# Advanced Software Engineering: Software Testing

## COMP 3705(L4)

Anneliese Andrews

Sada Narayanappa


Thomas Thelin

Carina Andersson

# News & Project

- News
- Project
  - Next deadline 10/5: Report to supervisor and peer review group
  - Book time

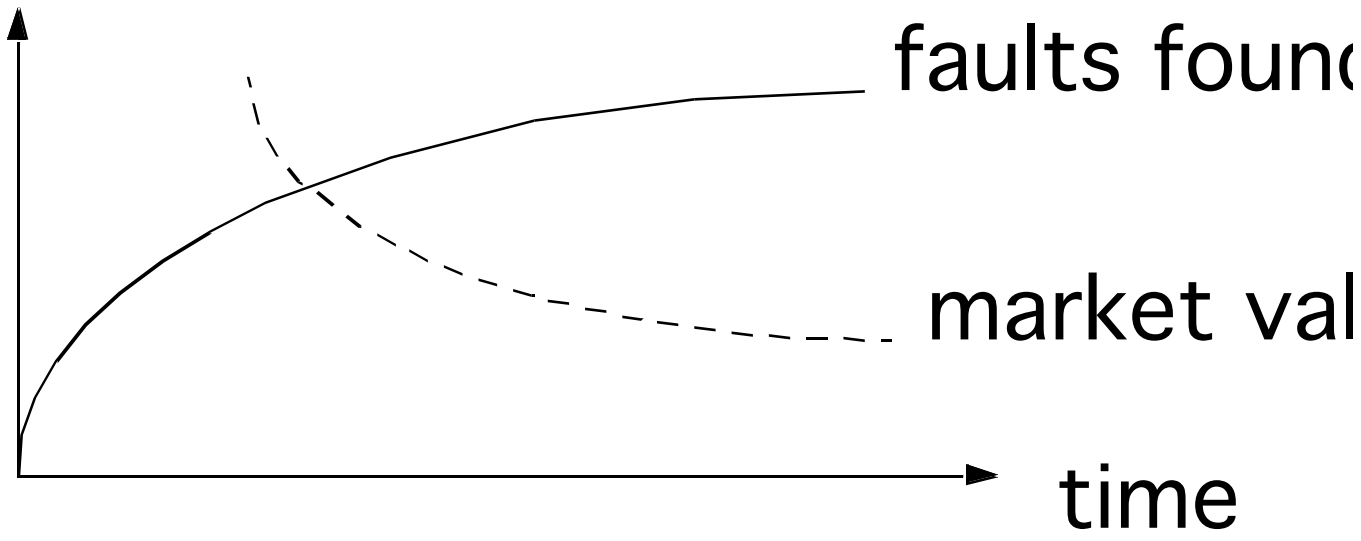
# Lecture

- Chapter 6 (Lab 5)
  - Process
  - Testing levels
- Chapter 7
  - Framework
  - Documentation
- Appendix II
  - Sample test plan

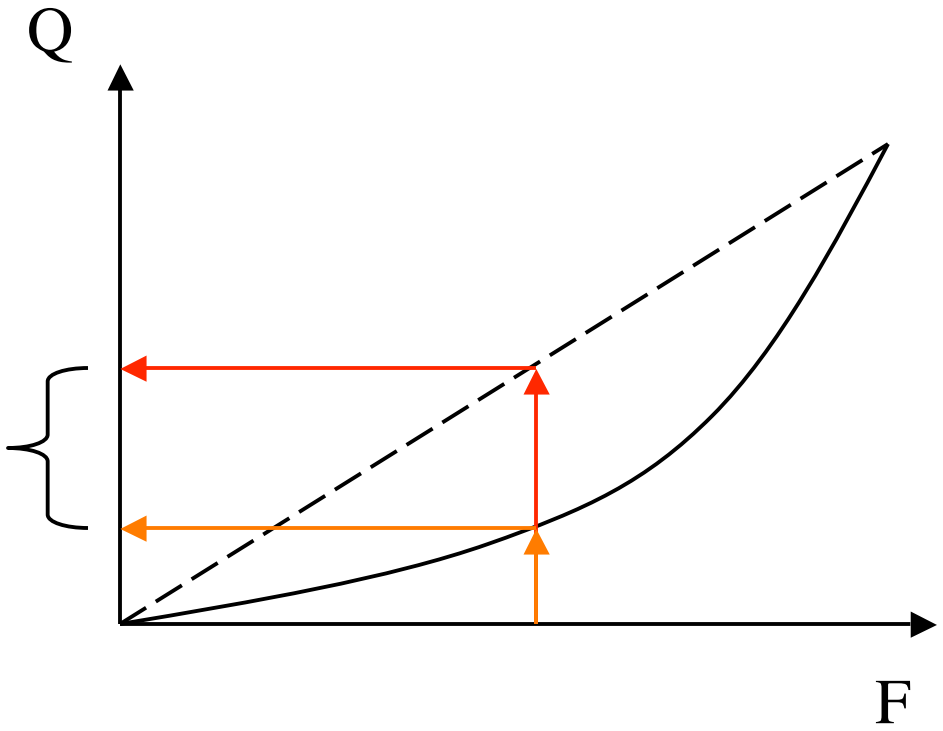
# Test purpose – priority

- Functionality
- Reliability
- Performance
- Security
- Safety
- Cost
- Maintainability
- Portability
- …

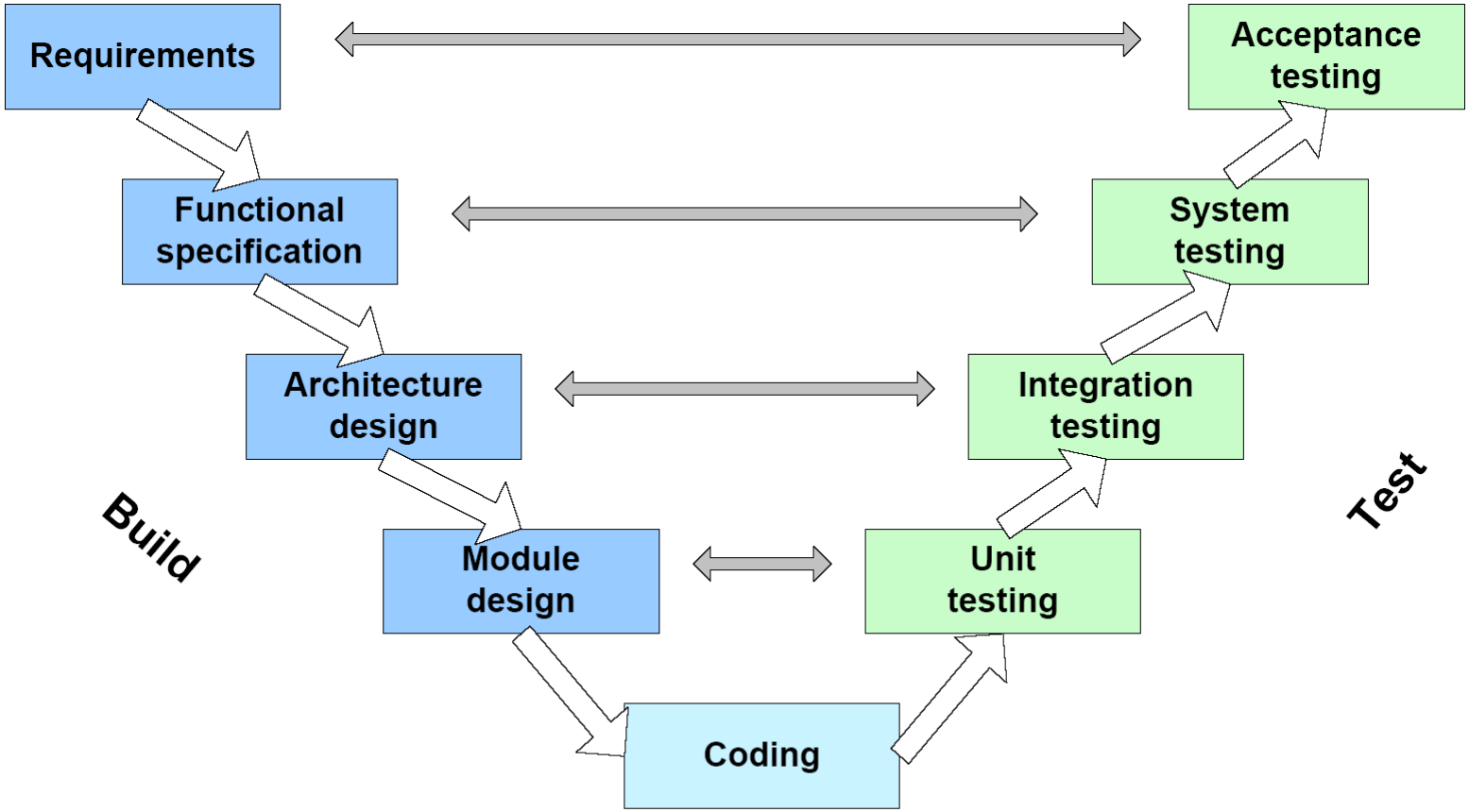
# Quality vs. Profit



# Quality vs. Functionality

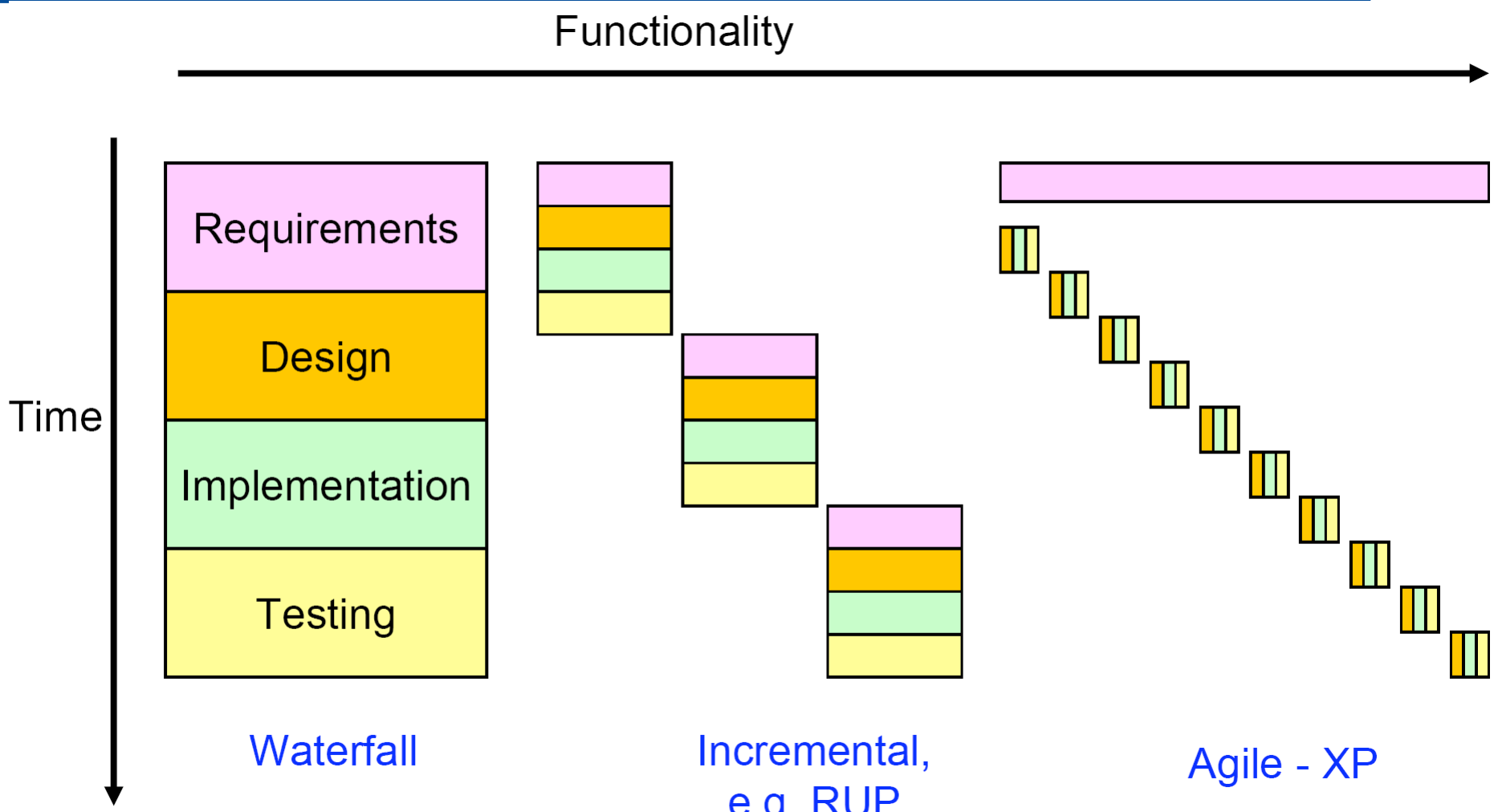


# Test process and levels – V-model

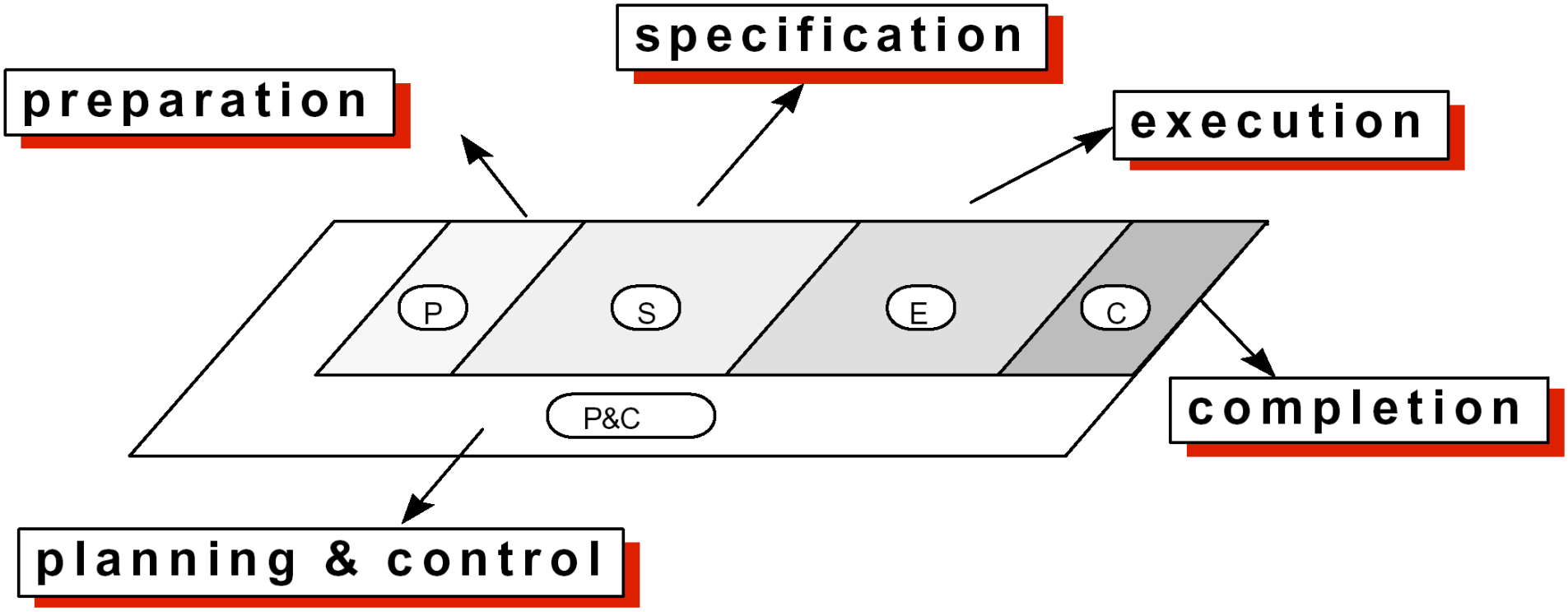



**Specifications ----> Planning ----> Testing**

# Process


Functionality
Time
Requirements
Design
Implementation
Testing
Waterfall
Incremental,
e.g. RUP
Agile - XP

# Testing process

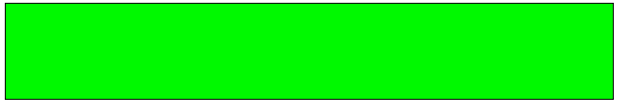


# Unit testing

- A unit = smallest possible testable software component
  - Objects
  - Procedures / functions
  - Reusable components
- Tested in isolation
- Usually done by programmer
  - A pair tester can help
- Should be
  - Planned
  - Public
- Also known as component, module or program testing

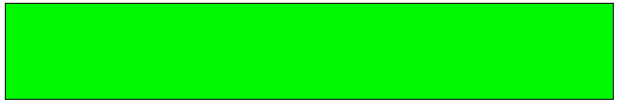
# Test-driven development

- Write a test

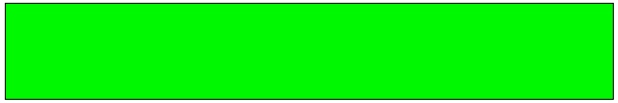


- See it fail

- Make it run



- Make it right



# Integration testing

- More than one (tested) unit
- Detecting defects
  - On the interfaces of units
  - Communication between units
- Helps assembling incrementally a whole system
- Non-functional aspects if possible
- Integration strategy: big-bang vs. incremental
- Done by developers/designers or independent testers
  - Preferably developers and testers in collaboration

# Concerns in Integration Testing

- Sequencing
- Interface testing
- Test scaffolding

# Test sequencing strategies

- Top-down
- Bottom-up
- Sandwich
- Vertical slice
- Inside-out
- Outside-in
- Combinations
- Degree of thoroughness

# Sequencing factors

- Start with system architecture
- Critical factors first
  - Complex units
  - I/O units
  - Interfaces (code from other development groups)
  - New approaches
  - New hardware
  - Time needed to test

# Sequencing factors (2)

- Rank with 3 point scale
- Decision procedure
  - Highest score first
  - Lexicographic ordering of factors
  - Subsetting of factors and summary score

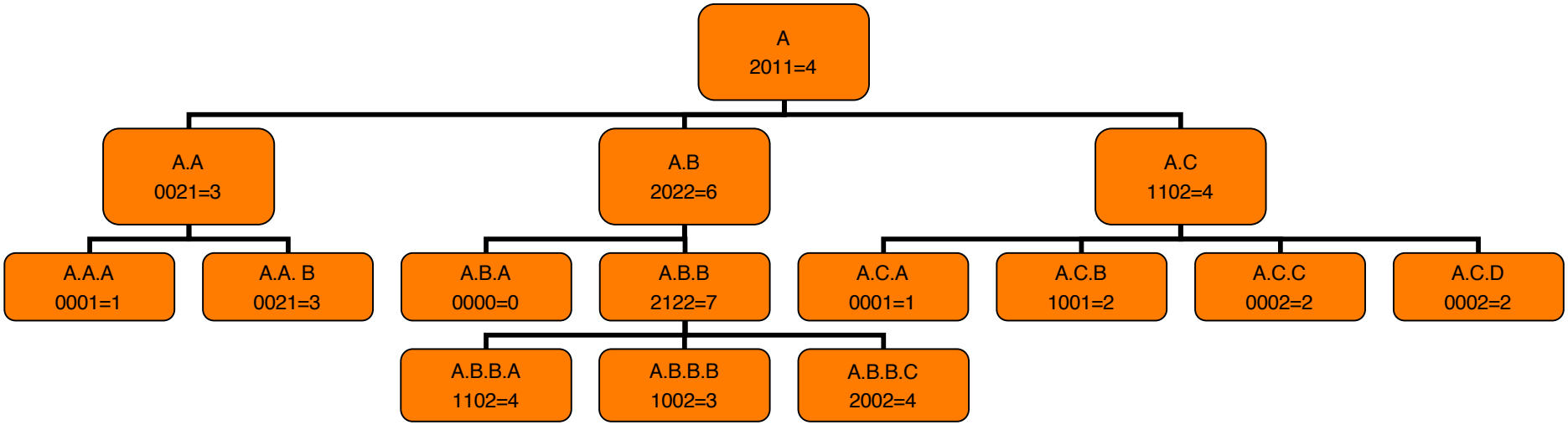
# Example: Critical Factor Rating

- Deadline
  - 0=not pressing
  - 1=important
  - 2=urgent
- Complexity of drivers/stubs
  - 0=hard to do
  - 1=medium
  - 2=easy
- Complexity of module
  - 0=low, 1=medium, 2=high

# Scoring methods - comments

- +: shows highly important modules
- -: does not identify technically preferable sequences

# Example: System module structure



# Possible sequences

| Score | Subsystem A.B | Subsystem A.C | Subsystem A.A |
|---|---|---|---|
| 7 | A.B.B | | |
| 6 | A.B | | |
| 5 | | | |
| 4 | A, A.B.B.A, A.B.B.C | A.C | |
| 3 | A.B.B.B | | A.A, A.A.B |
| 2 | | A.C.B, A.C.C, A.C..D | |
| 1 | | A.C.A | A.A.A |
| 0 | A.B.A | | |

# Example: Lexicographic Ordering

- Deadline < Cmpl(driver) < Cmpl(stubs) < Cmpl(module)

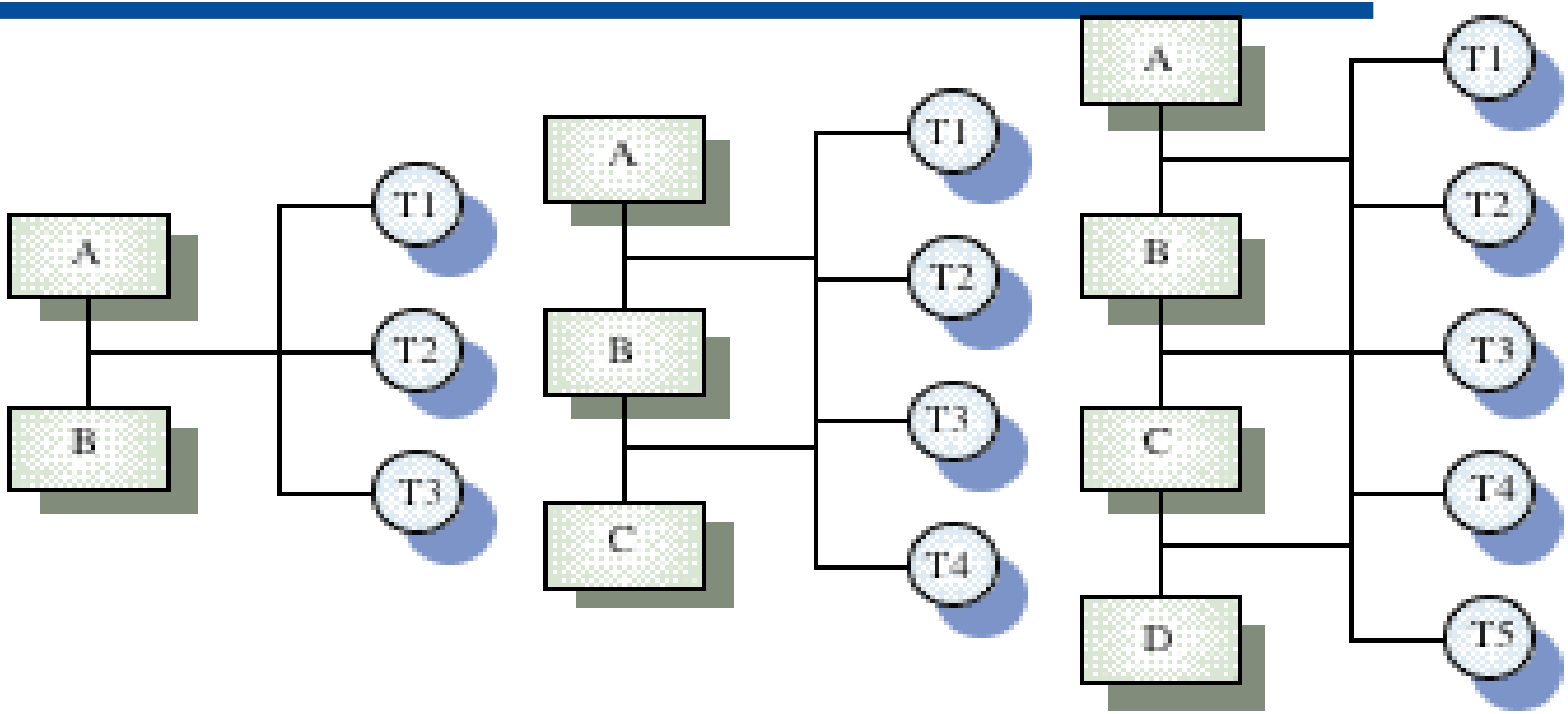
- +: not all factors have same weight
- +: more sensitive to more important factors
- +: less tradeoff
- -: may need more tradeoff than provided

# Possible sequences

| Score | Subsystem A.B | Subsystem A.C | Subsystem A.A |
|---|---|---|---|
| 2122 | A.B.B | | |
| 2022 | A.B | | |
| 2011 | A | | |
| 2002 | A.B.B.C | | |
| 1102 | A.B.B.A | A.C | |
| 1002 | A.B.B.B | | |
| 1001 | | A.C.B | |
| 0021 | | | A.A, A.A.B |
| 0002 | | A.C.C, A.C.D | |
| 0001 | | A.C.A | A.A.A |
| 0000 | A.B.A | | |

Advanced Software Engineering:
Software Testing

# Incremental integration testing

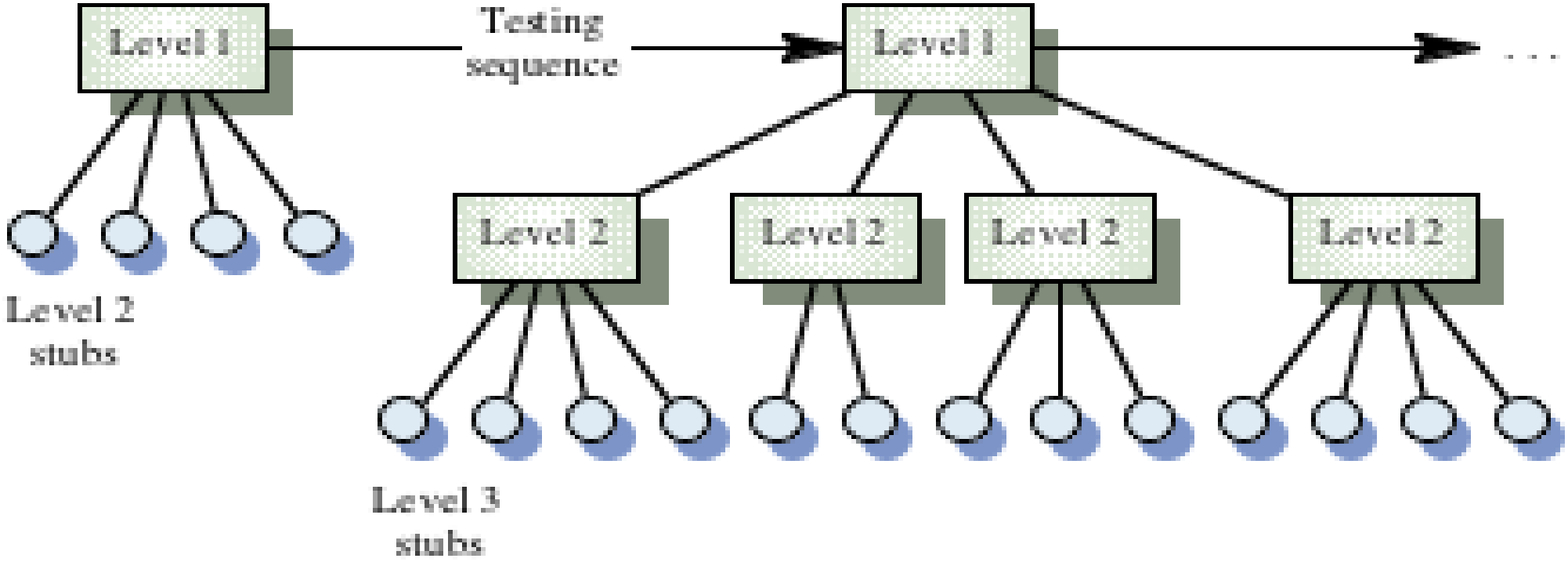



Test sequence 1

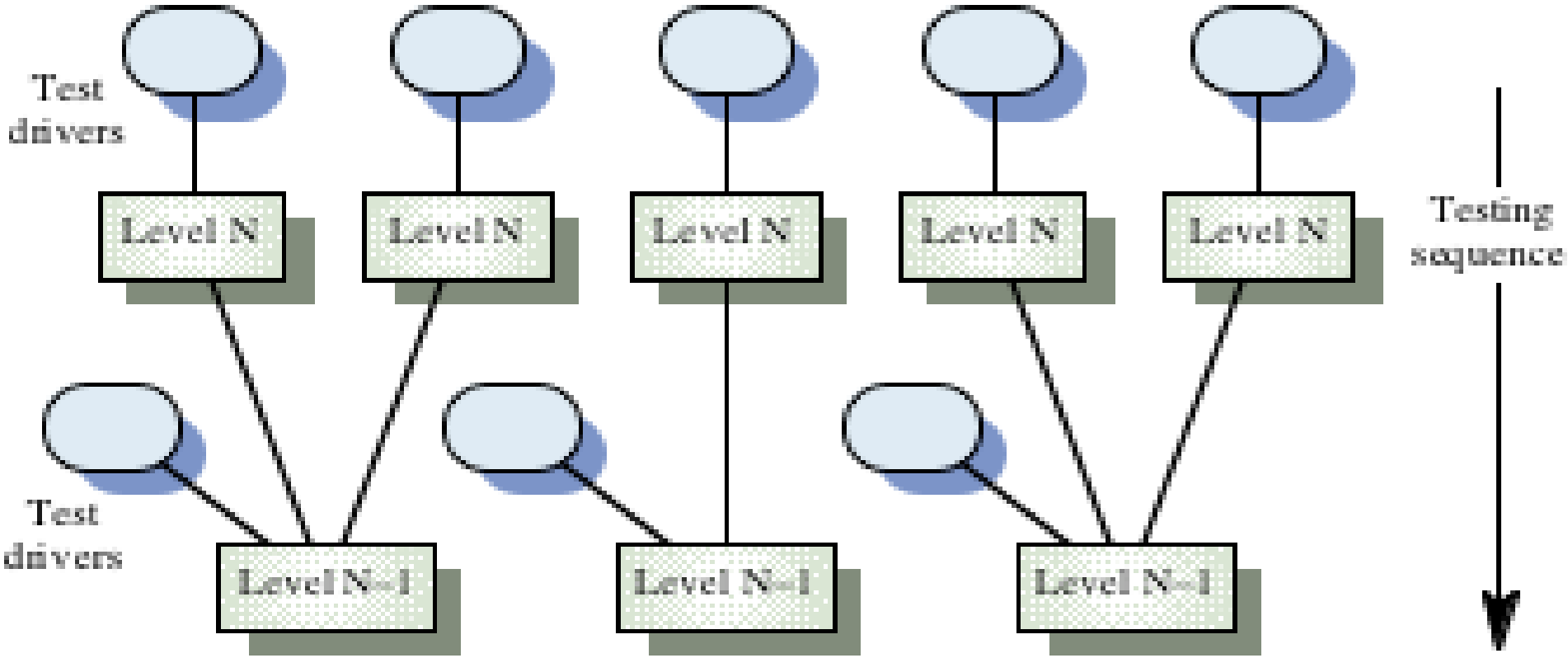
Test sequence 2

Test sequence 3

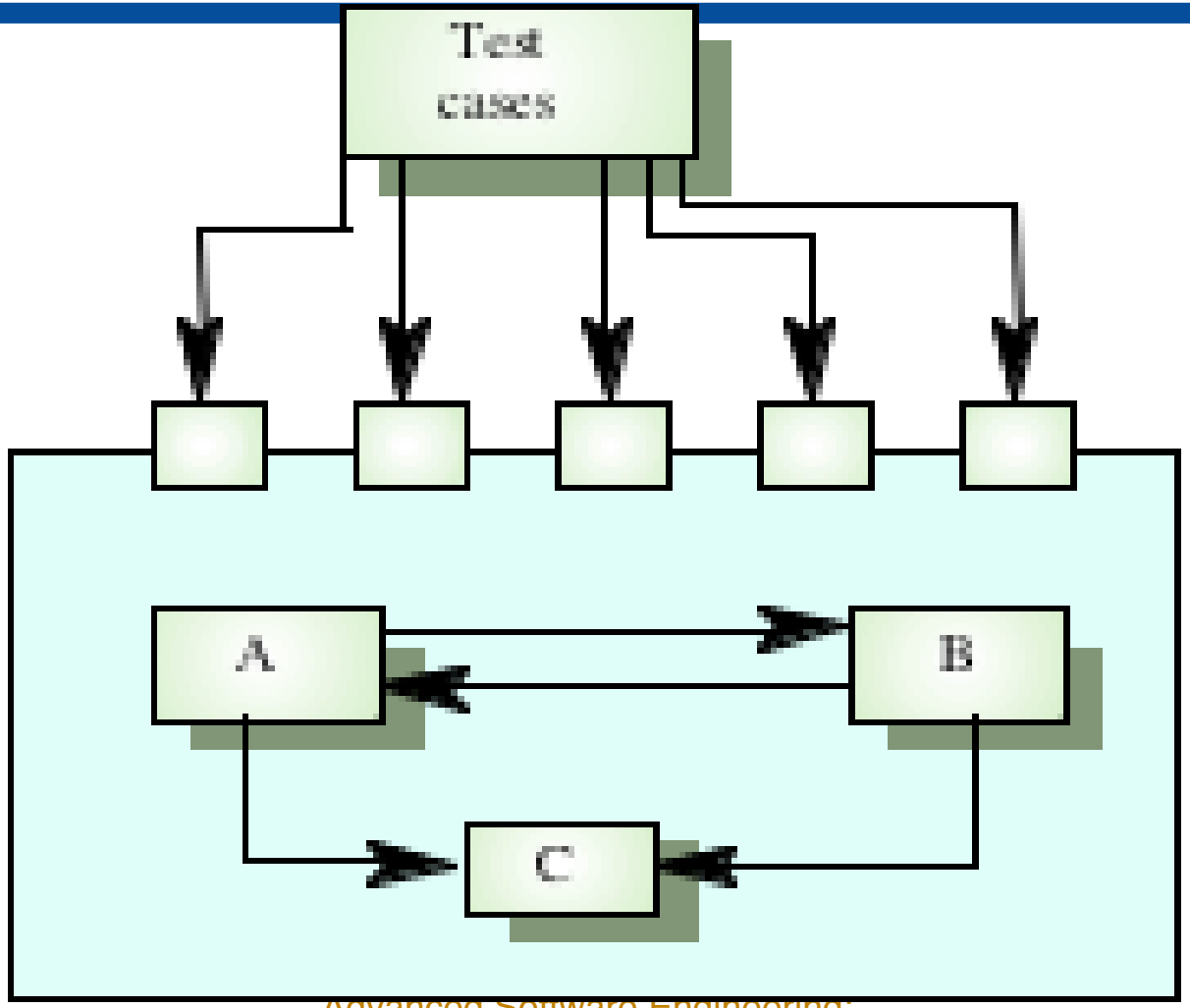
# Top-down testing



# Bottom-up testing



# Interface testing



# Interfaces

- Parameters, returned results
- Global variables
- Imported/exported properties
- Imported/exported methods
- Imported/exported assumptions, types

# Interface parameters

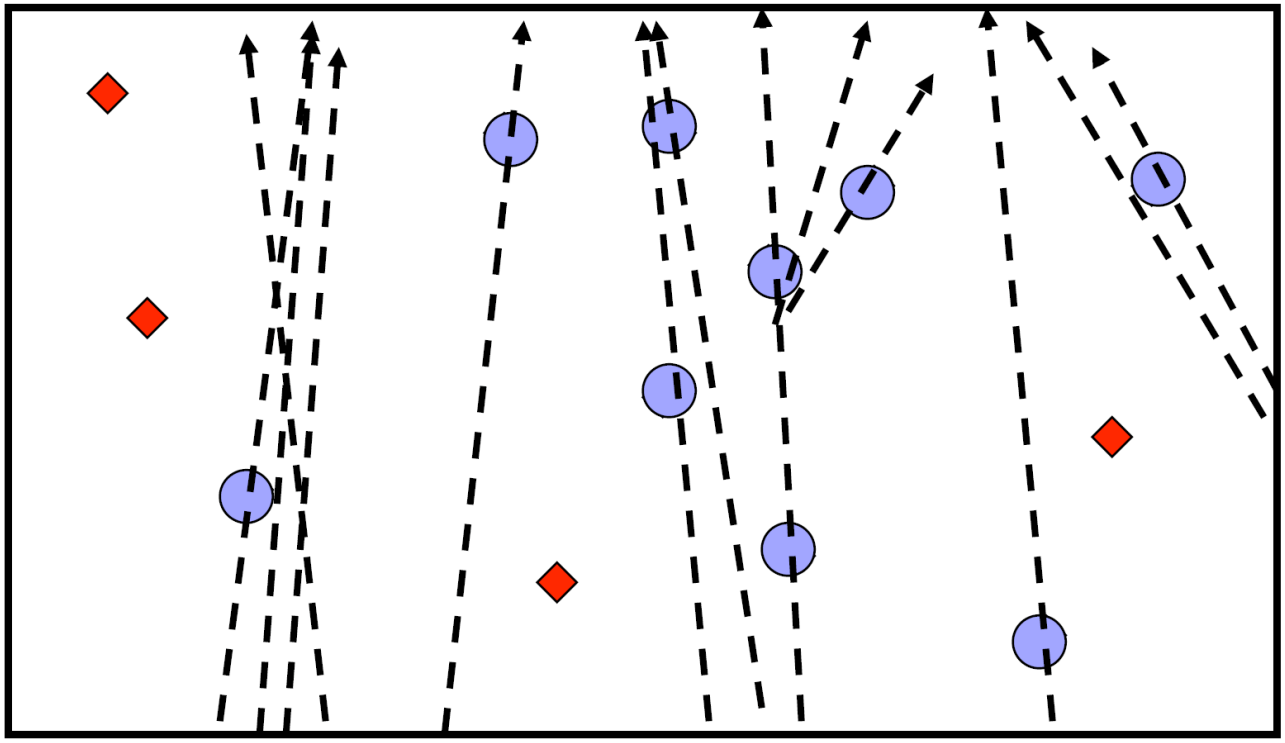
- Values passed correctly?
- Types passed/converted correctly?
- Calling/called assumptions consistent?
- Arguments in correct sequence?
- Array arguments dimensioned correctly?
- Timing assumptions correct
- For multiple execution: proper number and sequence?
- For objects: inheritance, pointers correct?

# System testing

- Testing the system as a whole
- Functional
  - Functional requirements and requirements-based testing
  - Business process based testing
- Non-functional
  - Performance, stress, configuration, security, ...
  - As important as functional requirements
  - Often poorly specified
  - Must be tested
- Often done by independent test group
  - Collaborating developers and testers

# Regression testing

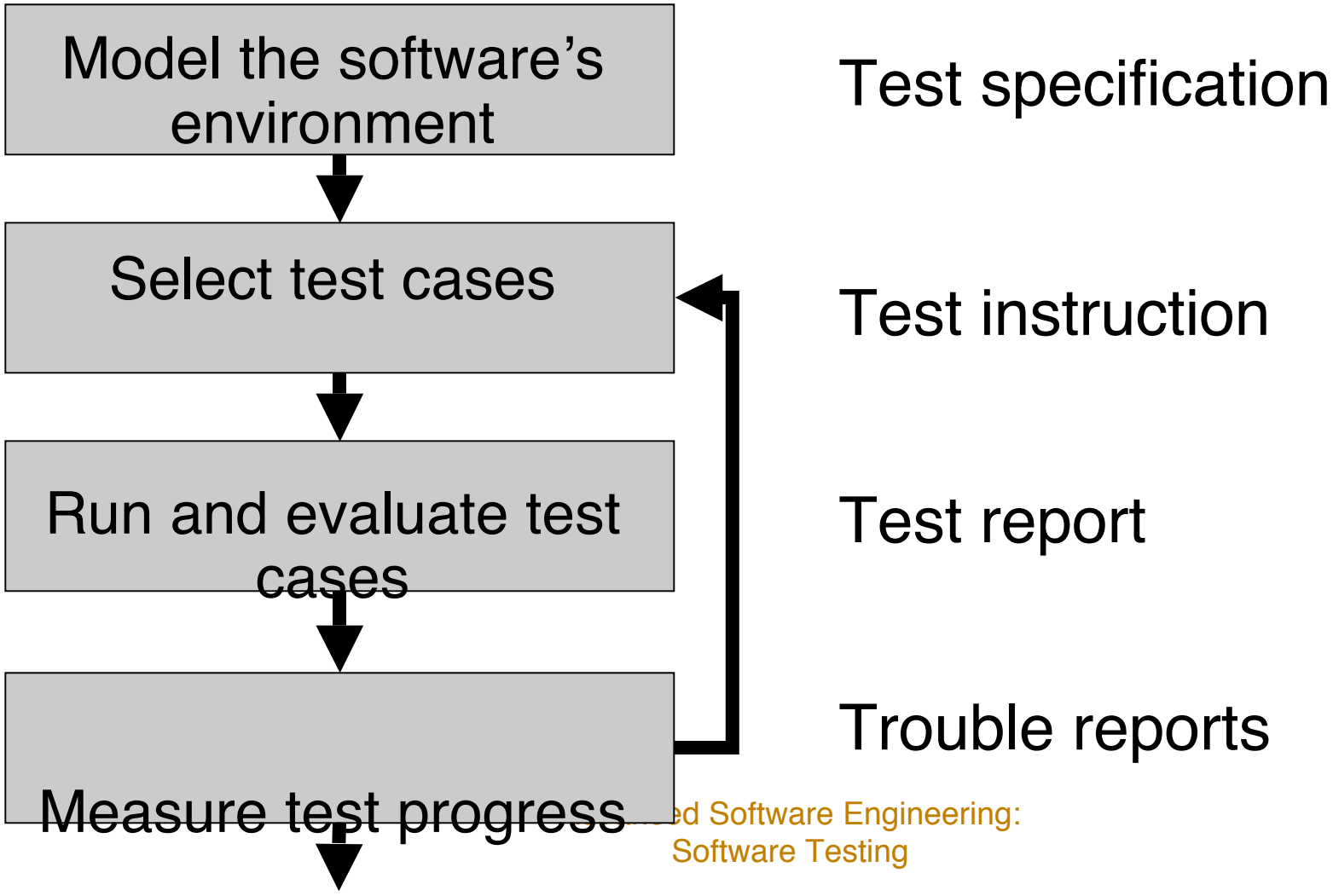
- Important? When?
- All test cases, some test cases (which test cases)?



# Acceptance testing (_, _)

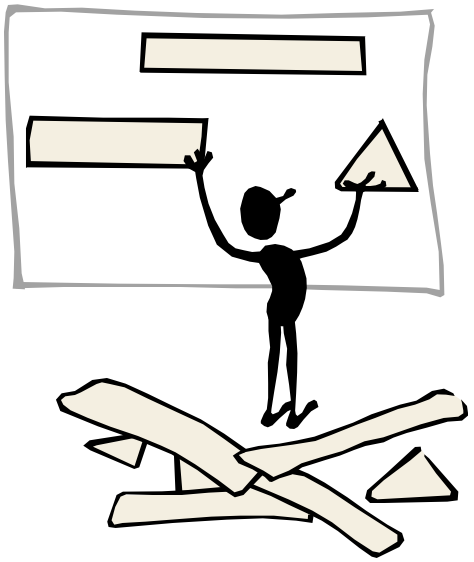
- Final stage of validation
  - Customer (user) should perform or be closely involved
  - Customer can perform any test they wish
  - Final user sign-off
- Approach
  - Mixture of scripted and unscripted testing
  - Performed in real operation environment
- No new failures allowed
- Project
  - Contract acceptance
  - Customer viewpoint
  - Validates that the right system was built
- Product
  - Final checks on releases
  - User viewpoint throughout the development; validation

# Test process / Document



# Model the software's environment

- Test Specification
  - Definition
  - Textual
  - Model-based
  - Frequencies of use
  - Priorities



# Select test cases

- Manual, ad hoc
- Manual, method based
- Semi-automatic
- Automatic

# Purpose of test case (planning)

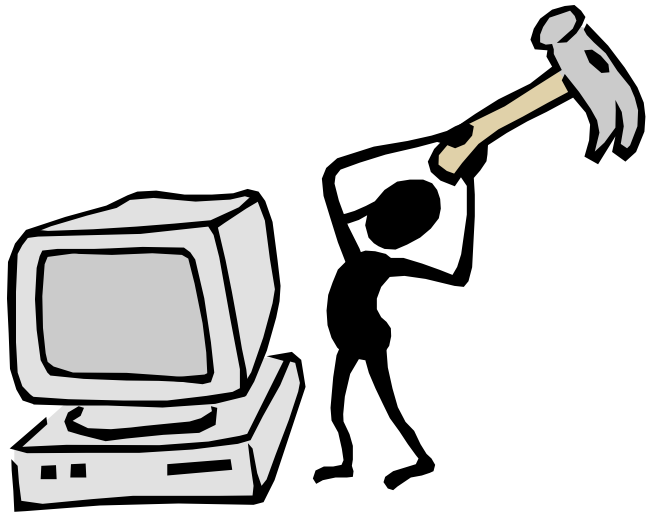
- Organization
  - All testers and other project team members can review and use them effectively
- Repeatability
  - Know what test cases were last run and how so that you could repeat the same tests
- Tracking
  - What requirements or features are tested?
  - Tracking information's value depends on the quality of the test cases
- Proof of testing
  - Confidence (quality)
  - Detect failures

# Attributes of test cases

- Power – When a problem exists, the test will reveal it
- Valid – When the test reveals a problem, it is actually a problem
- Value – It reveals things you want to know about
- Credible – It tests the right thing
- Representative – Detects failures with high probability to manifest
- Maintainable – Easy to revise when the product changes
- Repeatable – Easy and inexpensive to reuse
- Unique – It covers new parts of the software
- Easy to evaluate
- ...

# Run and evaluate test cases

- Running is (relatively) easy
- Evaluation is hard



- Exemption: automated regression test

# Measurements

- How much is tested?
- Which is the quality of the product?
- How many faults / failures are removed? Remaining?
- When to stop testing?
- More in lecture 6 (Metrics)

# Framework / Goals and Policies

- Documentation
- Testware
- Information sources
- Standards
- Planning
- Configuration management
- Measurements
- Tools

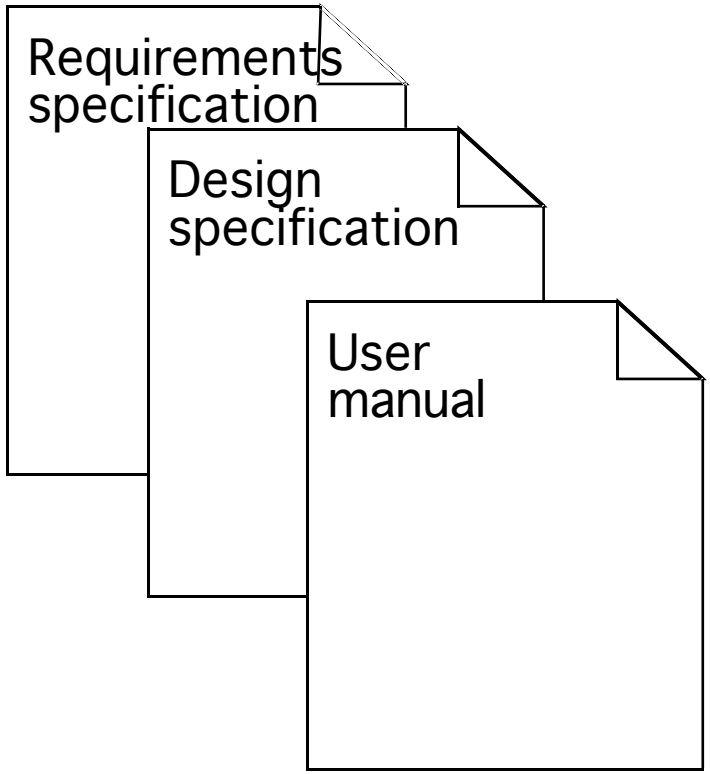
- Goals
  - Business goals – increase market share
  - Technical – reduce number of failures
  - Business/technical – reduce support calls
  - Political – increase number of minorities in management positions
- Quantitative / qualitative goals
- Goals translate into policies

# Documentation / Testware

- Plan
- Checklists
- Statistics
- Test data
- ...
- Test cases
- Test programs
- Test stubs
- Test drivers
- ...

# Information sources

- Requirements specification
- Design specification
- User manuals
- Prototypes
- Domain knowledge



# Standards

- *IEEE 829-1998*

IEEE Standard for Software Test Documentation

- *IEEE 1012-1998*

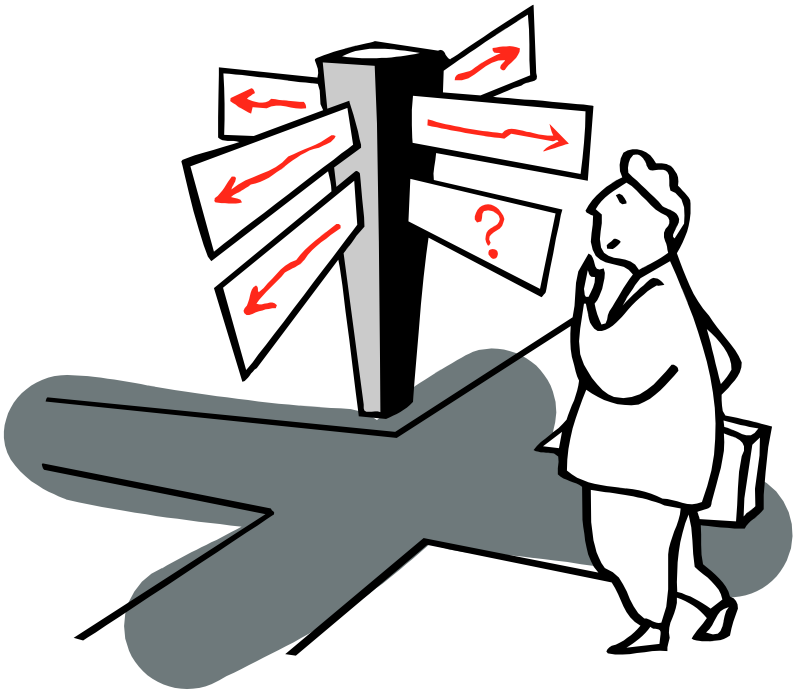
IEEE Standard for Software Verification and Validation

- *IEEE 1008-1993*

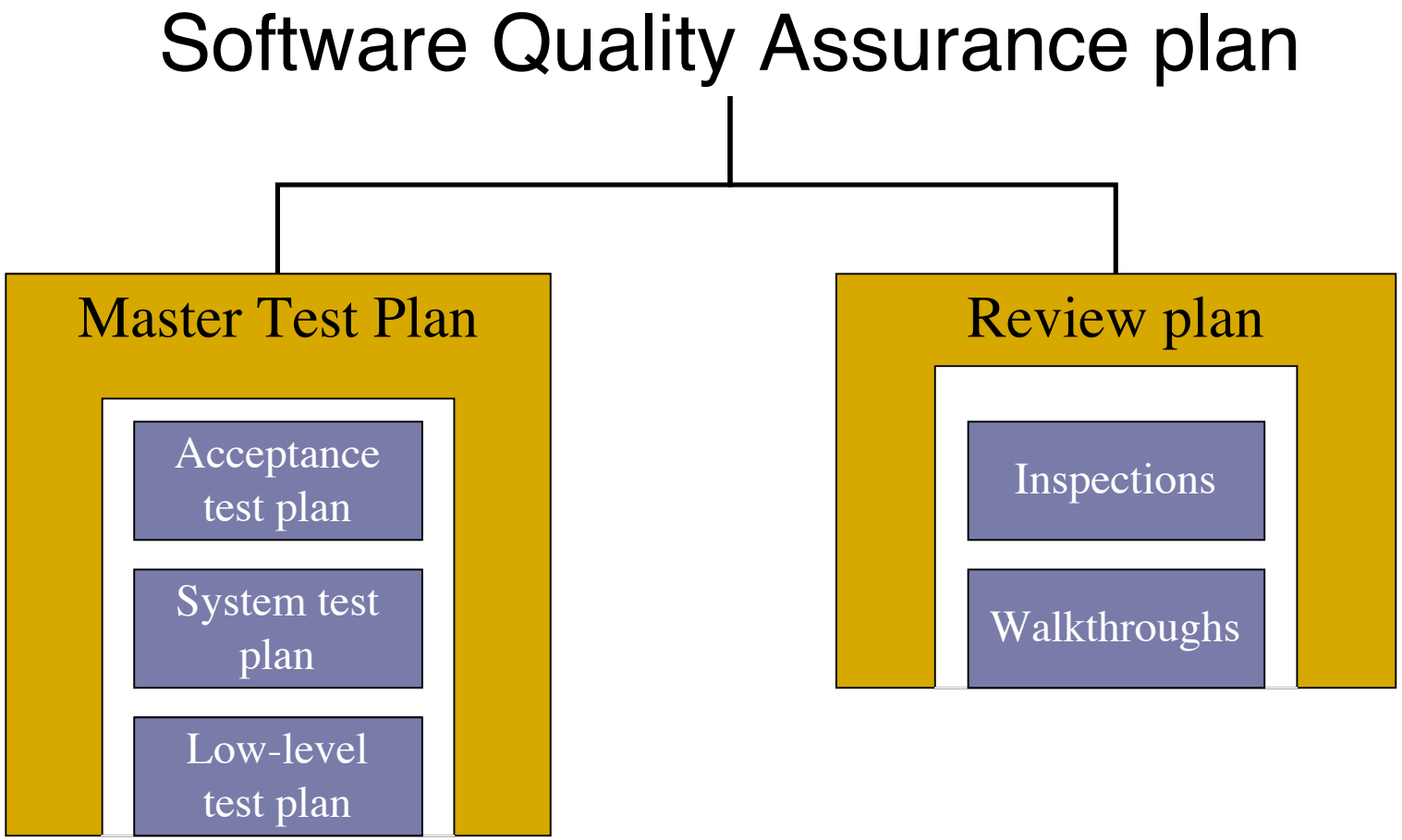
IEEE Standard for Software Unit Testing

# Test planning

- What to test
- Who will test
- When to test
- How to test
- When to stop



# Test Plan – What should be included?



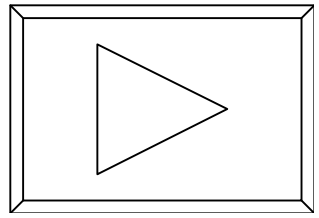
# Master Test Plan Structure

- The formulation of the assignment
- Test strategy
- Threats, risks and measurements
- Organization
- Infrastructure (tools, environment)
- Global planning (schedules)

# Test plan according to IEEEStd 829-1998 (Appendix II)

a) Test plan identifier
b) Introduction
c) Test items
d) Features to be tested
e) Features not to be tested
f) Approach
g) Item pass/fail criteria
h) Suspension criteria and resumption requirements
i) Test deliverables
j) Testing tasks
k) Environmental needs
l) Responsibilities
m) Staffing and training needs
n) Schedule
o) Risks and contingencies
p) Approvals





# Test plan

a) Test plan identifier

b) Introduction

- Product to be tested, objectives, scope of the test plan
- Software items and features to be tested
- References to project authorization, project plan, QA plan, CM plan, relevant policies & standards

c) Test items

- Test items including version/revision level
- Items include end-user documentation
- Defect fixes
- How transmitted to testing
- References to software documentation

# Test plan

d) Features to be tested
- Identify test design / specification techniques
- Reference requirements or other specs

e) Features not to be tested
- Deferred features, environment combinations, …
- Reasons for exclusion

f) Approach
- How you are going to test this system
  - Activities, techniques and tools
- Detailed enough to estimate
- Completion criteria (e.g. coverage, reliability)
- Identify constraints (environment, staff, deadlines)

# Test plan

g) Item pass/fail criteria
- What constitutes success of the testing
- Coverage, failure count, failure rate, number of executed tests, …
- Is NOT product release criteria

h) Suspension and resumption criteria
- For all or parts of testing activities
- Which activities must be repeated on resumption

i) Test deliverables
- Test plan
- Test design specification, Test case specification
- Test procedure specification, Test item transmittal report
- Test logs, Test incident reports, Test summary reports

# Test plan

j) Testing tasks
- Including inter-task dependencies & special skills
- Estimates

k) Environment
- Physical, hardware, software, tools
- Mode of usage, security, office space
- Test environment set-up

l) Responsibilities
- To manage, design, prepare, execute, witness, check, resolve issues, providing environment, providing the software to test

m) Staffing and Training needs

# Test plan

n) Schedule
- Test milestones in project schedule
- Item transmittal milestones
- Additional test milestones (environment ready)
- What resources are needed when

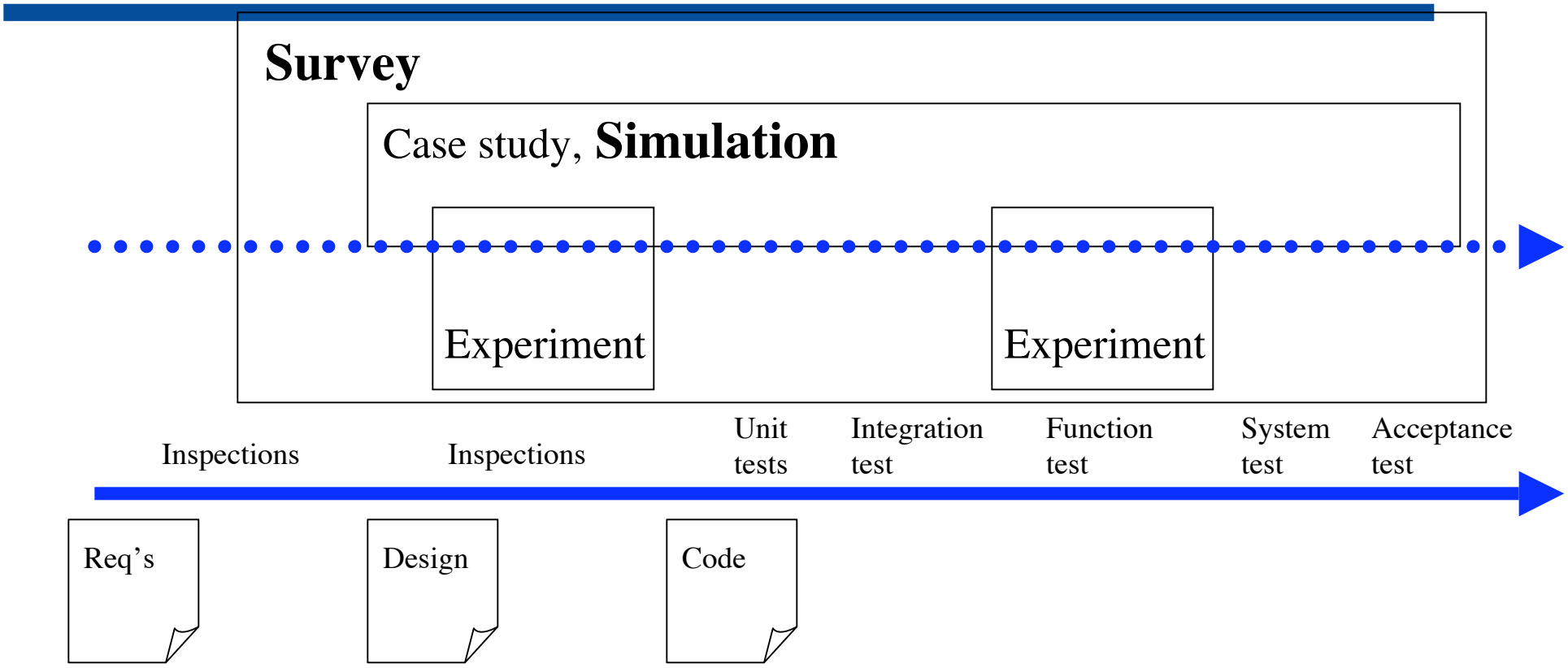
o) Risks and Contingencies
- Testing project risks
- Contingency and mitigation plan for each identified risk

p) Approvals
- Names and when approved

# Test plan quality criteria

- Usefulness – Will the test plan effectively serve its intended functions?
- Accuracy – Is the test plan document accurate with respect to any statements of fact?
- Efficiency – Does it make efficient use of available resources?
- Adaptability – Will it tolerate reasonable change and unpredictability in the project?
- Clarity – Is the test plan self-consistent and sufficiently unambiguous?
- Usability – Is the test plan document concise, maintainable, and helpfully organized?
- Compliance – Does the test plan meet externally imposed requirements?
- Foundation – Is the test plan the product of an effective test planning process?

# Aspects of the V&V process



# Process simulation

- Aid to decision making
  - Strategic management
    - Should work be distributed across sites or centralized?
  - Planning
    - Forecast staffing levels needed across time
  - Process improvement and technology adoption
    - What impact will the introduction of inspections have?
  - Understanding
  - Training and learning
- Motivation
  - Be able to model large-scale, real-world contexts

# Model

*No model is perfect,*
*but some models are useful*

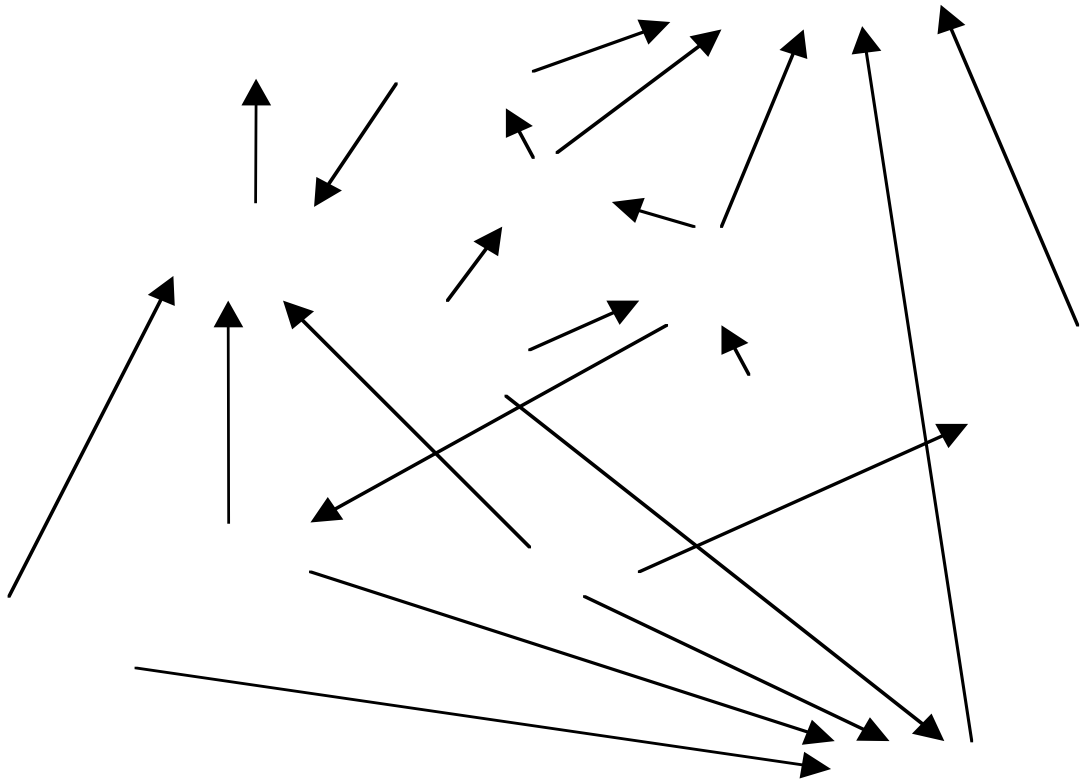
- Dynamic behaviour
  - System behaviour can change over time
  - Dynamic, discrete-event, etc.
- Feedback mechanisms
  - Behaviour and decisions made at one point in the process impact others in complex or indirect ways
- Model scope
  - A portion of the life cycle
  - A development project
  - Multiple, concurrent projects
- Input parameters
  - Amount of incoming work
  - Defect detection efficiency
  - Effort for code rework
- Result variables
  - Effort/cost
  - Cycle-time… etc

# Quality factors

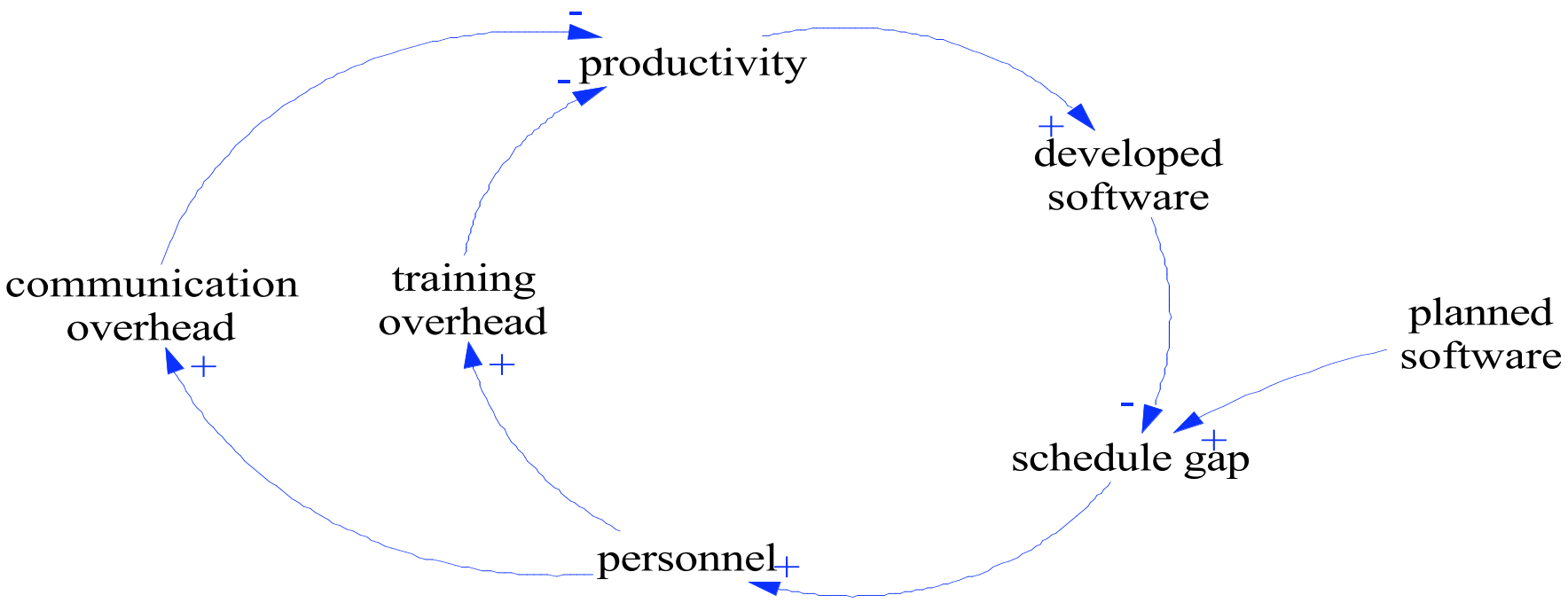
1. Number of people in the overall project
2. Number of people in the object or team
3. Personnel education
4. Personnel experience
5. Personnel salary
6. Staff turnover
7. Communication level
8. Geographical separation of the team
9. Software and hardware resources
10. Environment, for example temperature, light and ergonomics
11. Amount of overtime and workload
12. Schedule pressure
13. Budget pressure
14. Rate of requirement change
15. Amount of program documentation
16. Level of reusable artefacts, for example code and documentation
17. Level of structure in the project organisation
18. Standards that will be adhered to, for example ISO and IEEE
19. Software size and complexity
20. Testing and correcting environment and tools
21. Requirement specification accuracy
22. Amount of review

# Influence diagram

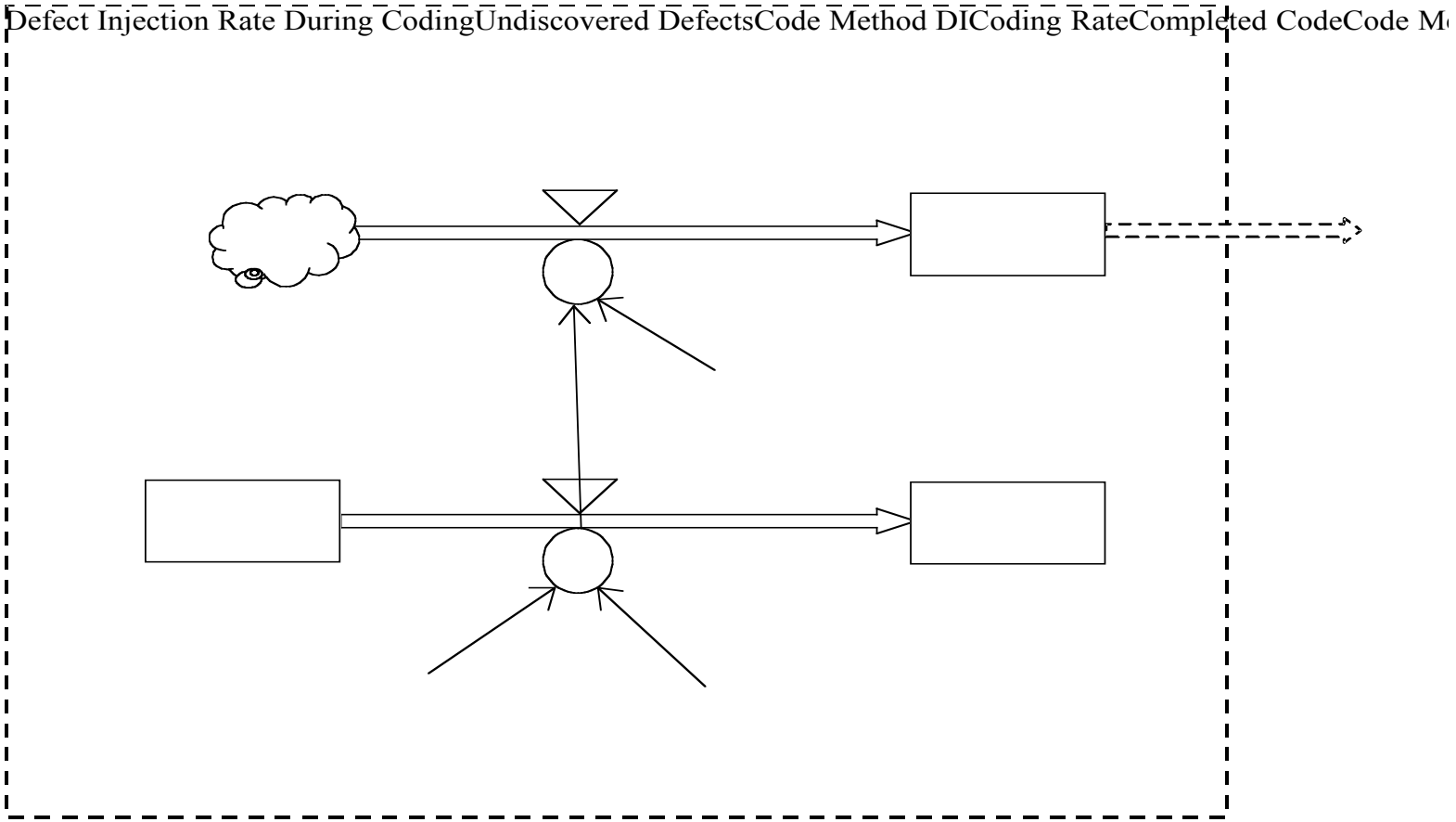
Amount of NewMarketRequirementsCommunicationLevelInadequateRequirementsTime in RequirementPhaseSpecificationAccur



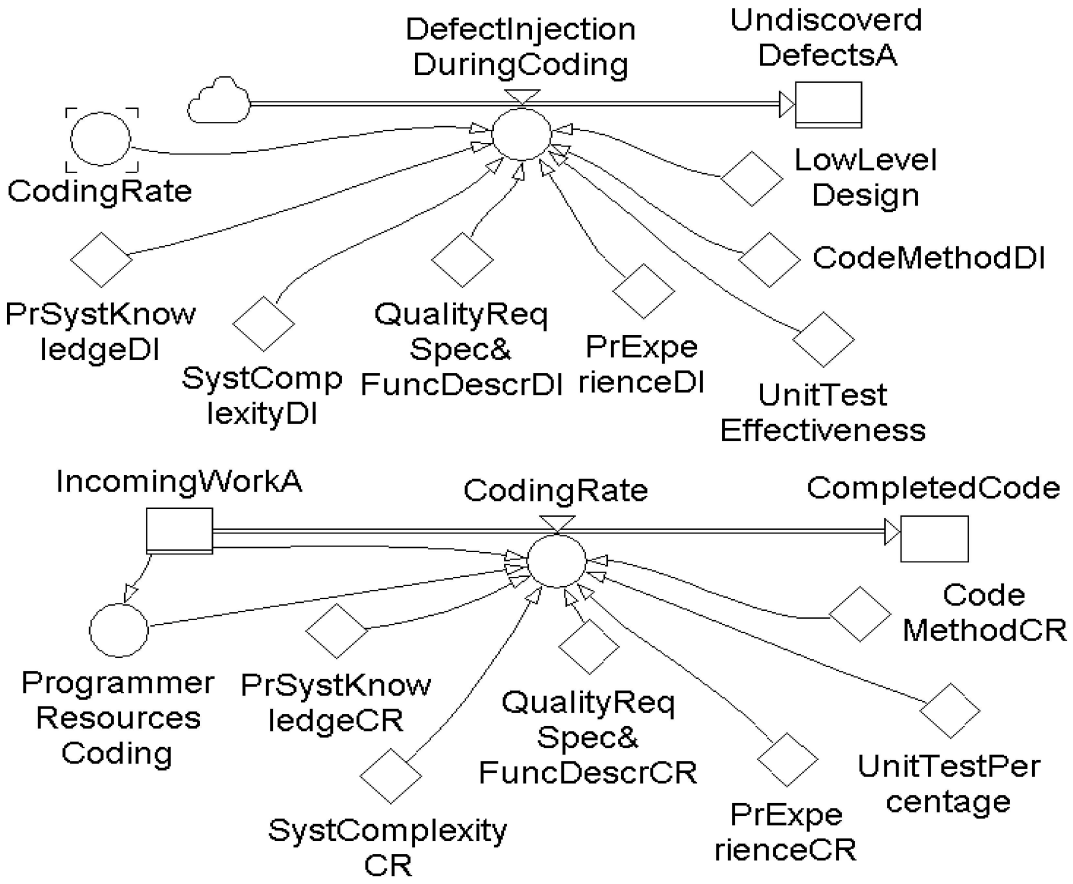
# Causal loop diagram



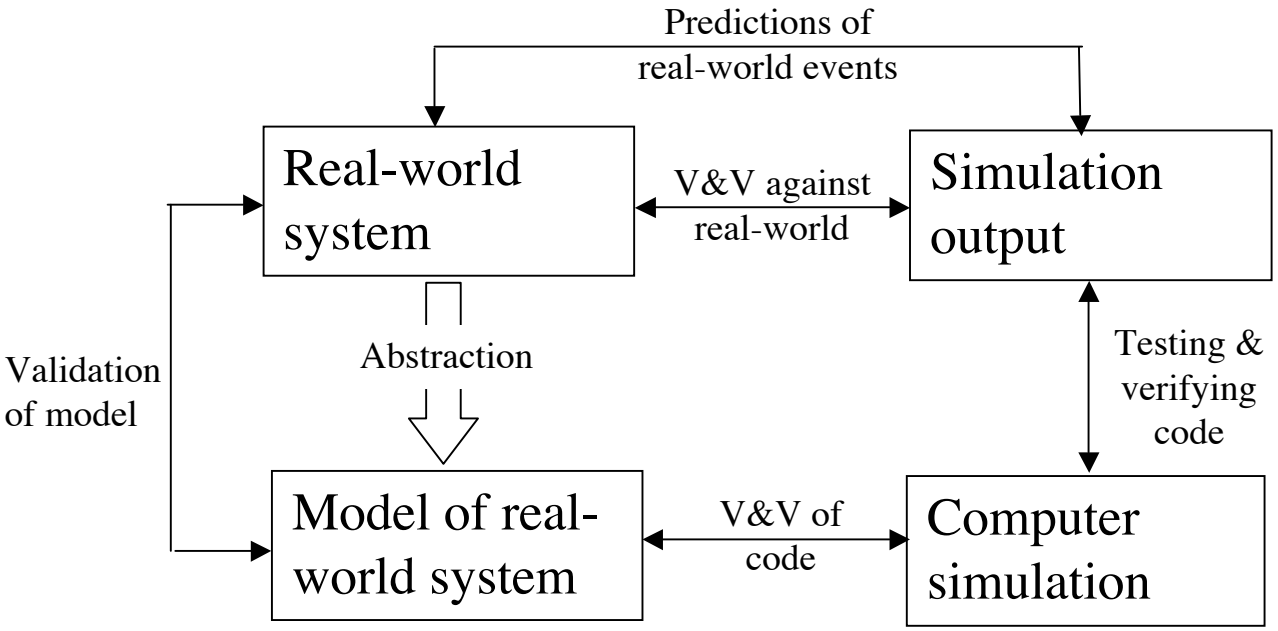
# Template model



# Extended model



# Calibration & Validation




- Model assessment
  - Is the model implementation error-free?
  - Do the model solve the end-user's problem?
- Calibration
  - Real-world data
  - Sensitivity analysis

# Survey

- Size
  - 6-200 developers
- Customer
  - Private/Business/Internal
- Business Model
  - Contract/Market
- Products
  - Image processing
  - Communication
  - CASE tools
  - Support systems
- Product Value
  - Functional/Non-functional

# State of the practice

- Ad hoc solutions
  - Testing is introduced late
  - Test cases are based on previous defects
  - Testing is a low-status job. (Is changing)
  - Level of automation is low
  - Use of quantitative methods is scarce