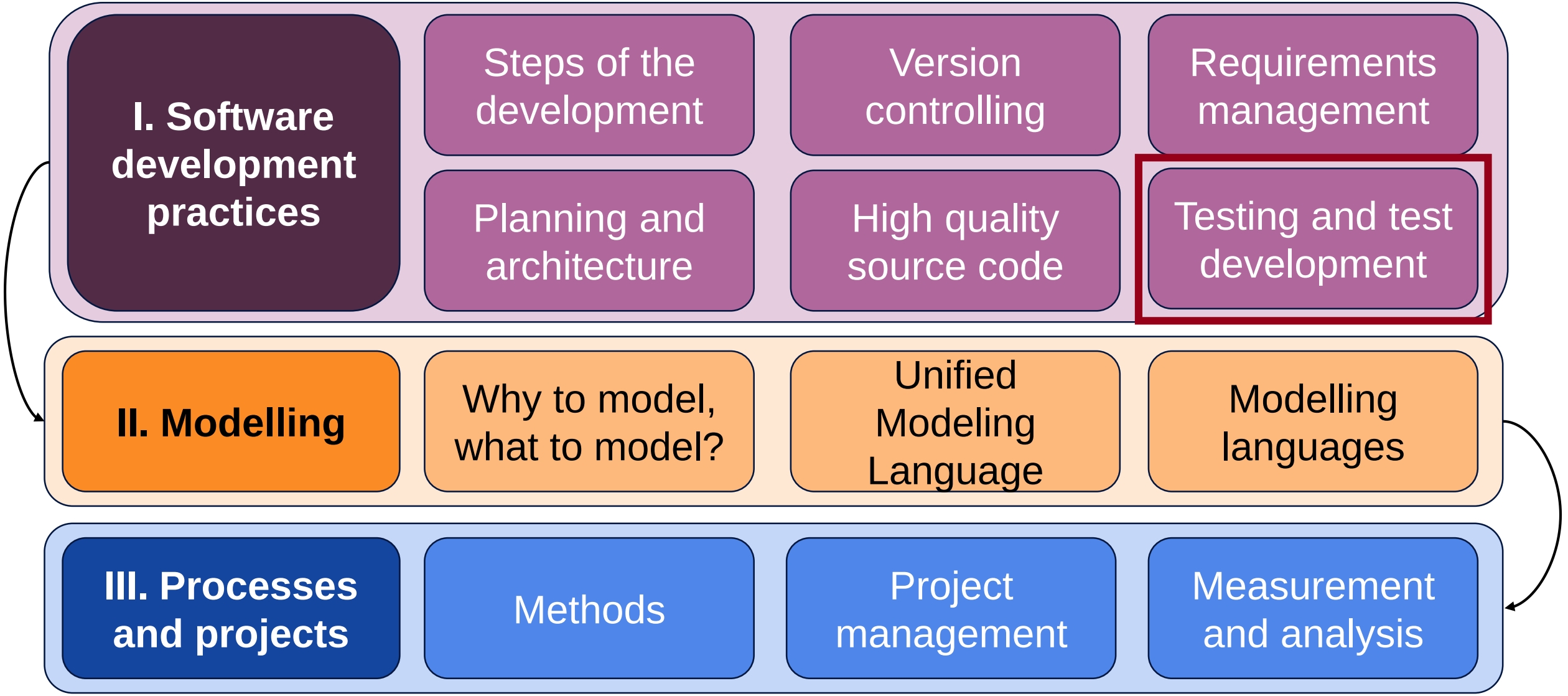
HUSZERL Gábor
huszerl@mit.bme.hu

MŰEGYETEM 1782
BME VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
Méréstechnika és Információs Rendszerek Tanszék
ftsrg Critical Systems Research Group

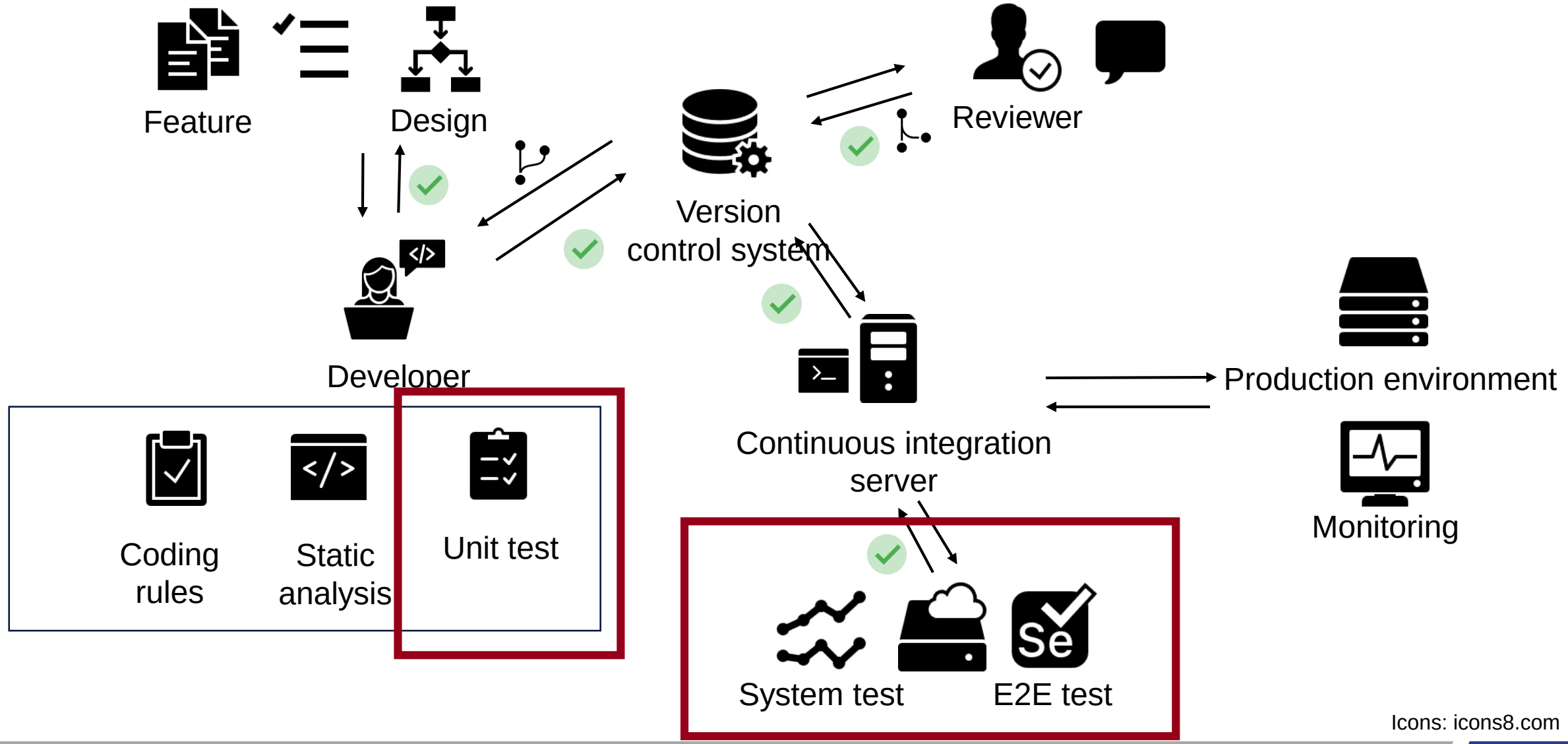
# Learning Outcomes

- At the end of the lecture the students are expected to be able to
- (K1) recall the definition and objectives of software testing,
- (K2) summarise the typical software testing process,
- (K2) distinguish between different levels and techniques of testing.

# Further Topics of the Subject



# Typical Development Workflow

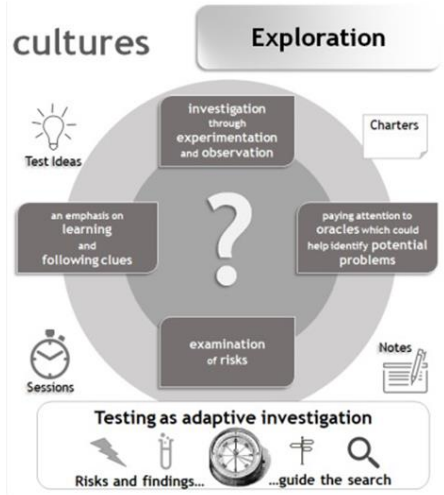



Icons: icons8.com

# Exercise: How would you test it?


Source: [ECB]

# Testing




## Definition and Goals

# Examination Methods (Reminder)

## Static

- **What**: any products (documentation, model, code)
- **How**: without execution
- **Example**: review, static analysis

## Dynamic

- **What**: executable products (code, model, …)
- **How**: executing it, running it
- **Example**: simulation, testing, …

# Testing – Definition (1)

> „Testing is an activity performed for evaluating product quality, and for improving it, by identifying defects and problems.”
>
> Source: IEEE, „Software Engineering Body of Knowledge” (SWEBOK) 2004
> URL: http://www.computer.org/portal/web/swebok/

# Testing – Definition (2)

> „An activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component.”
>
> Source: IEEE, "Systems and software engineering — Vocabulary," ISO/IEC/IEEE Standard 24765, 2010

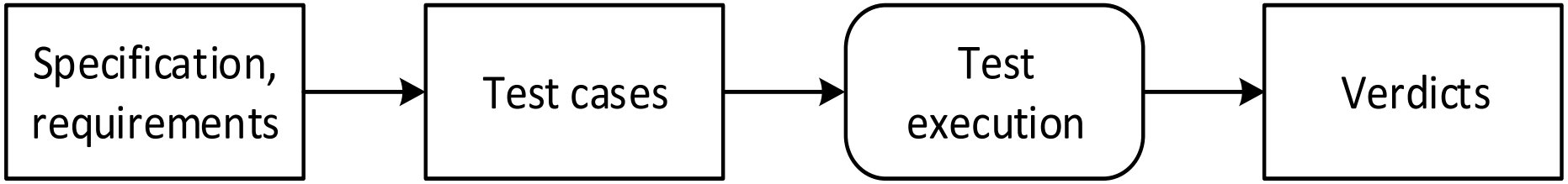
# Testing – Definition (3)

> „The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products
>
> - to determine that they satisfy specified requirements,
> - to demonstrate that they are fit for purpose and
> - to detect defects.”
>
> Source: International Software Testing Qualifications Board (ISTQB), URL: http://istqb.org/

# Testing – Definition (4)

> “Testing is the process of evaluating a product by learning about it through exploration and experimentation, which includes: questioning, study, modeling, observation and inference, output checking, etc.”
>
> Source: James Bach, Michael Bolton. Exploratory Testing 3.0

# Basic Concepts of Testing




- SUT: system under test
- Test case
  - a set of inputs, implementation conditions and expected results developed for a specific purpose
- Test suite
- Test oracle
  - A principle or method that helps the tester decide whether the test was successful
- Verdict: pass / fail / error / inconclusive…

# About Exhaustive (100%) Testing

- Let us consider a function with 2 integer input parameters
  - Number of test cases for exhaustive testing :
  - Duration of the test (assumed 1 sec / test case):
  - Power consumption in $5*10^{15}$ hours (assumed 100W):
  - Price of electricity for $5*10^{14}$ kWh (ass. 20 cent/kWh):

- GDP of EU: $14,5*10^{12}$ (in 2021) ~ $10^{14}$ Euro in 6,9 years
- GDP of USA: $26,9*10^{12}$ (in 2021) ~ $10^{14}$ Euro in 3,7 years
- GDP of world: $96,51*10^{12}$ (in 2021) ~ $10^{14}$ Euro in 1,04 years

# Tasks to Solve

## Test selection

- What input values and data to use?

## Oracle problem

- Where to find a reliable oracle?

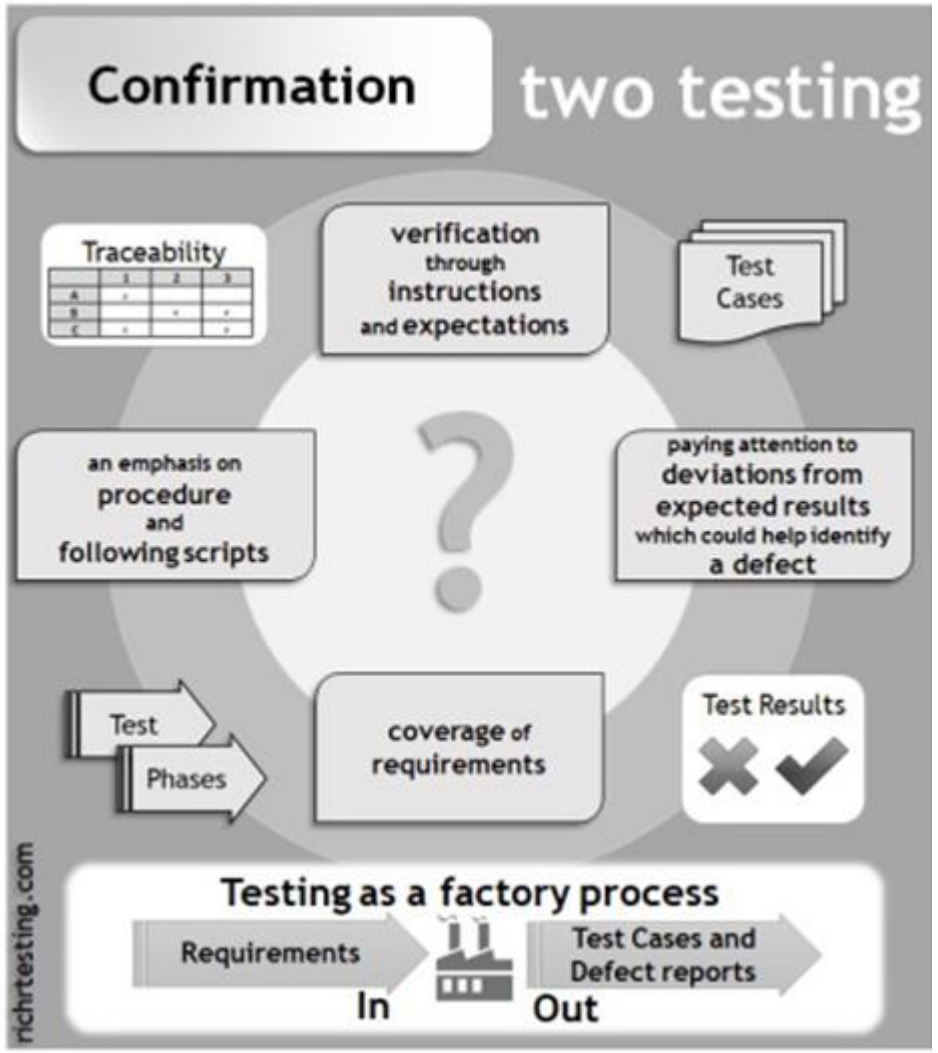
## Exit criteria

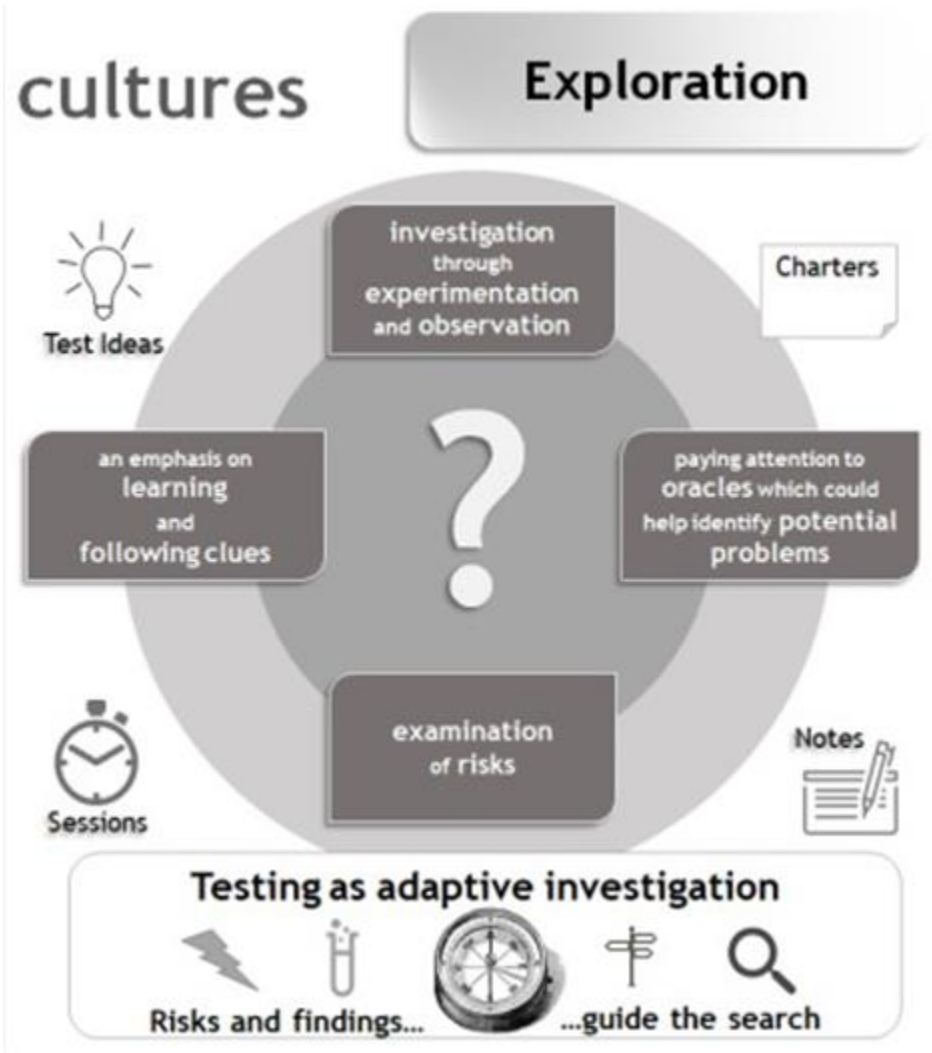
- When to stop testing?

# Possible purposes of testing


and many more …

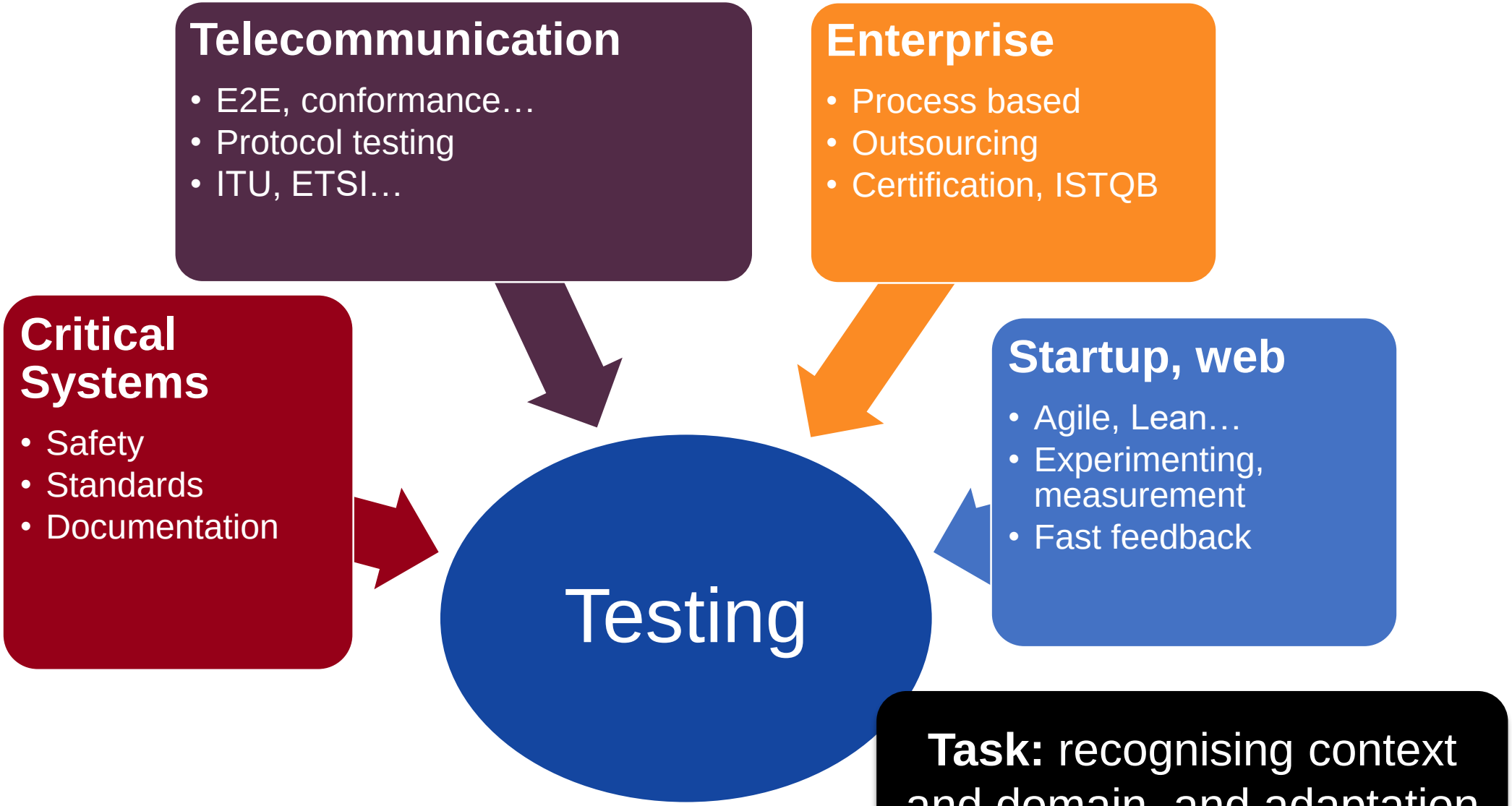
# Confirmation and Exploration



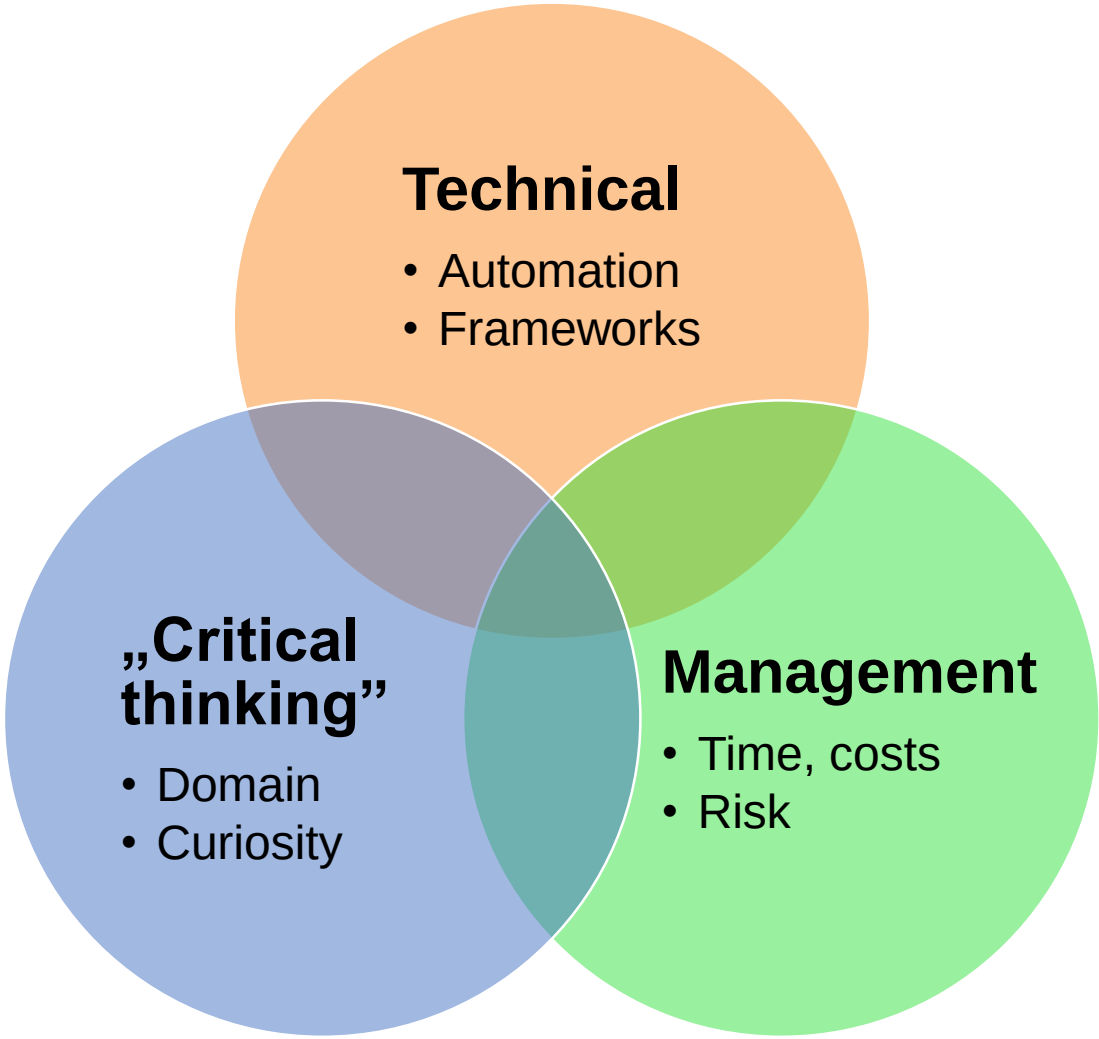

Source: Rich Rogers. “Confirmation and Exploration”

# Testing (also) Depends on the Context!

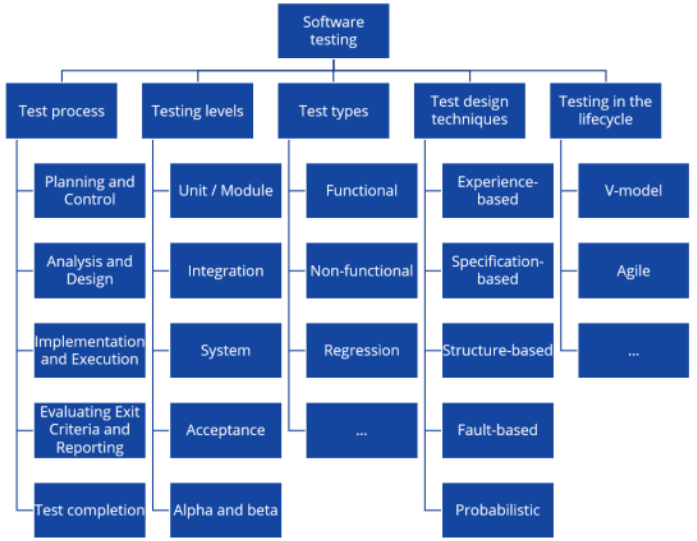



**Task:** recognising context and domain, and adaptation

# View Points on Testing



# Testing Process

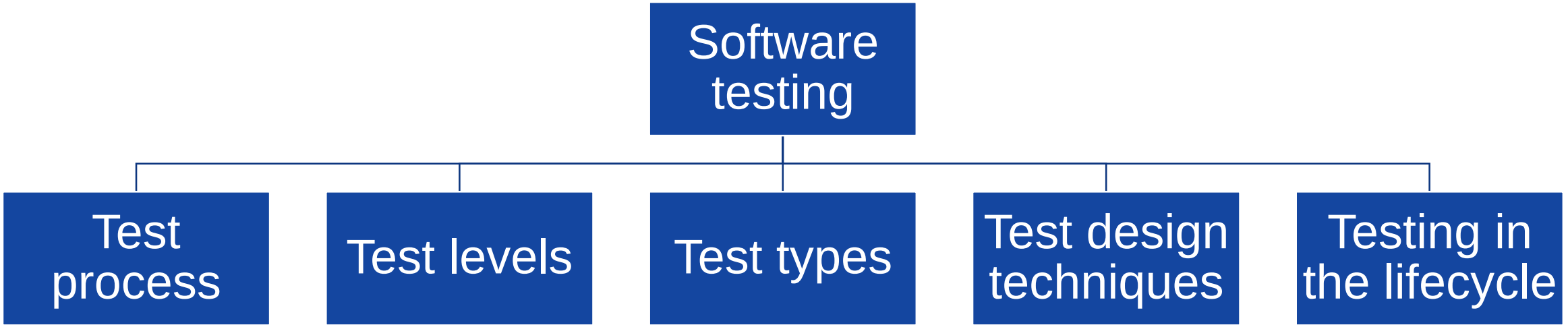


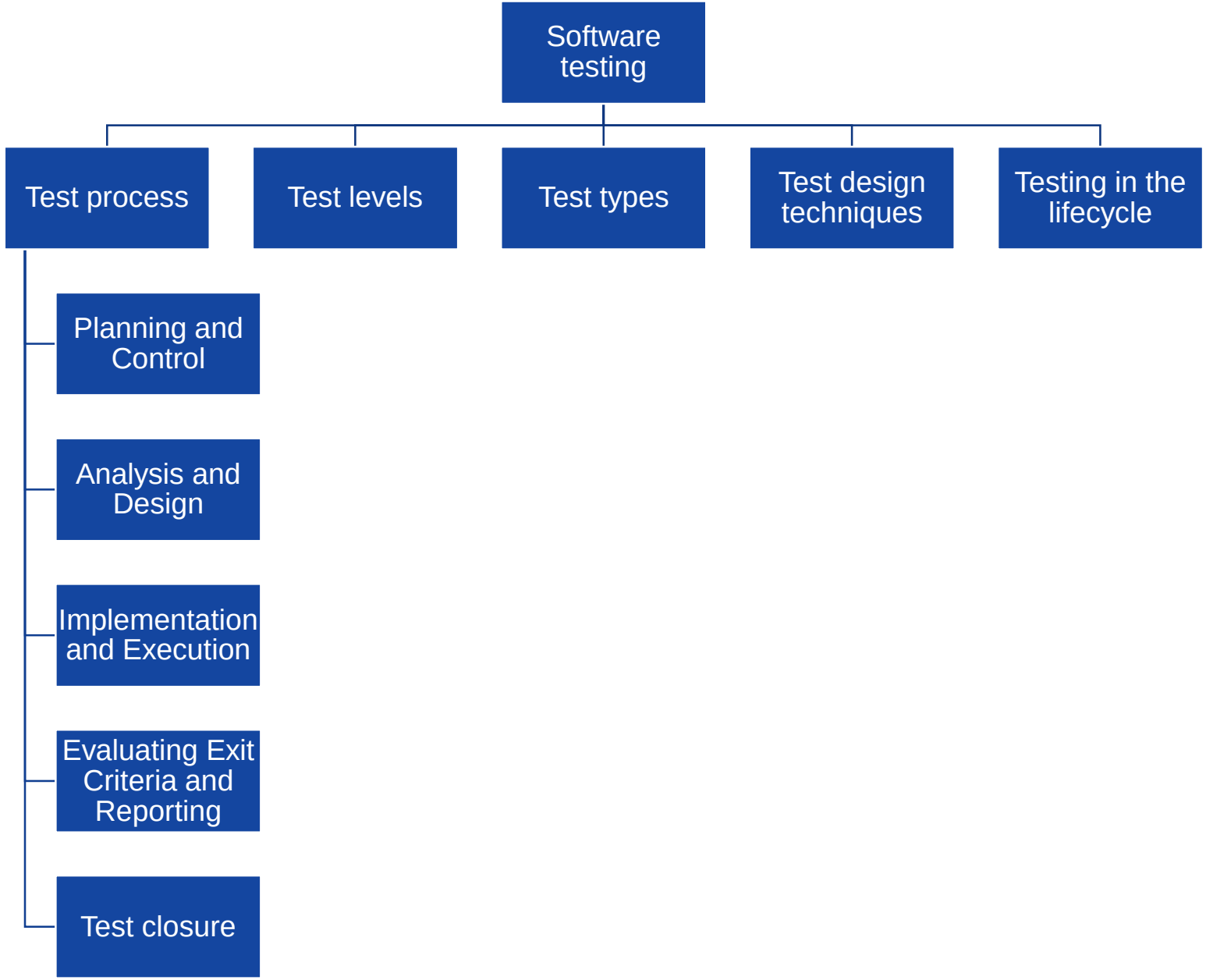

Steps, Tasks, Techniques

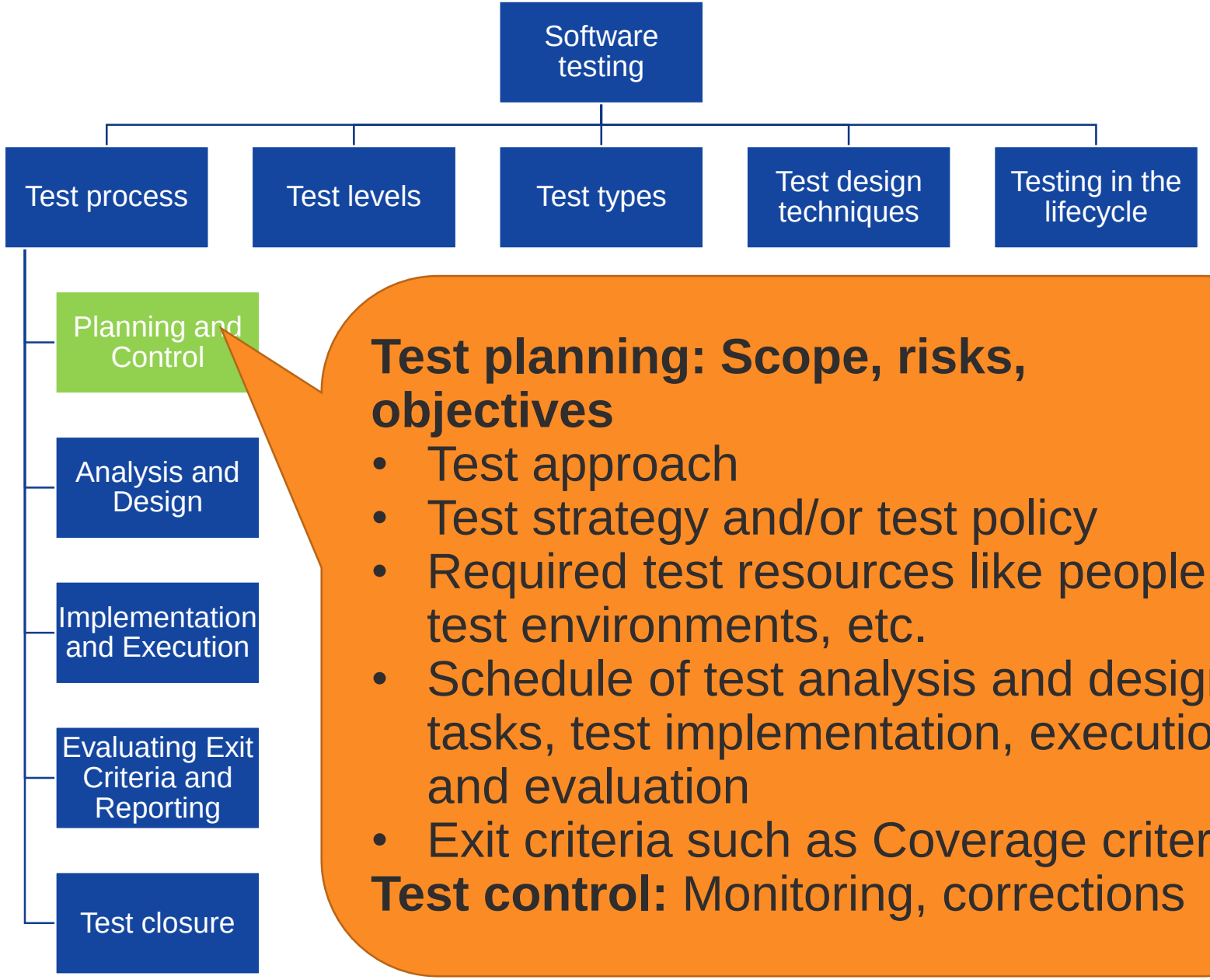
# V&V: Verification and Validation (recap)

| Verification | Validation |
|---|---|
| **„Am I building the system right?”** | **„Am I building the right system?”** |
| Check consistency of development phases | Check the result of the development |
| Conformance of designs/models and their specification | Conformance of the finished system and the user requirements |
| Objective; can be automated | Subjective; checking acceptance |
| Fault model: Design and implementation faults | Fault model: problems in the requirements |
| Not needed if implementation is automatically generated from specification | Not needed if the specification is correct (very simple) |

# Testing Process

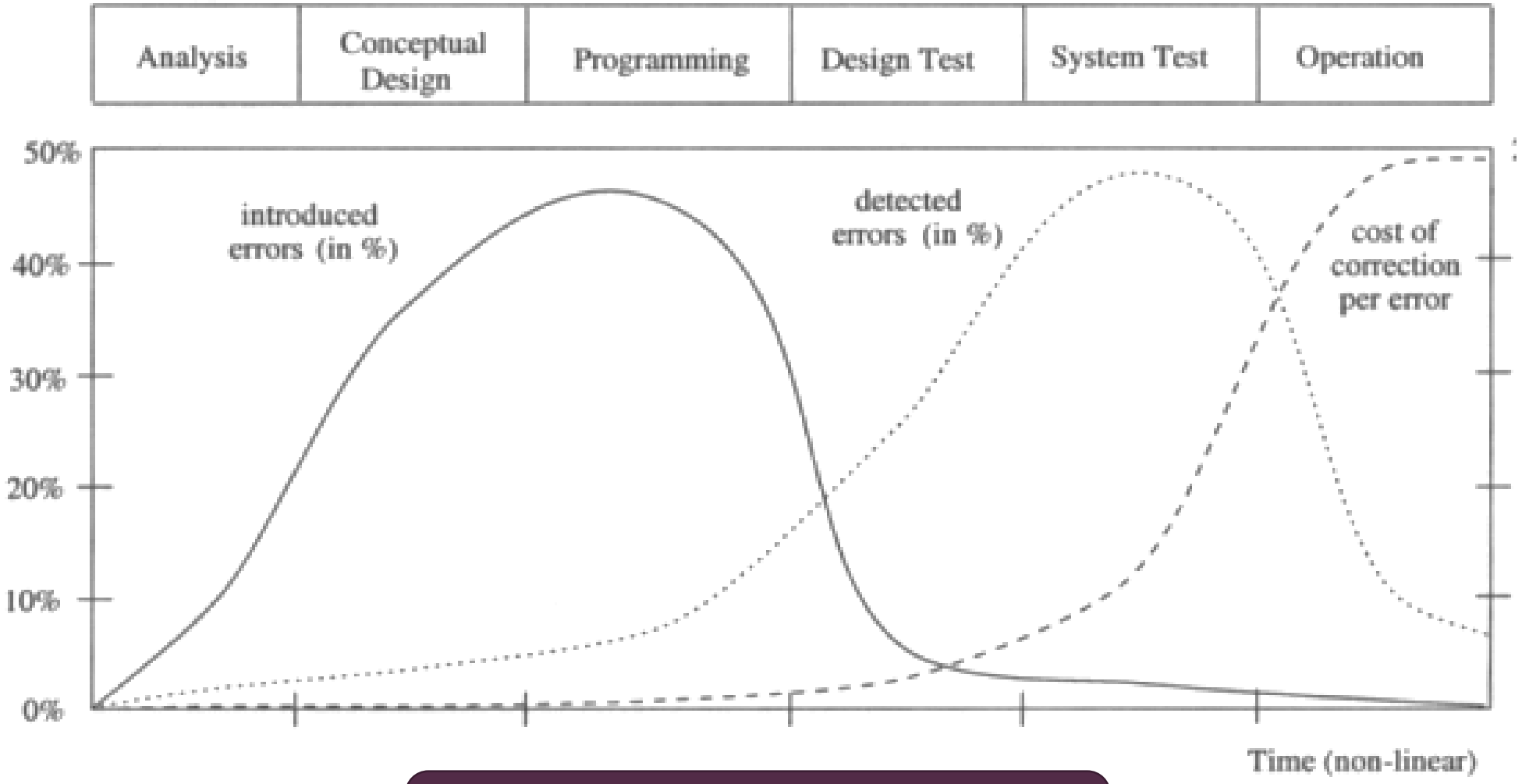



Software testing
Test process
Test levels
Test types
Test design techniques
Testing in the lifecycle
Planning and Control
Analysis and Design
Implementation and Execution
Evaluating Exit Criteria and Reporting
Test closure




See Annex F of ISO 29119-3:2013 for examples

# Distribution and Cost of Bugs




Early V&V reduces cost!

# Test Strategy

- Possible (simple) example:
  - Test-driven development
  - Module & system
  - JUnit & GUI Tester
  - Developers & test engineers
  - At least 90% statement coverage
    & cover every use case / requirement
  - Test Report according to IEEE 29119-3

See Annex F of ISO 29119-3:2013 for a more complex example

# Test Plan

- Mapping test strategy to the actual test project
  - Test objectives
  - Test objects, test environment
  - Resources, roles
  - Schedules

- Defining test phases
  - Length of phase
  - Exit criteria
  - Measuring quality of testing

# Test Plan Outline: ISO 29119-3:2013(E)

a) Context of the testing:
    i) Project/Test sub-process
    ii) Test item(s)
    iii) Test scope
    iv) Assumptions and constraints
    v) Stakeholders
b) Testing communication
c) Risk register:
    i) Product risks
    ii) Project risks
d) Test strategy:
    i) Test sub-processes
    ii) Test deliverables
    iii) Test design techniques
    iv) Test completion criteria
    v) Metrics to be collected
    vi) Test data requirements
    vii) Test environment requirements
    xi) Retesting and regression testing
    xii) Suspension and resumption criteria
    xiii) Deviations from the Organizational T
e) Testing activities and estimates
f) Staffing:
    i) Roles, activities, and responsibilities
    ii) Hiring needs
    iii) Training needs
g) Schedule

See Annex F of the standard for examples (agile / traditional)

# Risk-based Testing

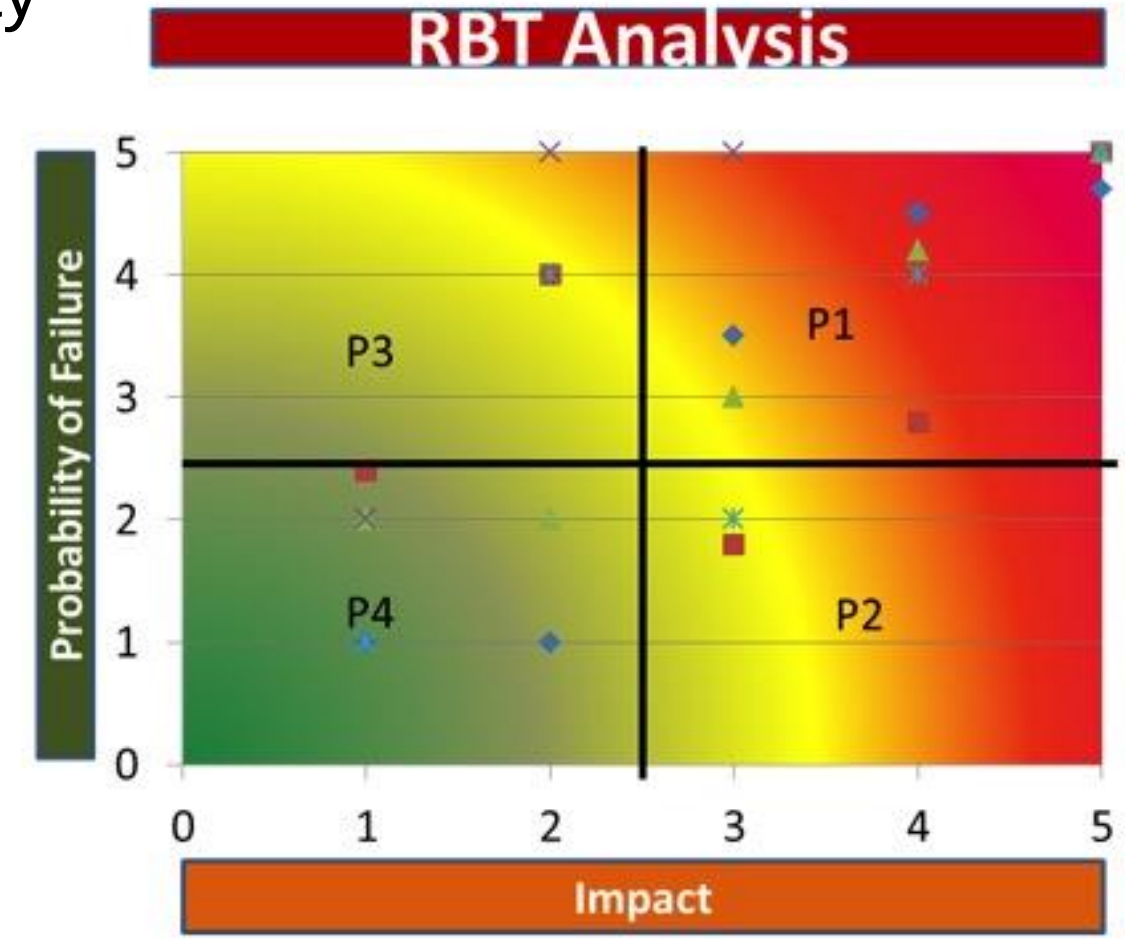
- Level of risk = impact * probability

- P1 – Critical Impact: Must be tested; ideal candidate for automation (candidate for automated smoke testing)

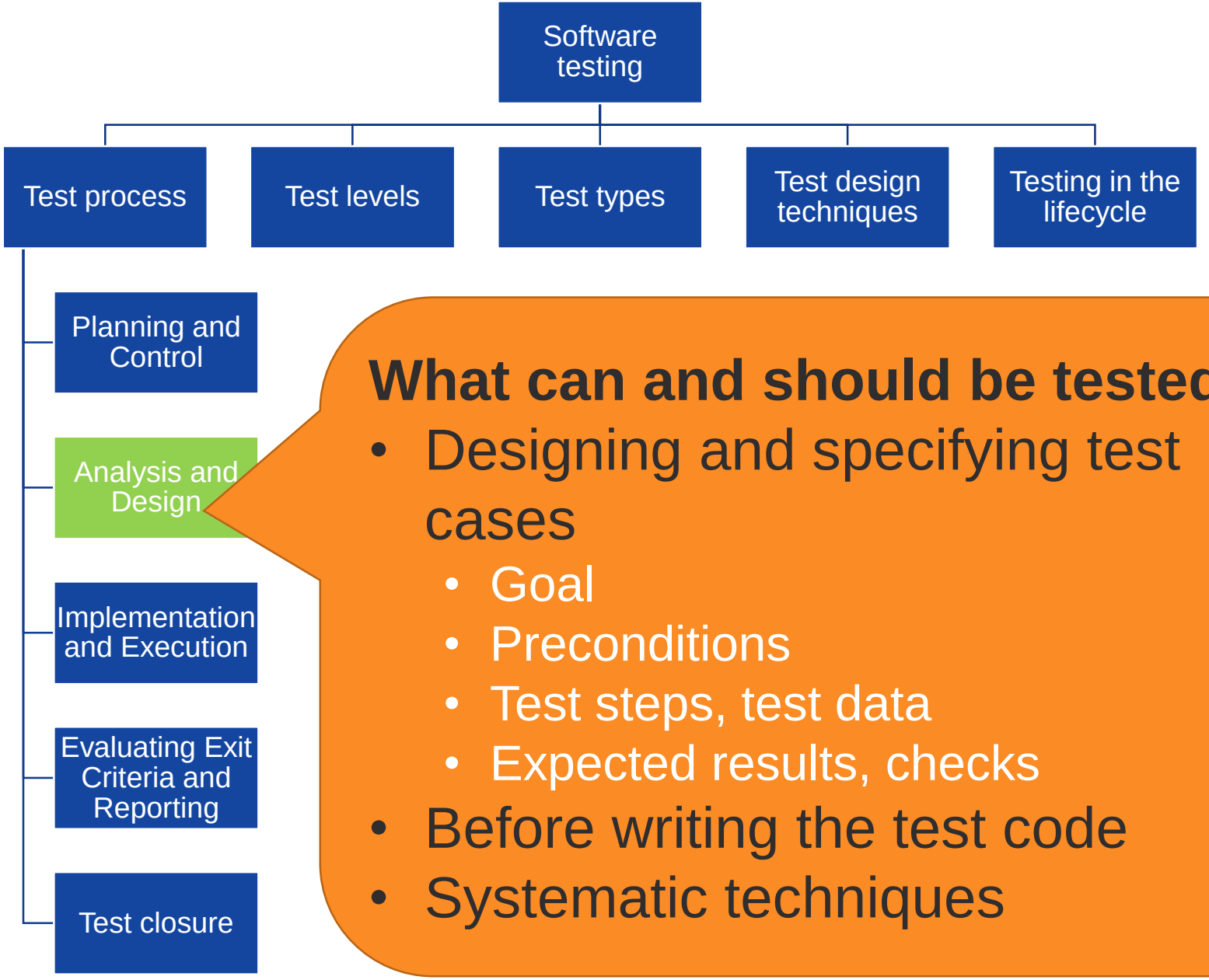
- P2 – High Impact: Should be tested

- P3 – Medium Impact: Can be tested if budget and schedule permits

- P4 – Low Impact: May not be tested; no impact on application and no need of automation

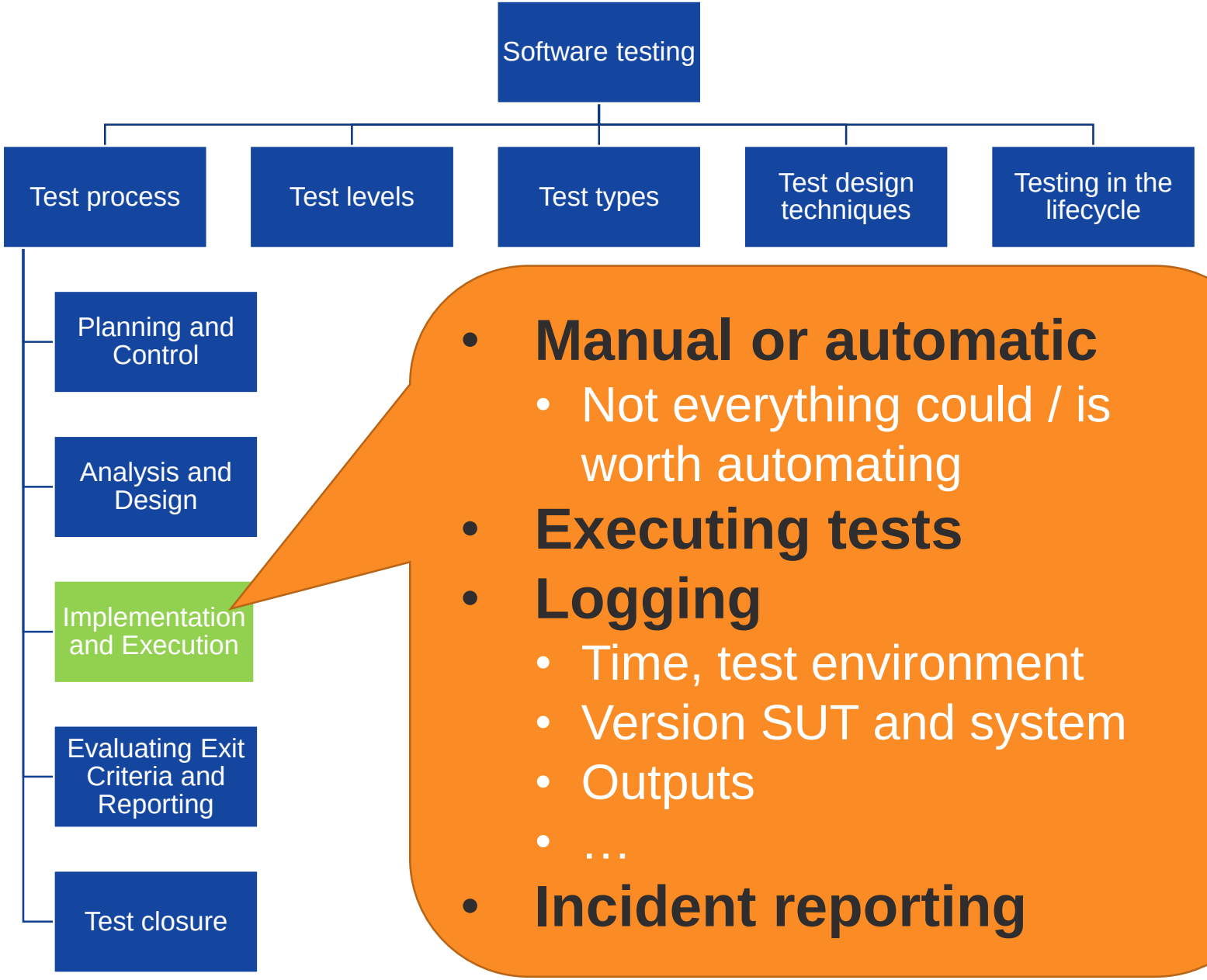


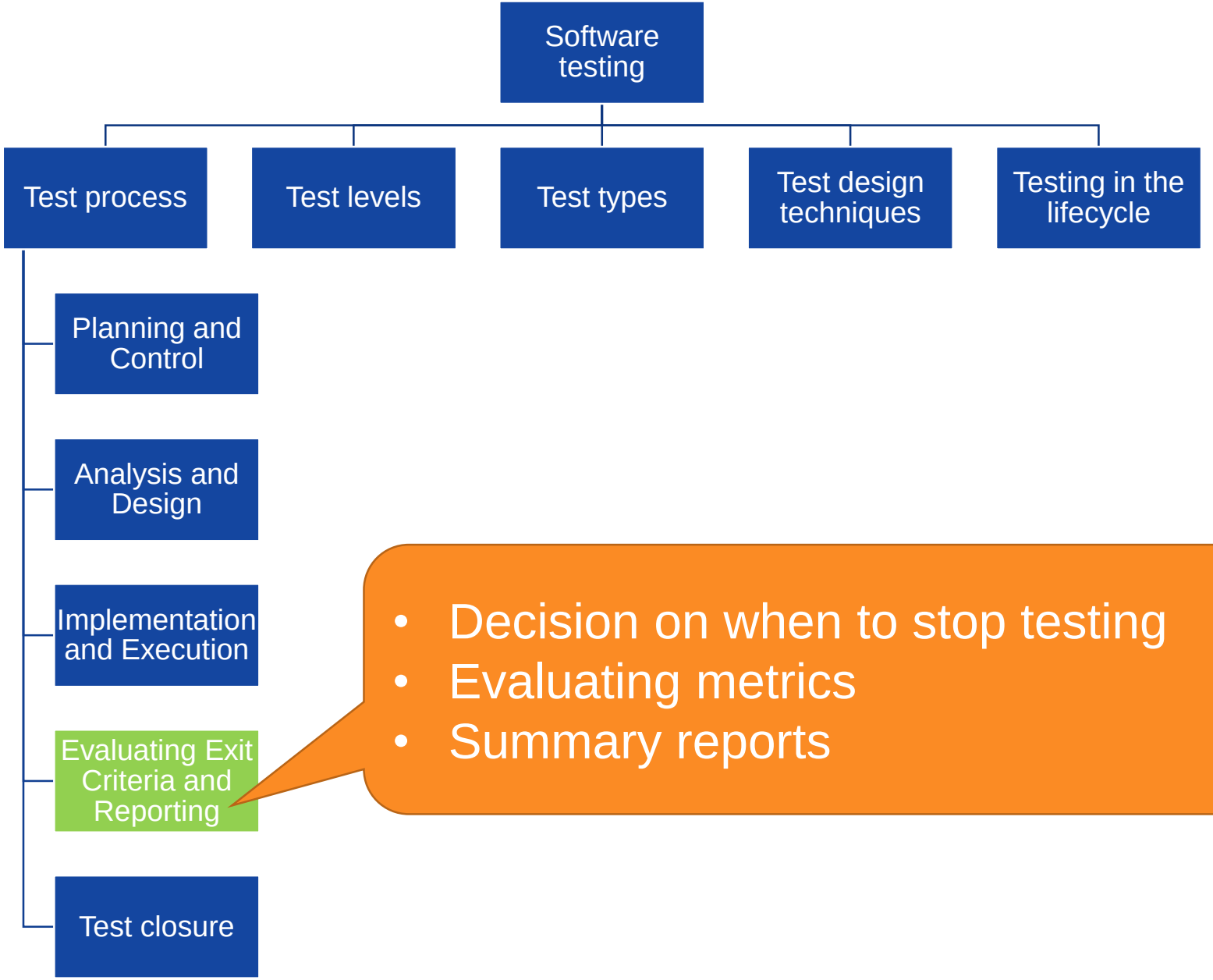

Source: [Risk-Based Testing: Test Only What Matters]



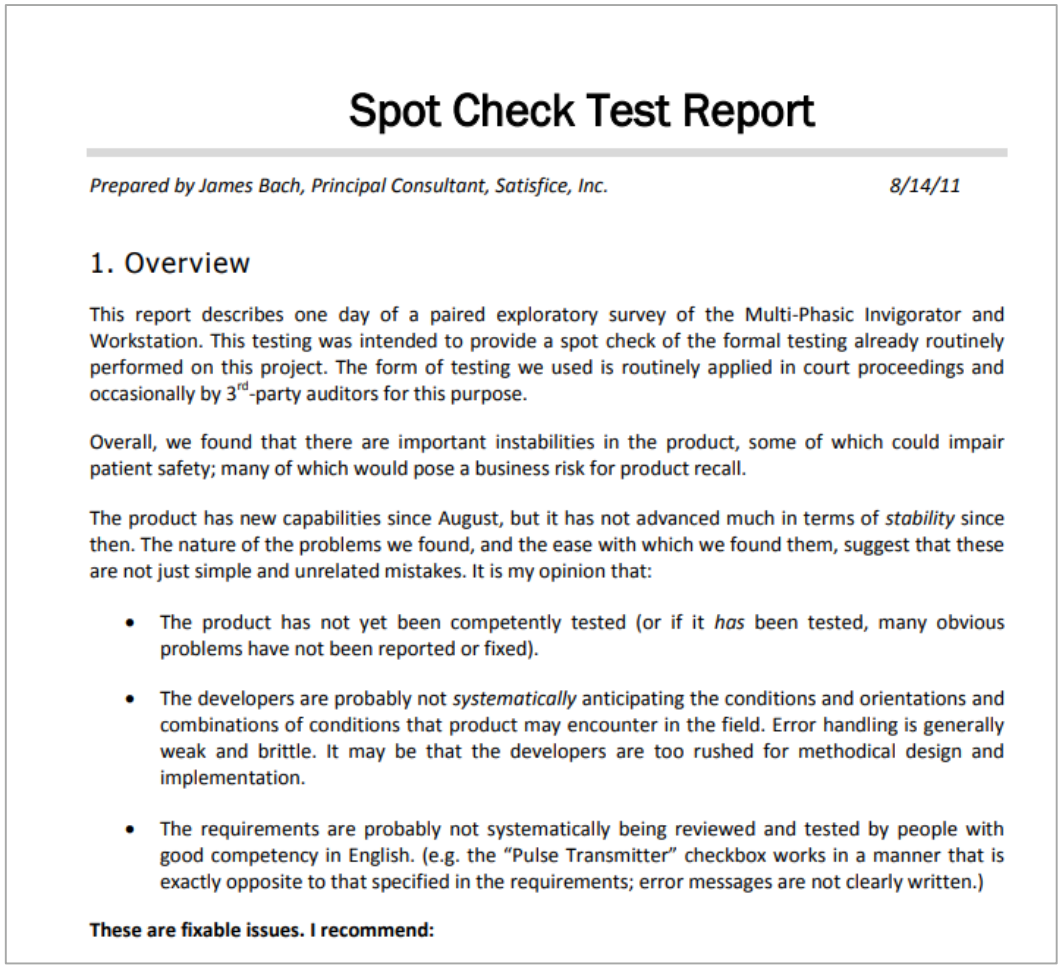

**What can and should be tested?**

- Designing and specifying test cases
  - Goal
  - Preconditions
  - Test steps, test data
  - Expected results, checks
- Before writing the test code
- Systematic techniques


Software testing
Test process
Test levels
Test types
Test design techniques
Testing in the lifecycle
Planning and Control
Analysis and Design
Implementation and Execution
Evaluating Exit Criteria and Reporting
Test closure
Manual or automatic
Not everything could / is worth automating
Executing tests
Logging
Time, test environment
Version SUT and system
Outputs
…
Incident reporting


Software testing
Test process
Test levels
Test types
Test design techniques
Testing in the lifecycle
Planning and Control
Analysis and Design
Implementation and Execution
Evaluating Exit Criteria and Reporting
Test closure
Decision on when to stop testing
Evaluating metrics
Summary reports

# Test Report (Example)



## Spot Check Test Report

*Prepared by James Bach, Principal Consultant, Satisfice, Inc.* *8/14/11*

### 1. Overview

This report describes one day of a paired exploratory survey of the Multi-Phasic Invigorator and Workstation. This testing was intended to provide a spot check of the formal testing already routinely performed on this project. The form of testing we used is routinely applied in court proceedings and occasionally by 3rd-party auditors for this purpose.

Overall, we found that there are important instabilities in the product, some of which could impair patient safety; many of which would pose a business risk for product recall.

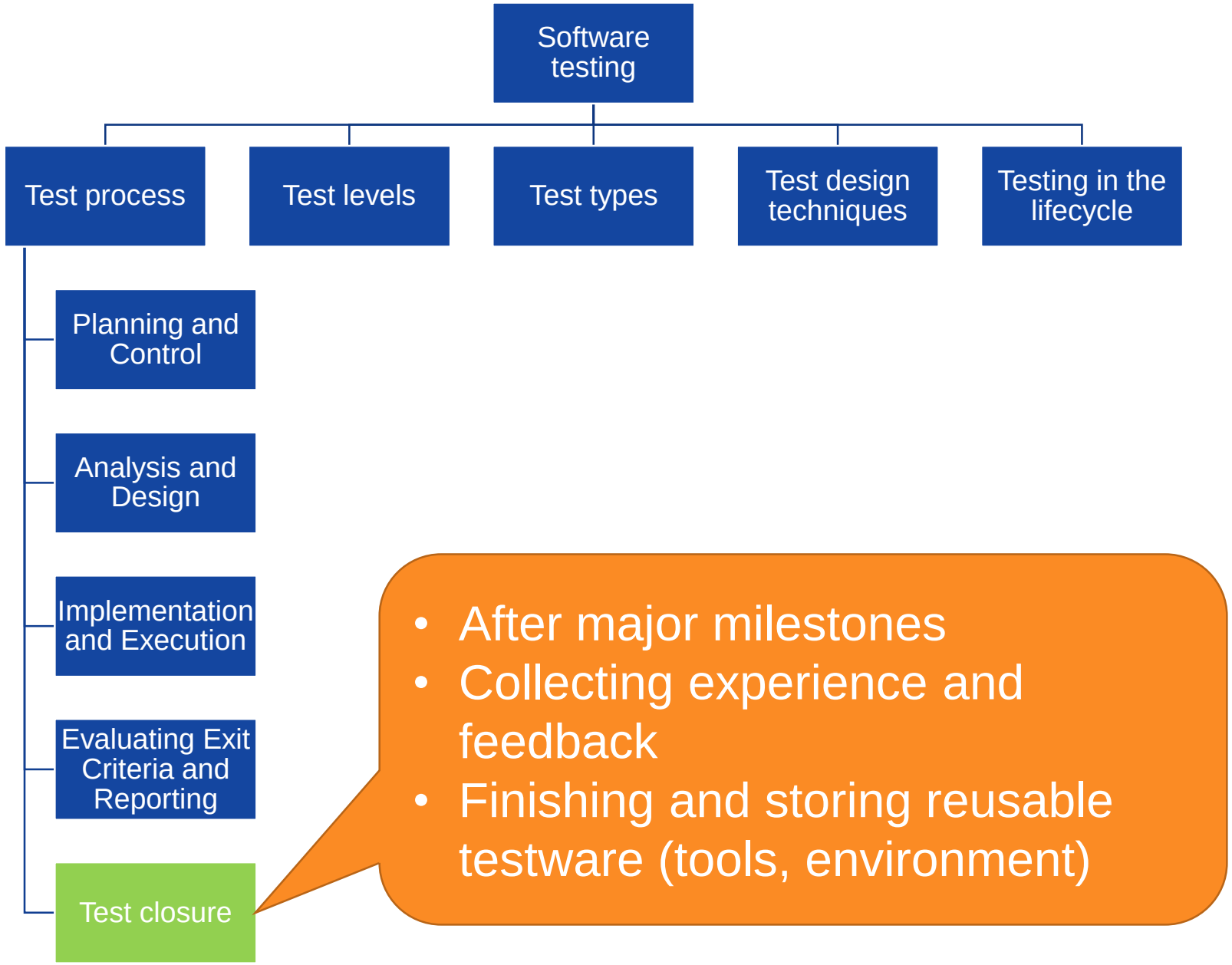
The product has new capabilities since August, but it has not advanced much in terms of *stability* since then. The nature of the problems we found, and the ease with which we found them, suggest that these are not just simple and unrelated mistakes. It is my opinion that:

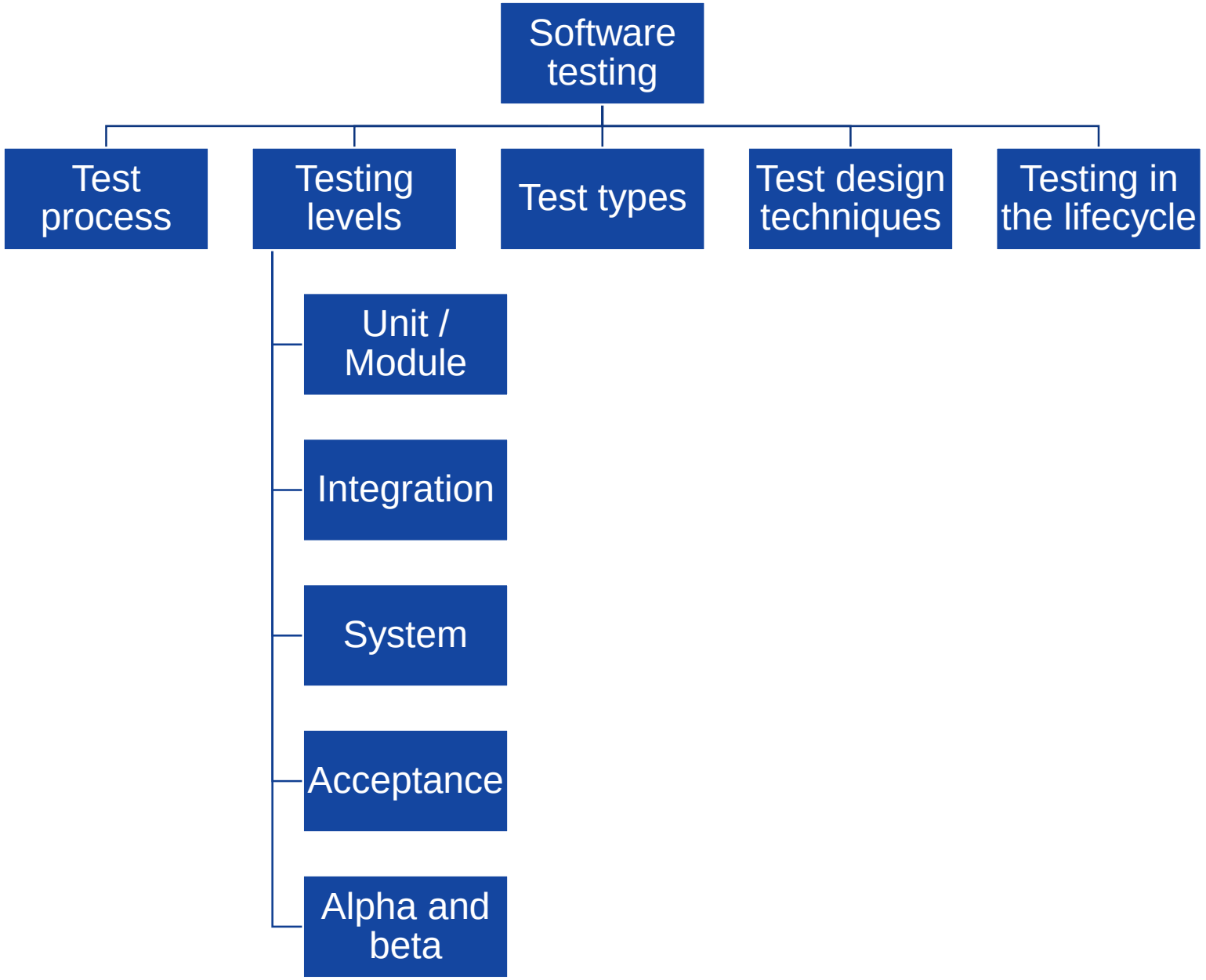
- The product has not yet been competently tested (or if it *has* been tested, many obvious problems have not been reported or fixed).
- The developers are probably not *systematically* anticipating the conditions and orientations and combinations of conditions that product may encounter in the field. Error handling is generally weak and brittle. It may be that the developers are too rushed for methodical design and implementation.
- The requirements are probably not systematically being reviewed and tested by people with good competency in English. (e.g. the "Pulse Transmitter" checkbox works in a manner that is exactly opposite to that specified in the requirements; error messages are not clearly written.)

**These are fixable issues. I recommend:**

- Overview: executive summary
  - Main message (assessment)
  - Important issues
  - Recommendations
- Test process
  - What have we done
- Details
  - Severe problems
  - Other issues

https://www.developsense.com/resources/ReportingExamples/mpim-report.pdf


Software testing
Test process
Test levels
Test types
Test design techniques
Testing in the lifecycle
Planning and Control
Analysis and Design
Implementation and Execution
Evaluating Exit Criteria and Reporting
Test closure
• After major milestones
• Collecting experience and feedback
• Finishing and storing reusable testware (tools, environment)


Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Unit / Module
Integration
System
Acceptance
Alpha and beta

# Characteristics of Tests in Different Levels

- Recommendations from *How Google Tests Software*:

| | Small | Medium | Large |
|---|---|---|---|
| Execution time | < 100 ms | < 1 sec | As fast as poss. |
| Time limit (kill) | 1 minute | 5 minutes | 1 hour |

| Resource | Small | Medium | Large |
|---|---|---|---|
| Network (socket) | Mocked | only localhost | Yes |
| Database | Mocked | Yes | Yes |
| File access | Mocked | Yes | Yes |
| System call | No | Not recommended | Yes |
| Multiple threads | Not recommended | Yes | Yes |
| Sleep | No | Yes | Yes |
| System properties | No | Yes | Yes |

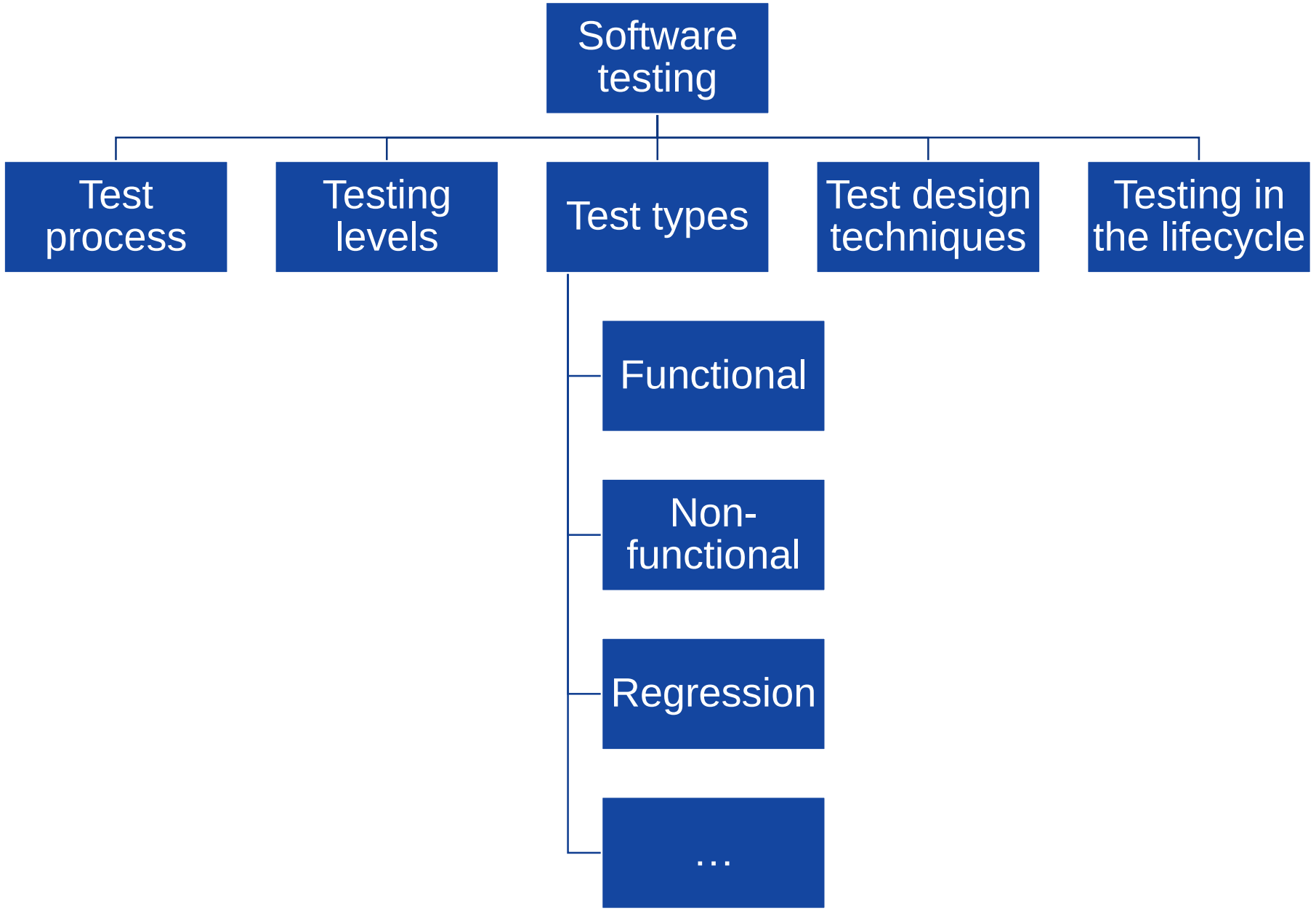
# Difference Between Unit and Other Tests

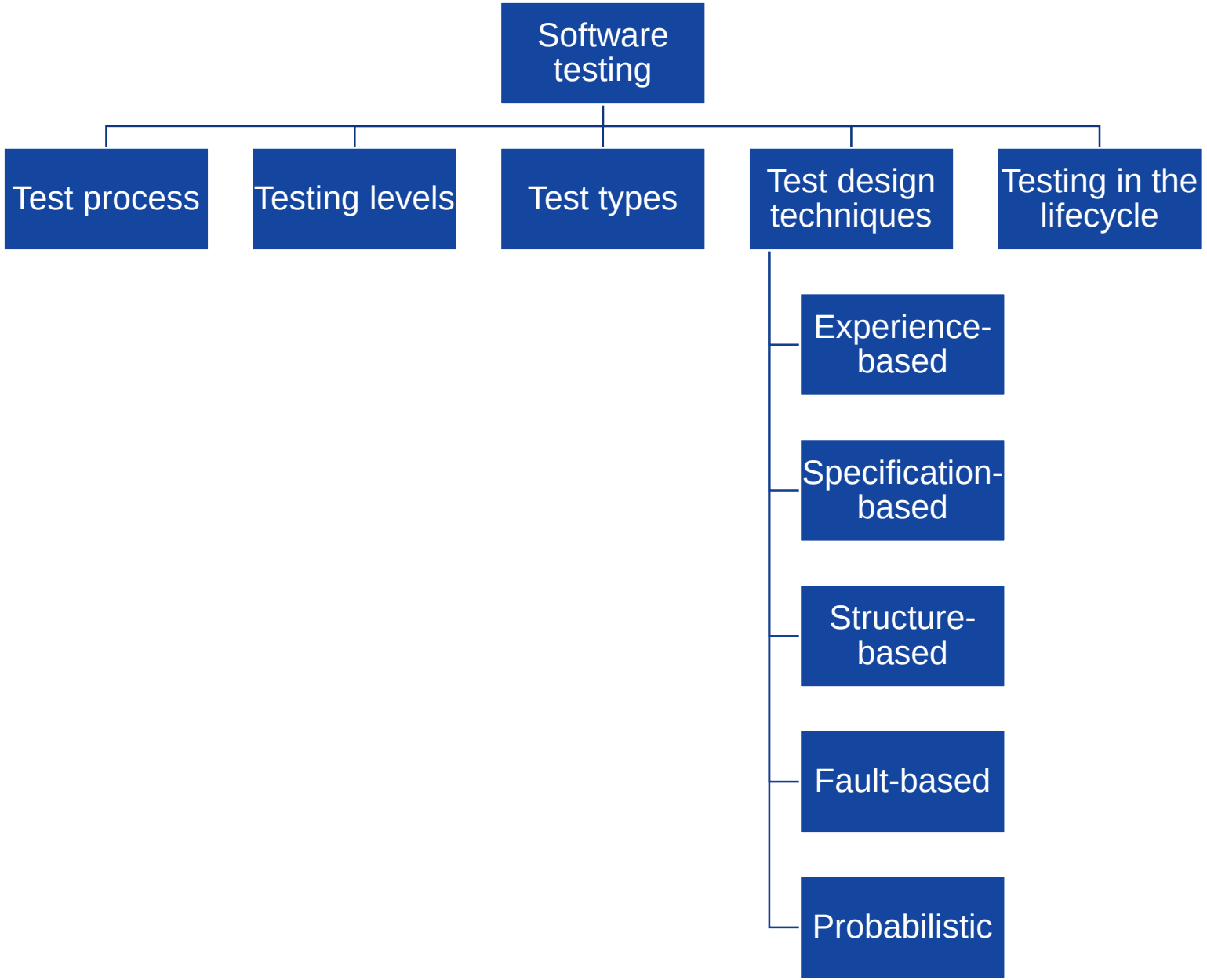
**Unit tests**

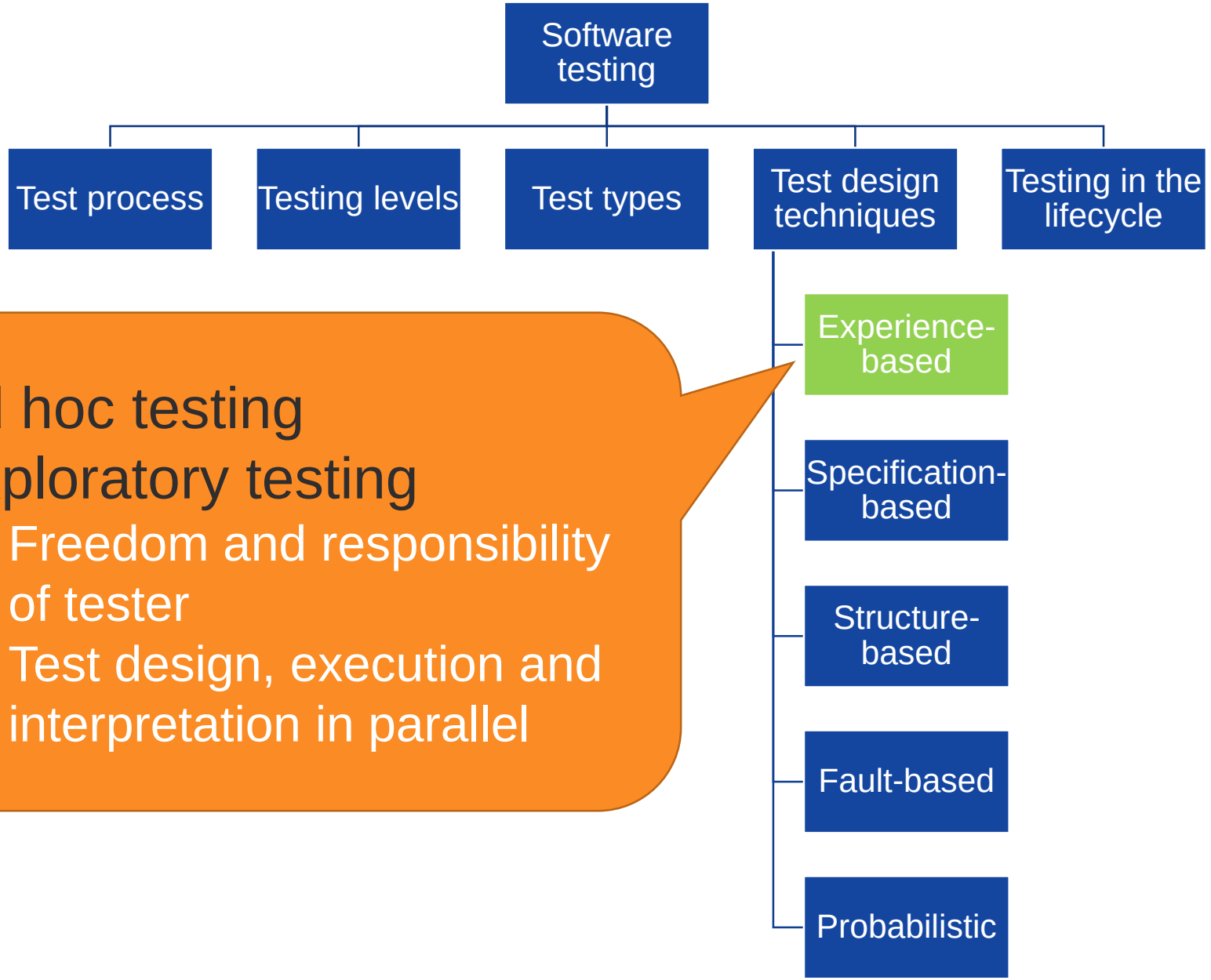
- Calling some methods + varying parameters
- Mostly from test source code
- Working in the IDE mostly

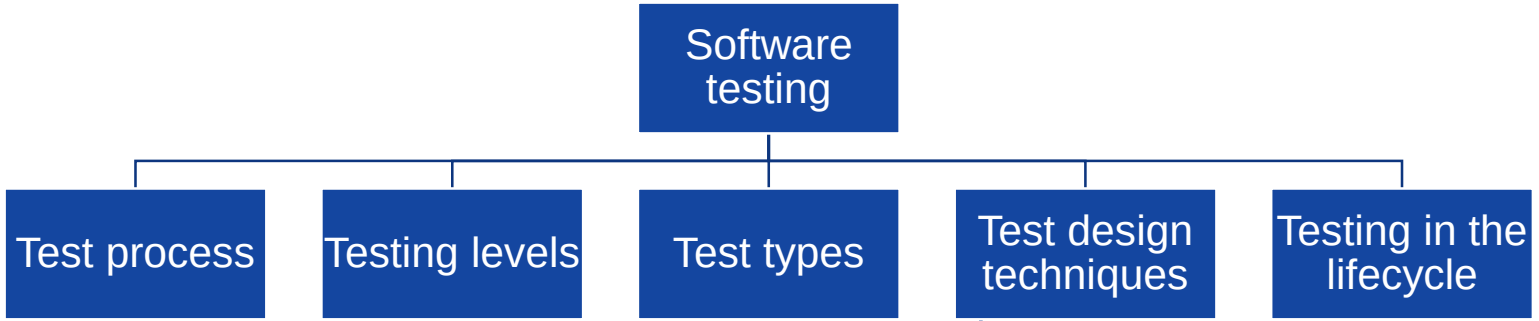
**Integration+ tests**

- Call + parameters are only a small part
- Configuration and context are much bigger effort
- Files, databases, containers, networking…
- Lots of new failure modes

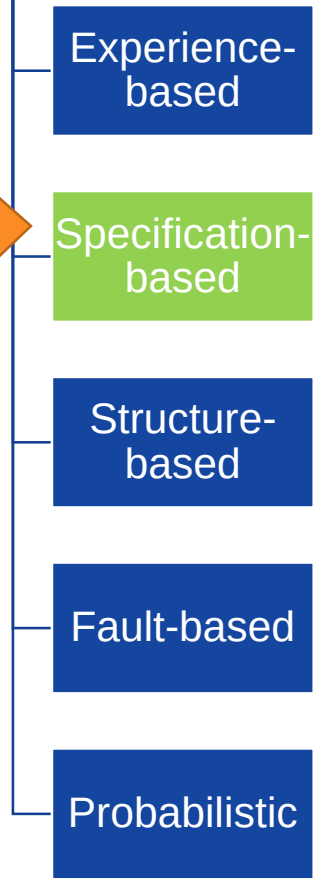

Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Functional
Non-functional
Regression
…

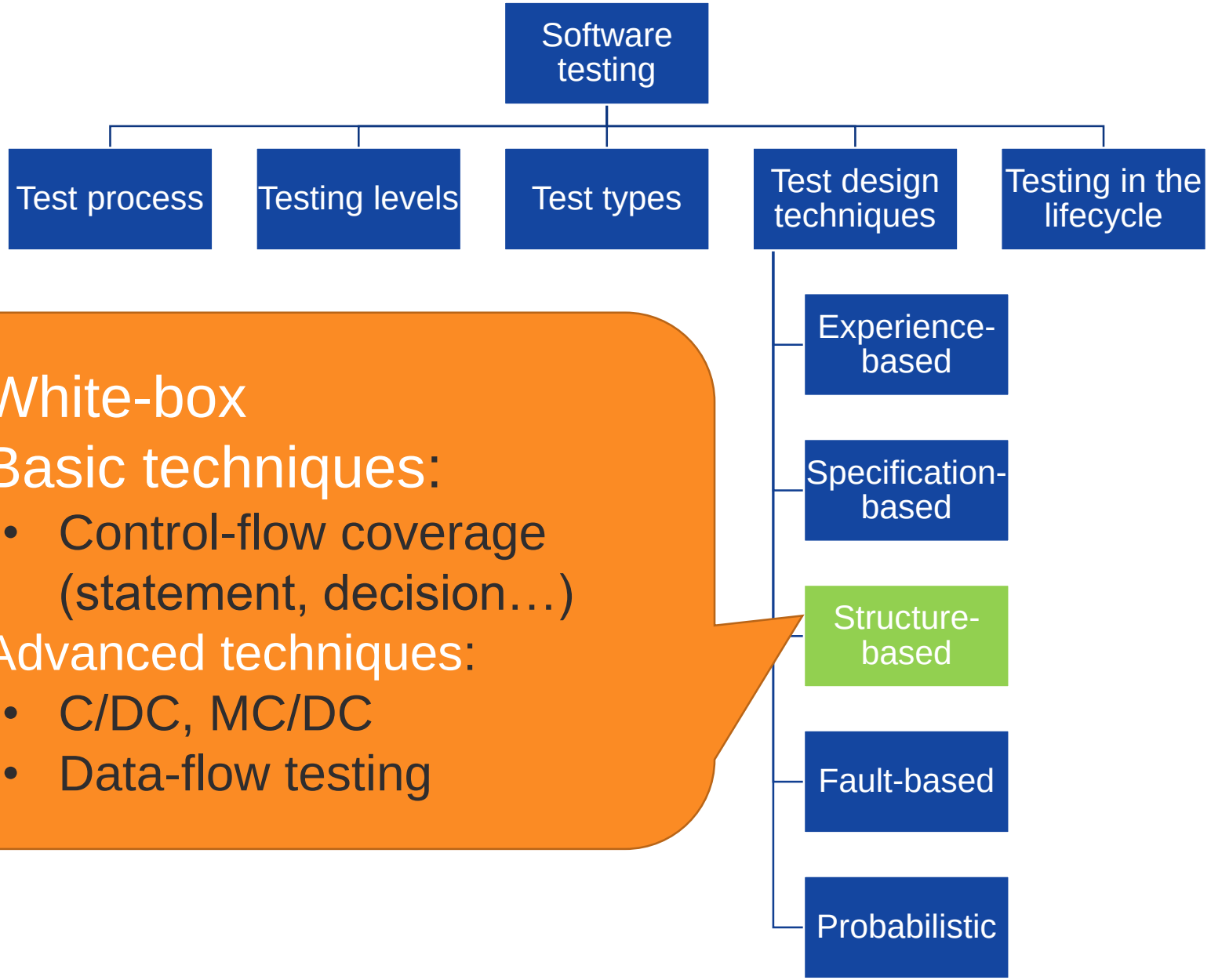

Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Experience-based
Specification-based
Structure-based
Fault-based
Probabilistic

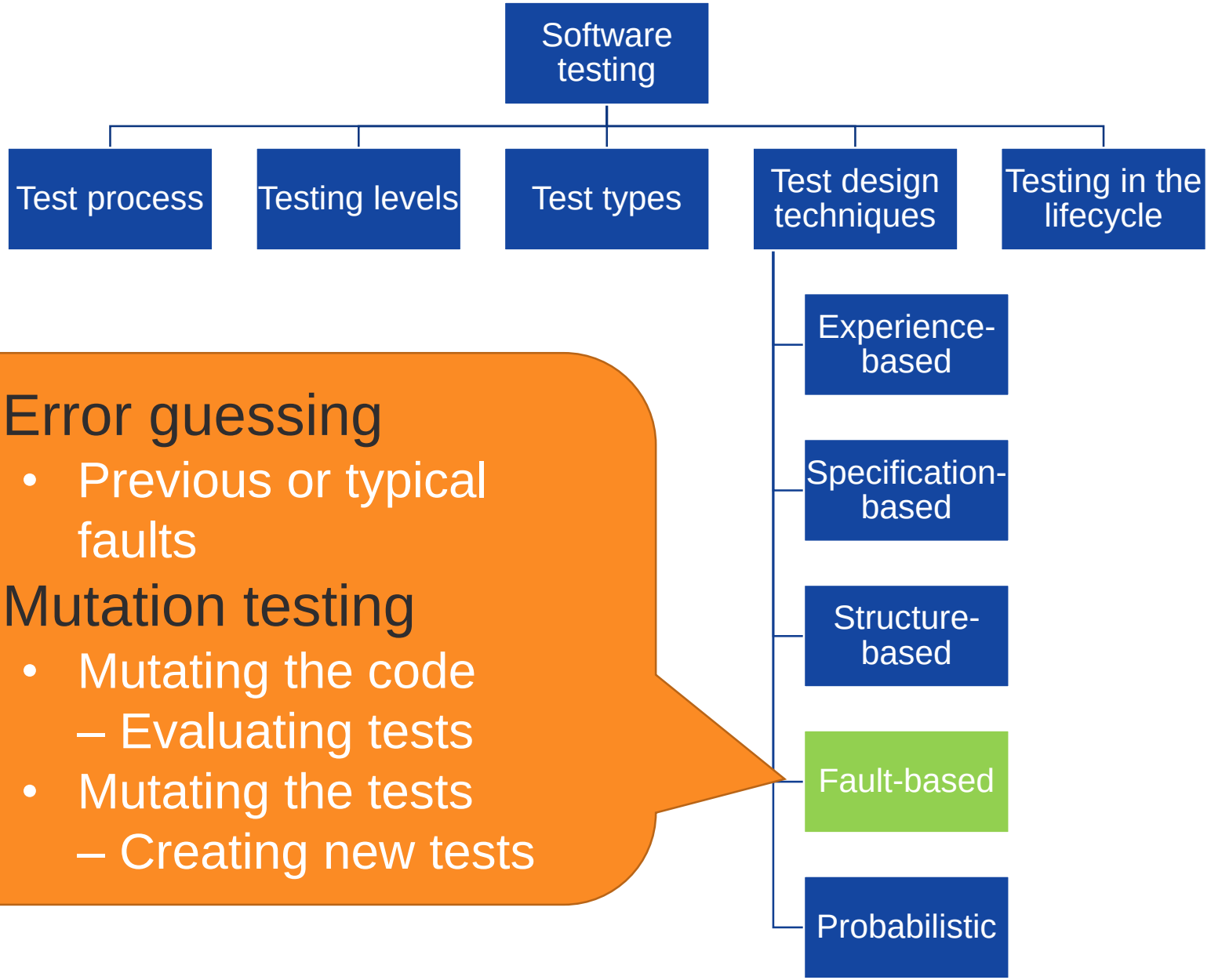

Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Experience-based
Specification-based
Structure-based
Fault-based
Probabilistic
• Ad hoc testing
• Exploratory testing
• Freedom and responsibility of tester
• Test design, execution and interpretation in parallel

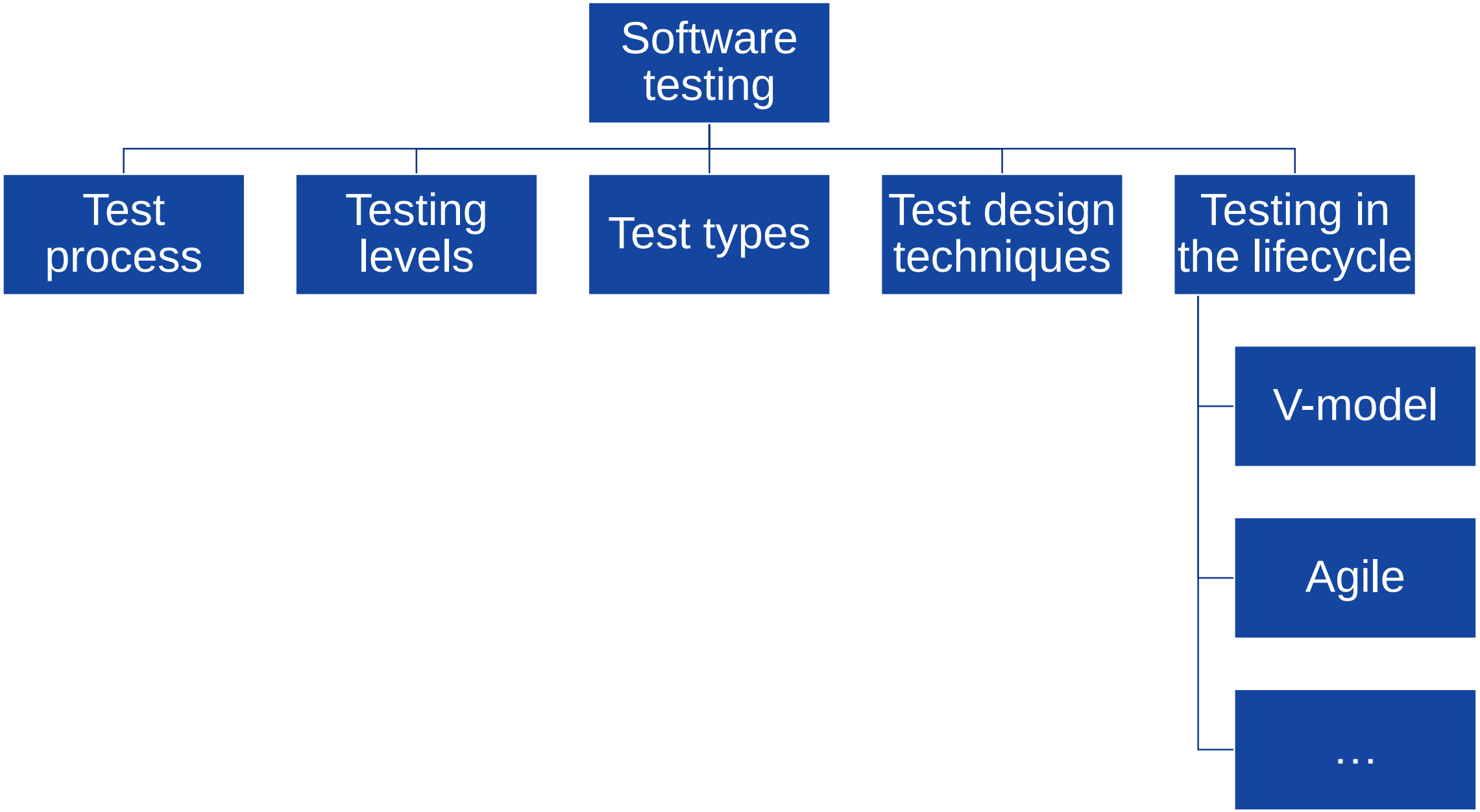


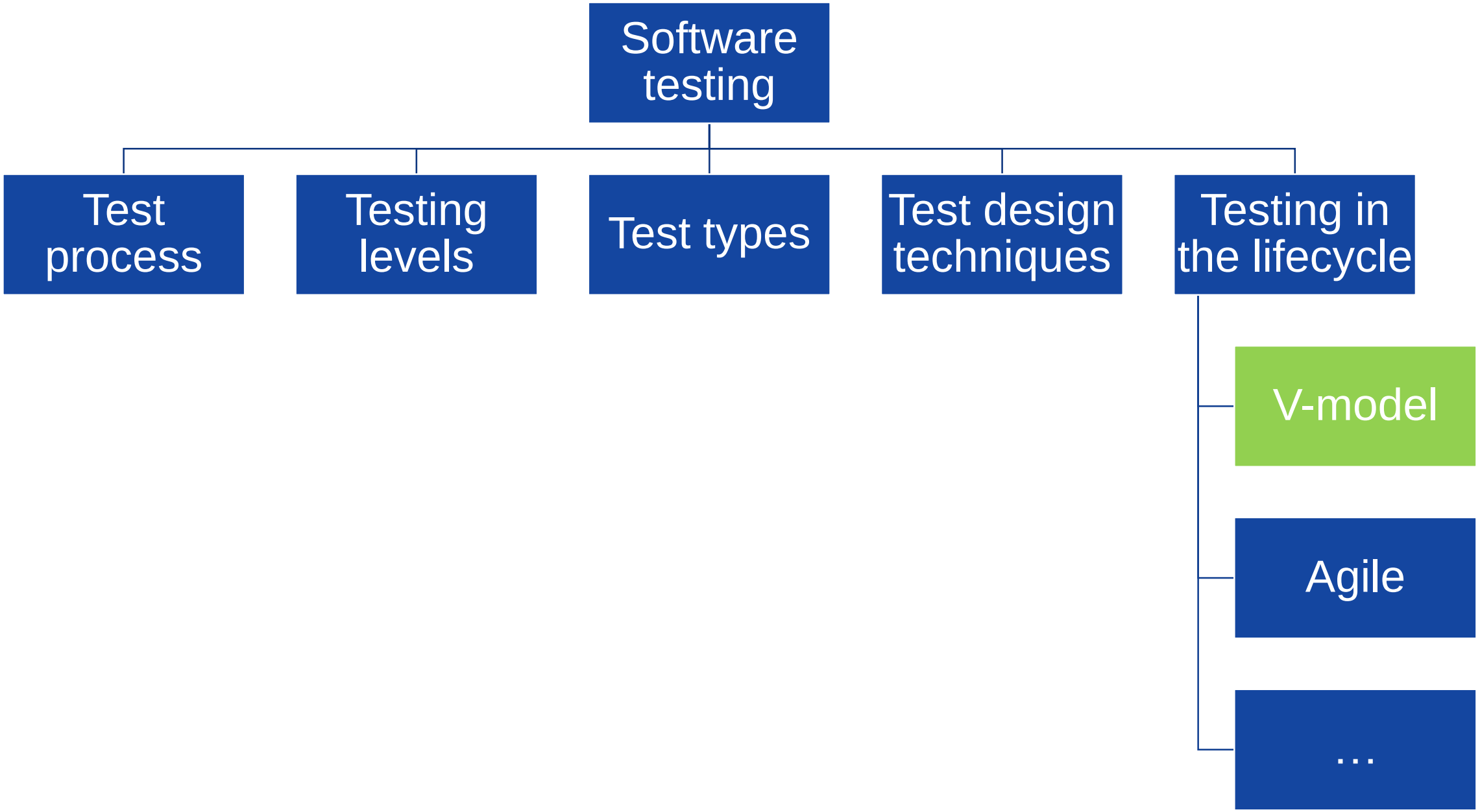

- Black box
- Basic techniques:
  - Equivalence partitions, boundary values, use cases, user stories…
- Advanced techniques:
  - Decision tables, combinatorial testing (n-wise)…



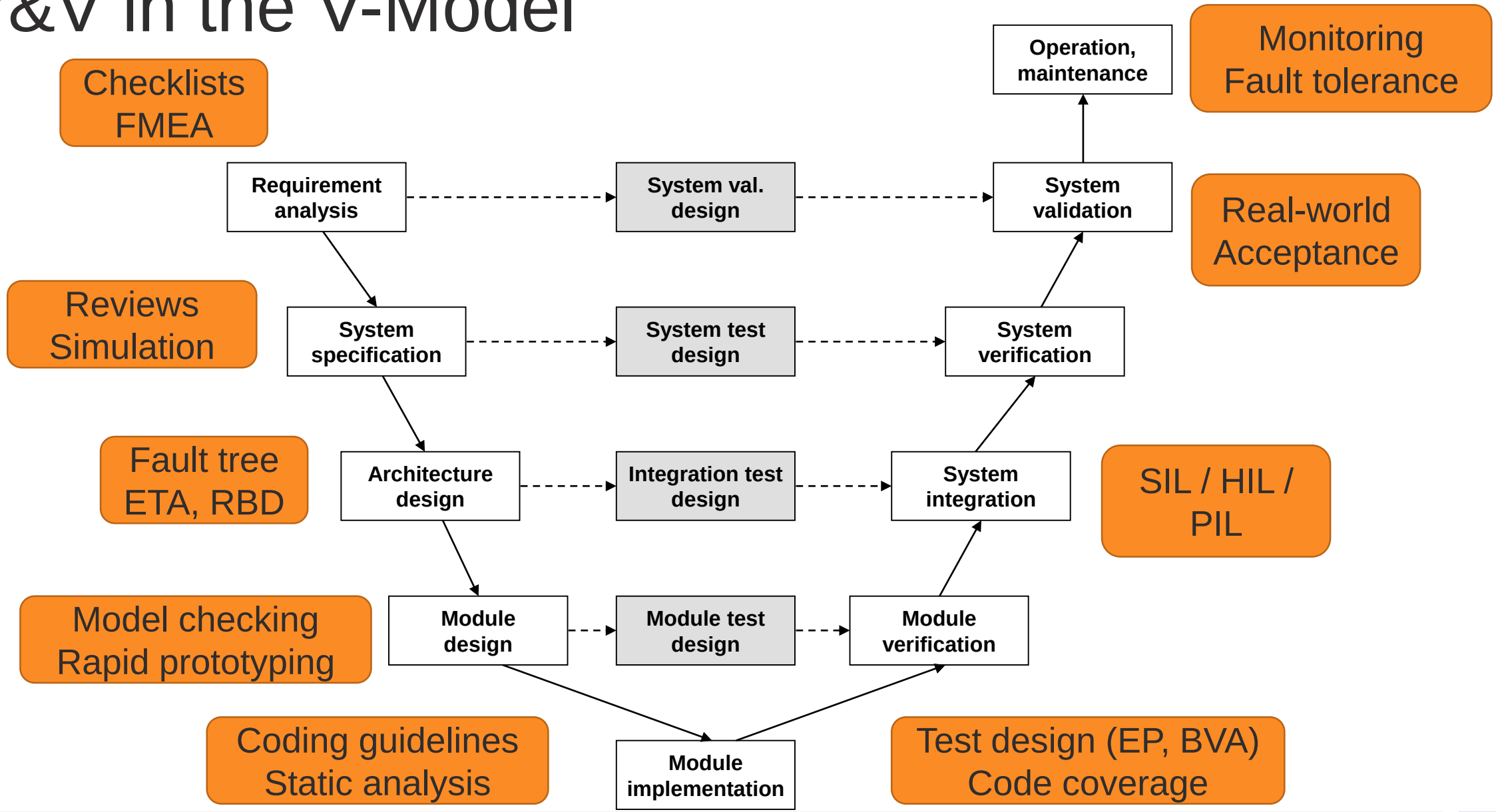

Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Experience-based
Specification-based
Structure-based
Fault-based
Probabilistic
White-box
Basic techniques:
Control-flow coverage (statement, decision…)
Advanced techniques:
C/DC, MC/DC
Data-flow testing

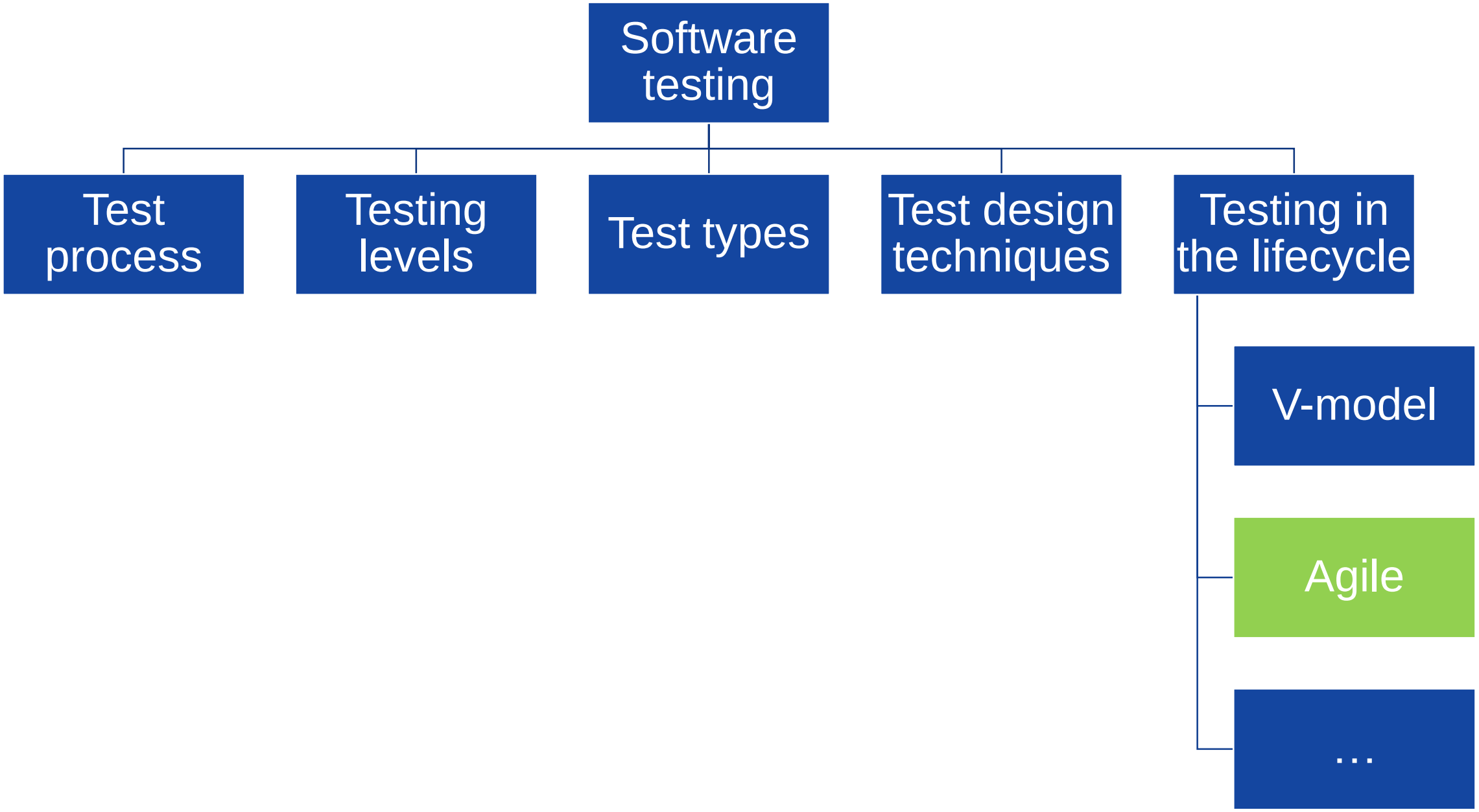

Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Experience-based
Specification-based
Structure-based
Fault-based
Probabilistic
• Error guessing
• Previous or typical faults
• Mutation testing
• Mutating the code
– Evaluating tests
• Mutating the tests
– Creating new tests


Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
V-model
Agile
…

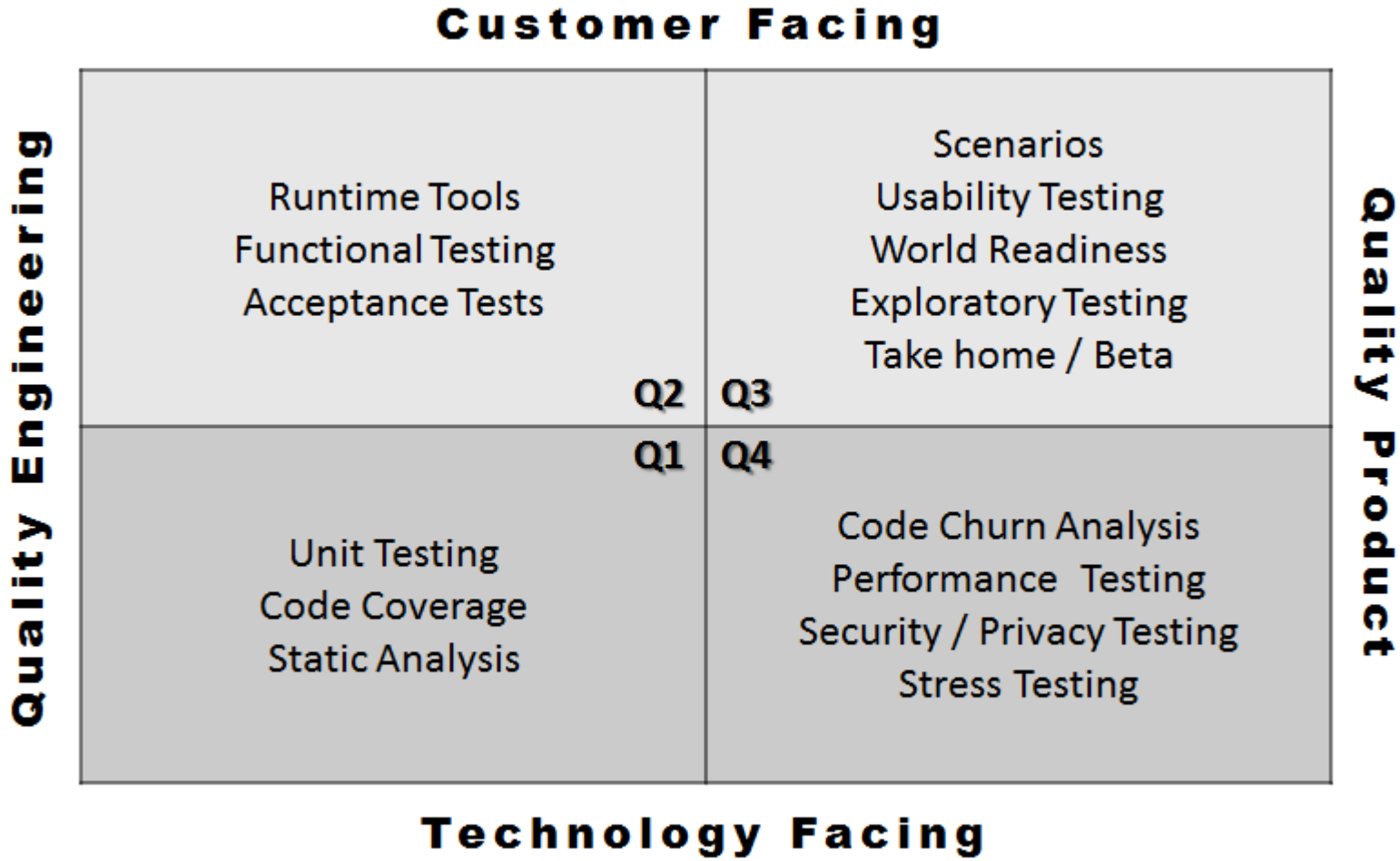

Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
V-model
Agile
…

# V&V in the V-Model

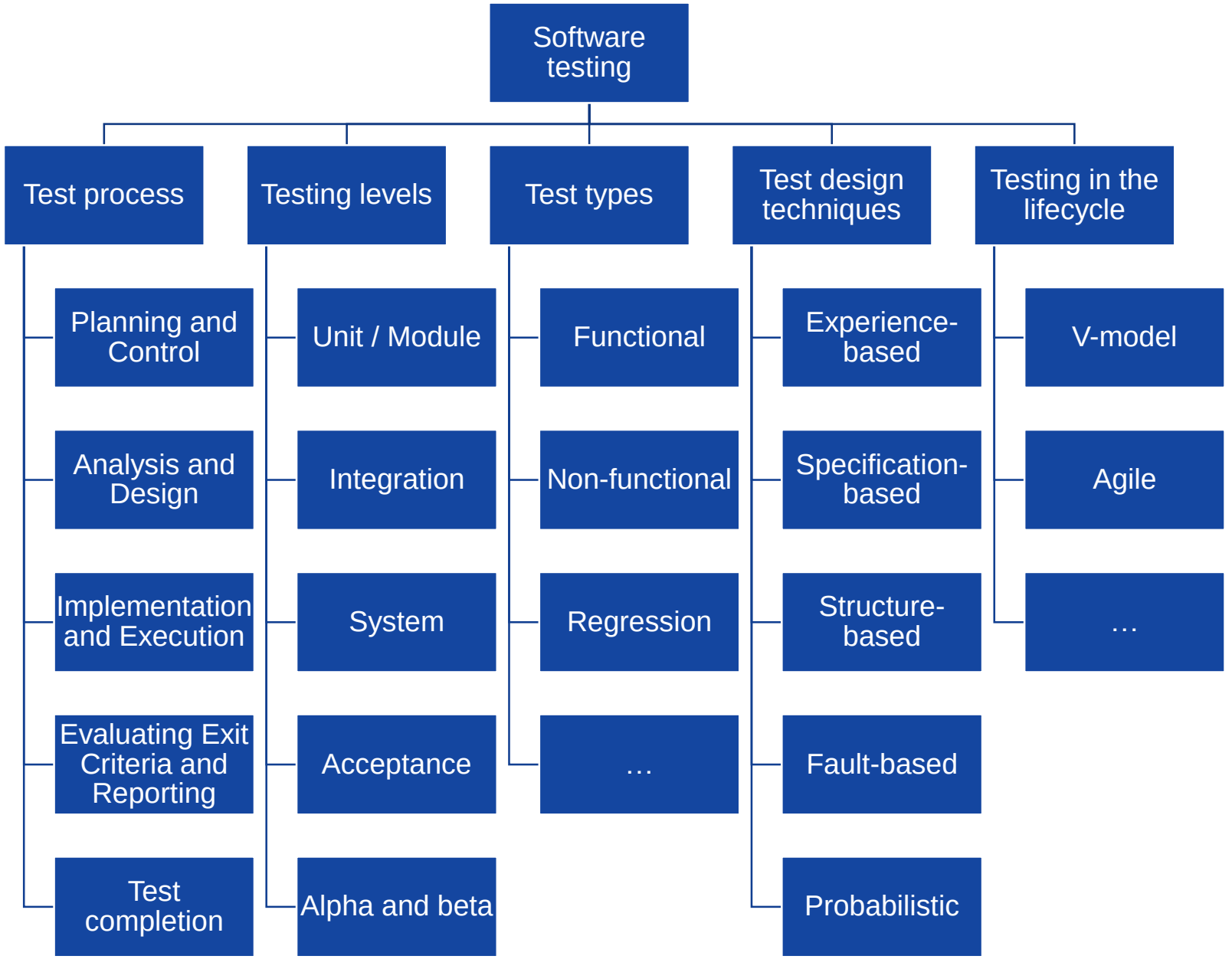



Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
V-model
Agile
…

# Agile Testing Quadrants



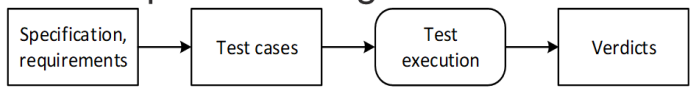

Source: http://angryweasel.com/blog/riffing-on-the-quadrants/


Software testing
Test process
Testing levels
Test types
Test design techniques
Testing in the lifecycle
Planning and Control
Analysis and Design
Implementation and Execution
Evaluating Exit Criteria and Reporting
Test completion
Unit / Module
Integration
System
Acceptance
Alpha and beta
Functional
Non-functional
Regression
…
Experience-based
Specification-based
Structure-based
Fault-based
Probabilistic
V-model
Agile
…

# Summary

# Summary

## Basic Concepts of Testing




- SUT: system under test
- Test case
  - a set of inputs, implementation conditions and expected results developed for a specific purpose
- Test suite
- Test oracle
  - A principle or method that helps the tester decide whether the test was successful
- Verdict: pass / fail / error / inconclusive…



## Tasks to Solve

| Test selection | Oracle problem | Exit criteria |
|---|---|---|
| • What input values and data to use? | • Where to find a reliable oracle? | • When to stop testing? |



## Difference Between Unit and Other Tests

**Unit tests**

- Calling some methods + varying parameters
- Mostly from test source code
- Working in the IDE mostly

**Integration+ tests**

- Call + parameters are only a small part
- Configuration and context are much bigger effort
- Files, databases, containers, networking…
- Lots of new failure modes