Testing Concepts

Brian Nielsen

{bnielsen@cs.aau.dk}

BRICS
Basic Research
in Computer Science

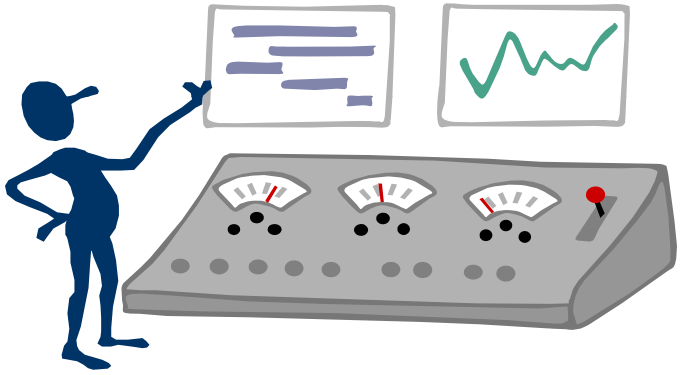
Testing

Testing:

- to check the **quality** (functionality, reliability, performance, …) of an (software) object

  -by performing experiments

  -in a controlled way



- In avg. 10-20 errors per 1000 LOC
- 30-50 % of development time and cost in embedded software

- To find errors
- To determine risk of release

Risk

- *Make best possible use of resources by identifying and prioritizing quality aspects and subsystems*
  - Higher risk ⇒ more testing
  - No risk ⇒ no testing
- **Risk** = chance of failure × damage

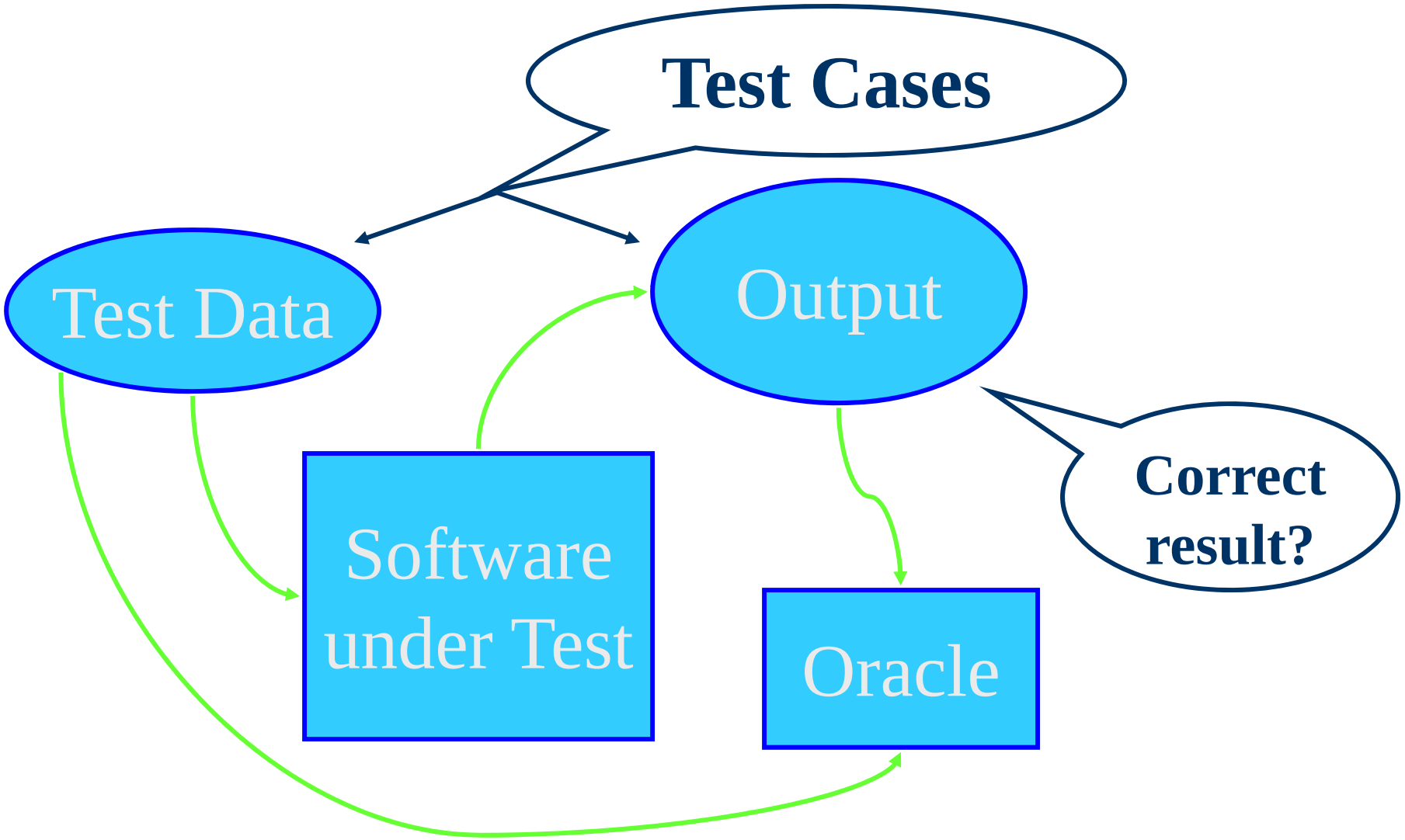
  - Use frequency
  - Chance of error being present
    - Complexity
    - New tools/techniques
    - Inexperienced developers

  - Cost of repair
  - Loss of market share
  - Legal claim

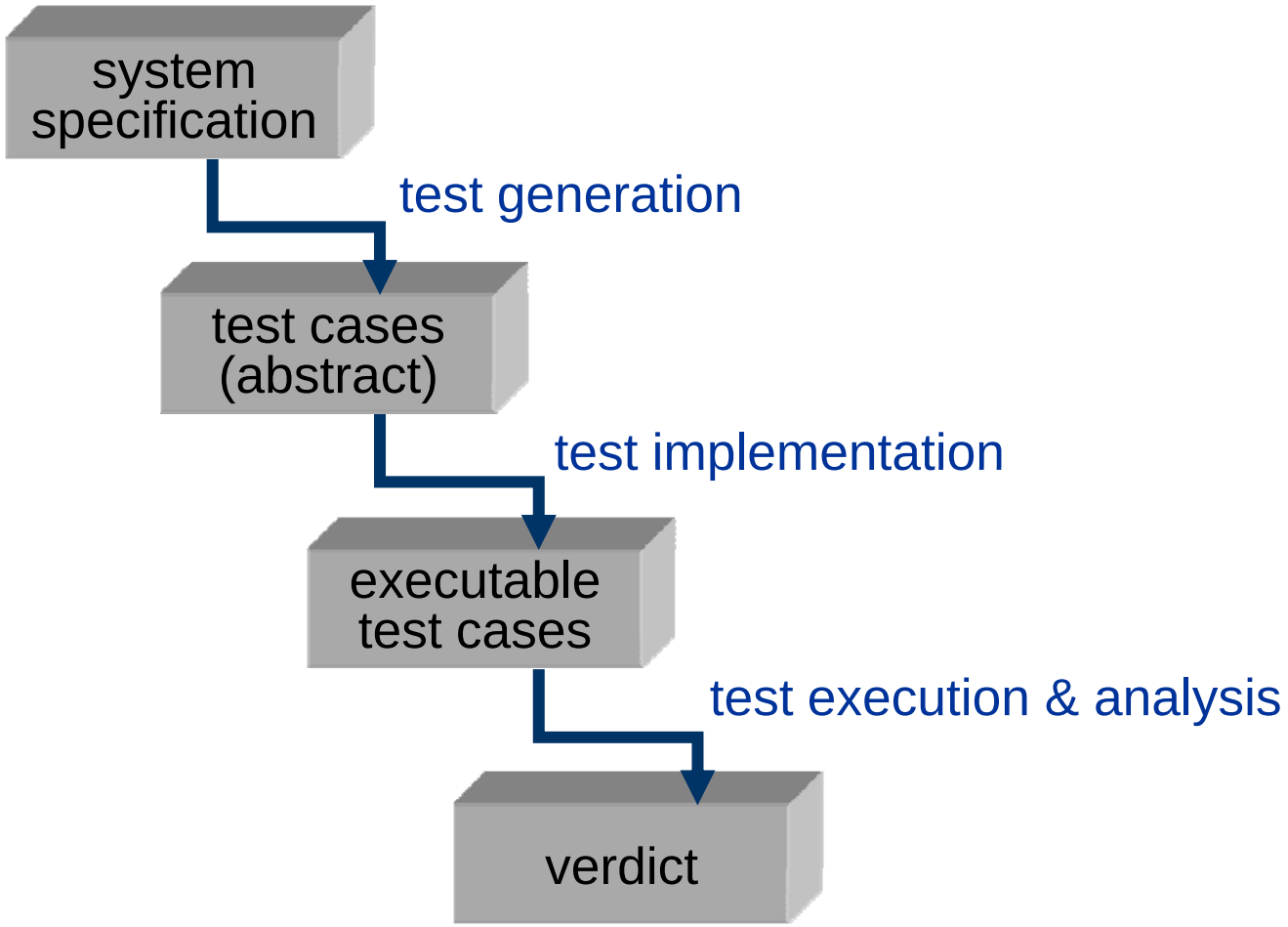
Testing

- ***Dynamic testing*** is the process of executing a program or system with the intent of finding error
  (Glenford Meyers' definition)
- ***Static testing*** is any activity that aims at finding defects by inspecting, reviewing, walking through, and analyzing any static component of the software (code, documents, and models)
- ***Debugging*** is an ad hoc activity performed by individual developers to find and remove bugs from a program.
- ***Testing*** is a *planned* activity

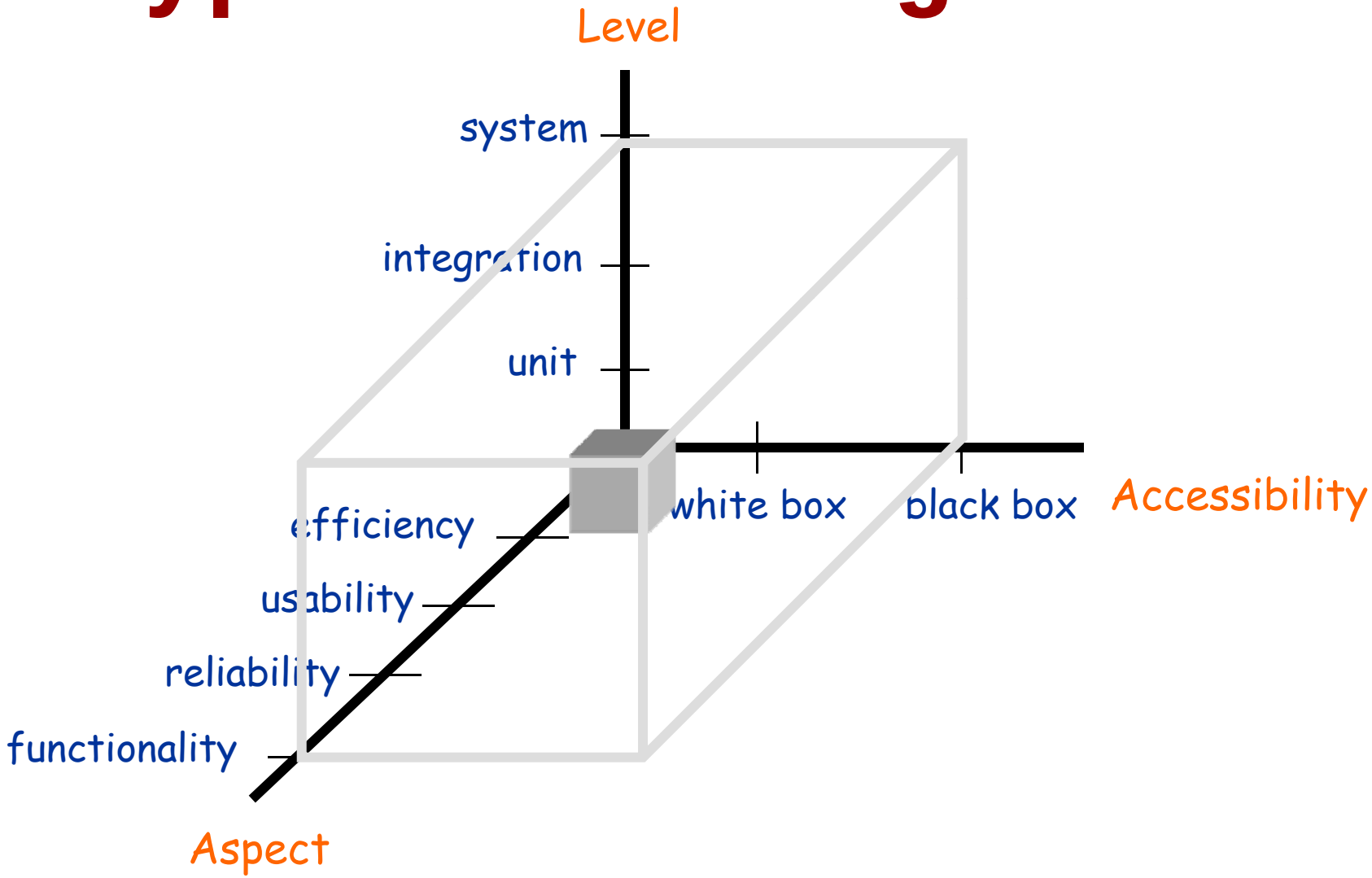
What is a Test?



Testing Process



Types of Testing



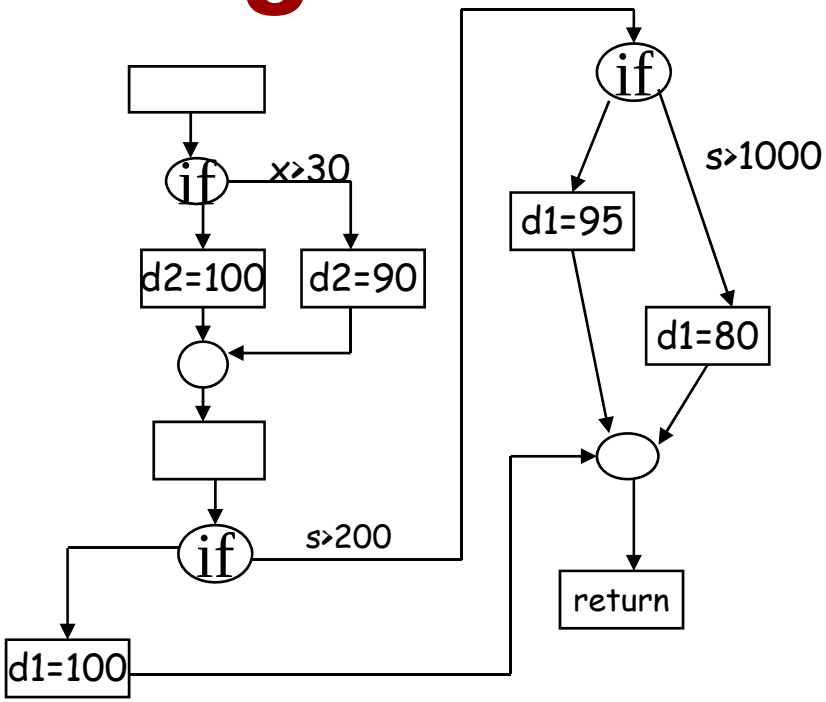
CISS

Quality-Characteristics (ISO-9126)

- Functionality ⇒ functional testing
  - Suitability, accuracy, security, compliance, interoperability
- Reliability ⇒ reliability testing
  - maturity, fault tolerance, recoverability
- Usability ⇒ usability testing
  - understandability, learnability, operability
- Efficiency ⇒ performance testing
  - time behaviour, resource utilization
- Maintainability ⇒ maintainability testing ??
  - Analysability, changeability, stability, testability
- Portability ⇒ portability testing ?
  - Adaptability, installability, conformance, replaceability

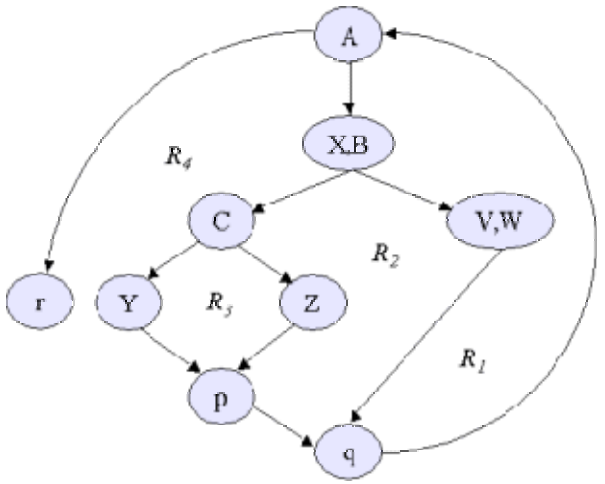
Whitebox Testing

```
int invoice (int x, int y) {
  int d1, d2, s;
  if (x<=30) d2=100;
  else d2=90;
  s=5*x + 10 *y;
  if (s<=200) d1=100;
  else if (s<=1000) d1 = 95;
          else d1 = 80;
  return (s*d1*d2/10000);
}
```



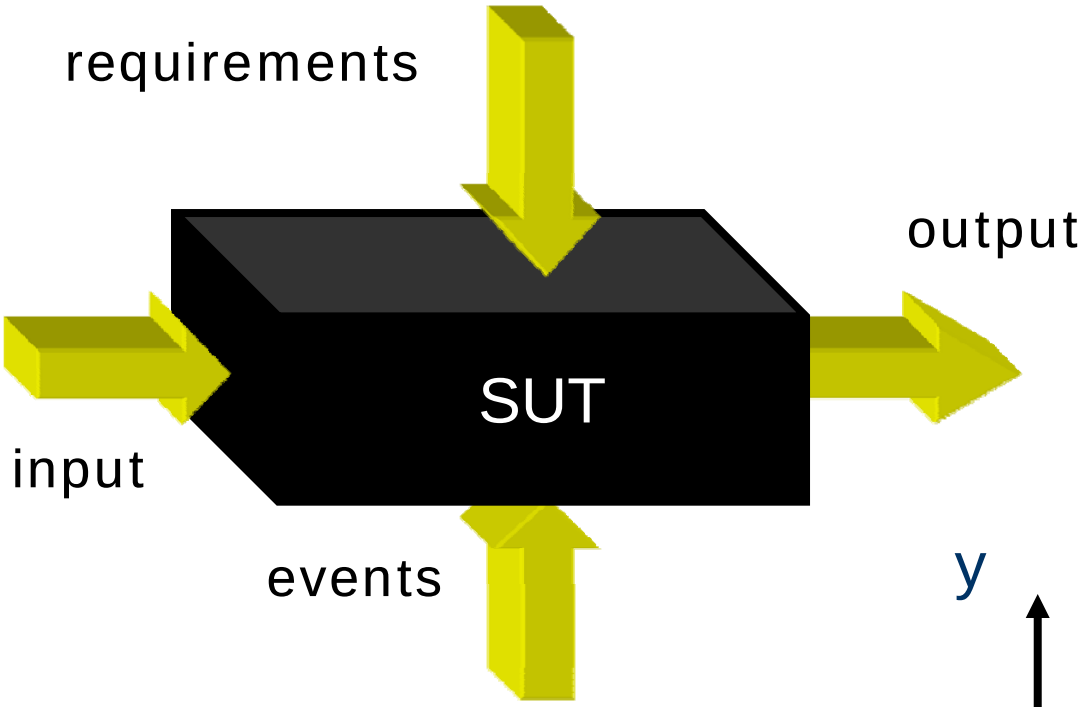
Test Cases

Test Data	Expected Output
X=5 Y=5	75
X=31 Y=10	229.5
X=30 Y=100	977.5

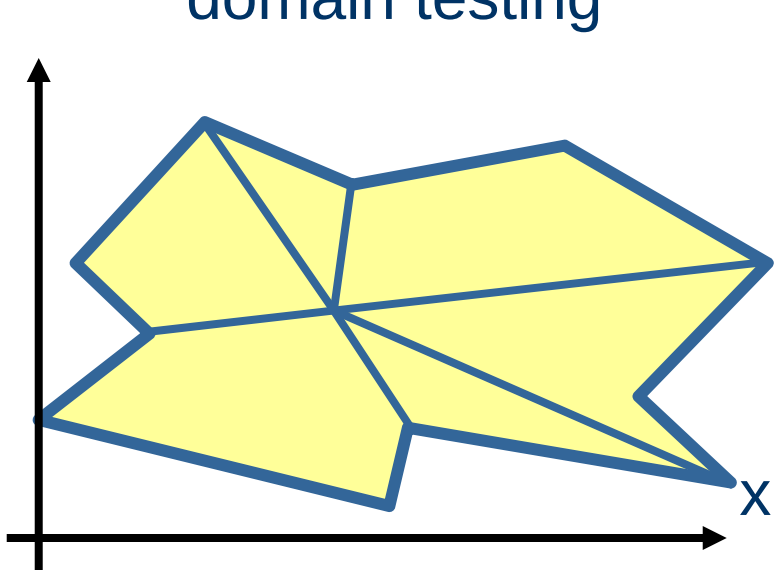


CISS

Blackbox testing

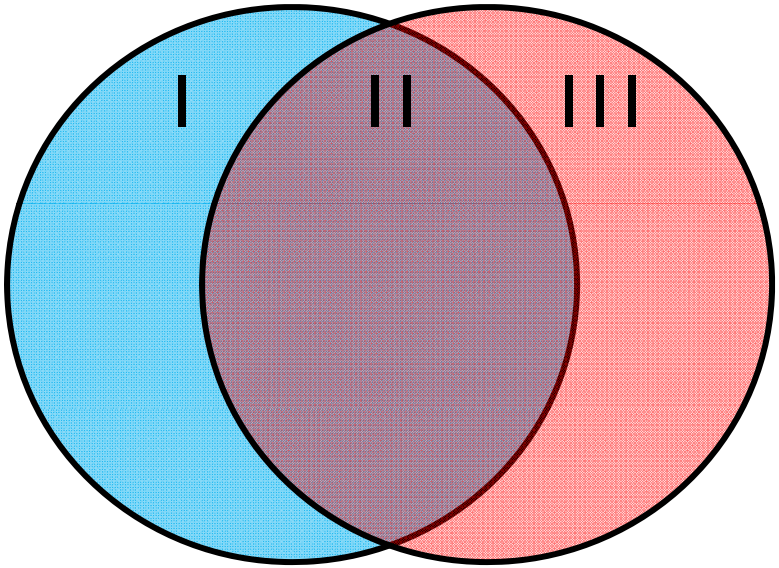


domain testing



CISS

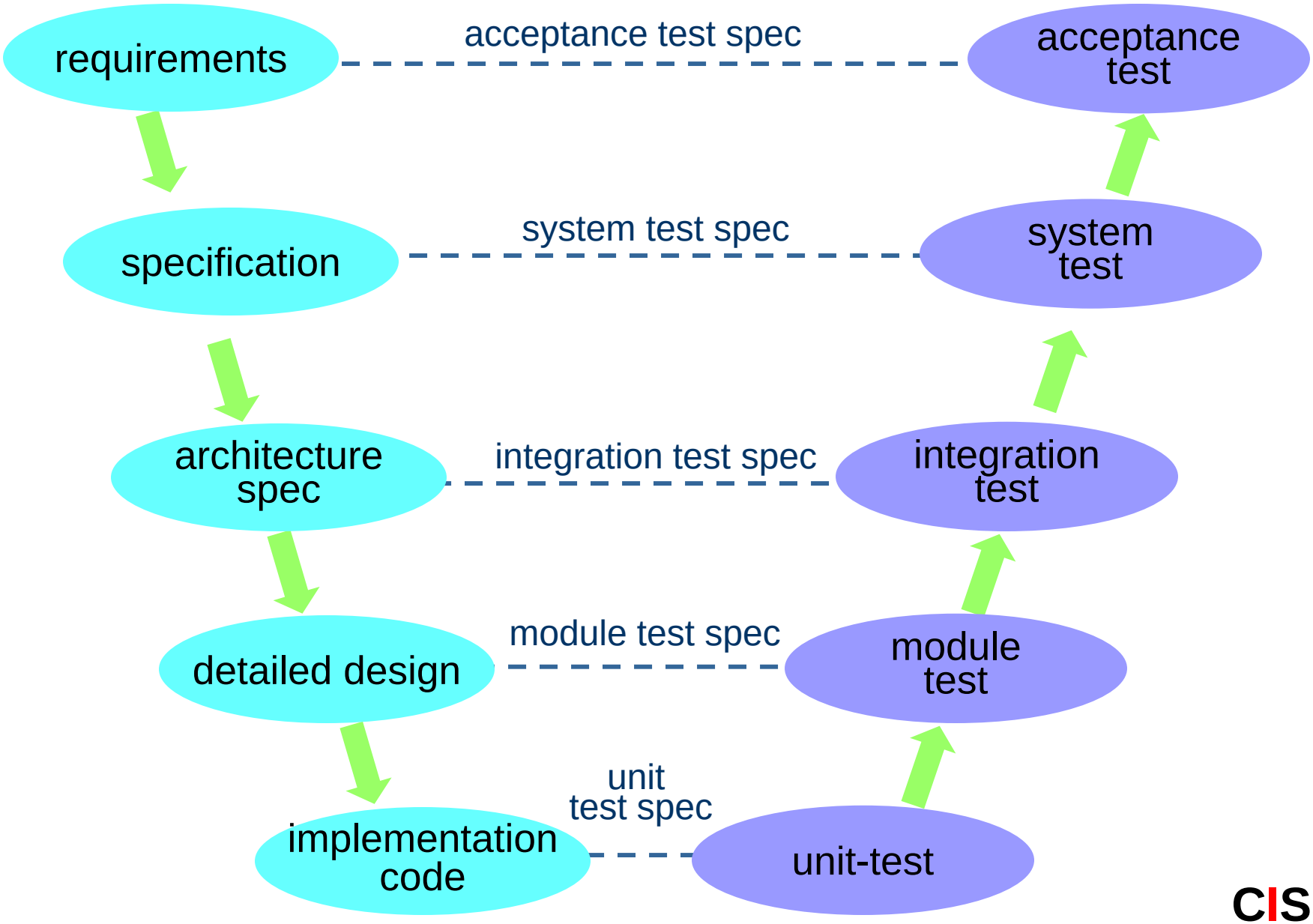
Blackbox vs. Whitebox



Specified Behavior
(Blackbox)

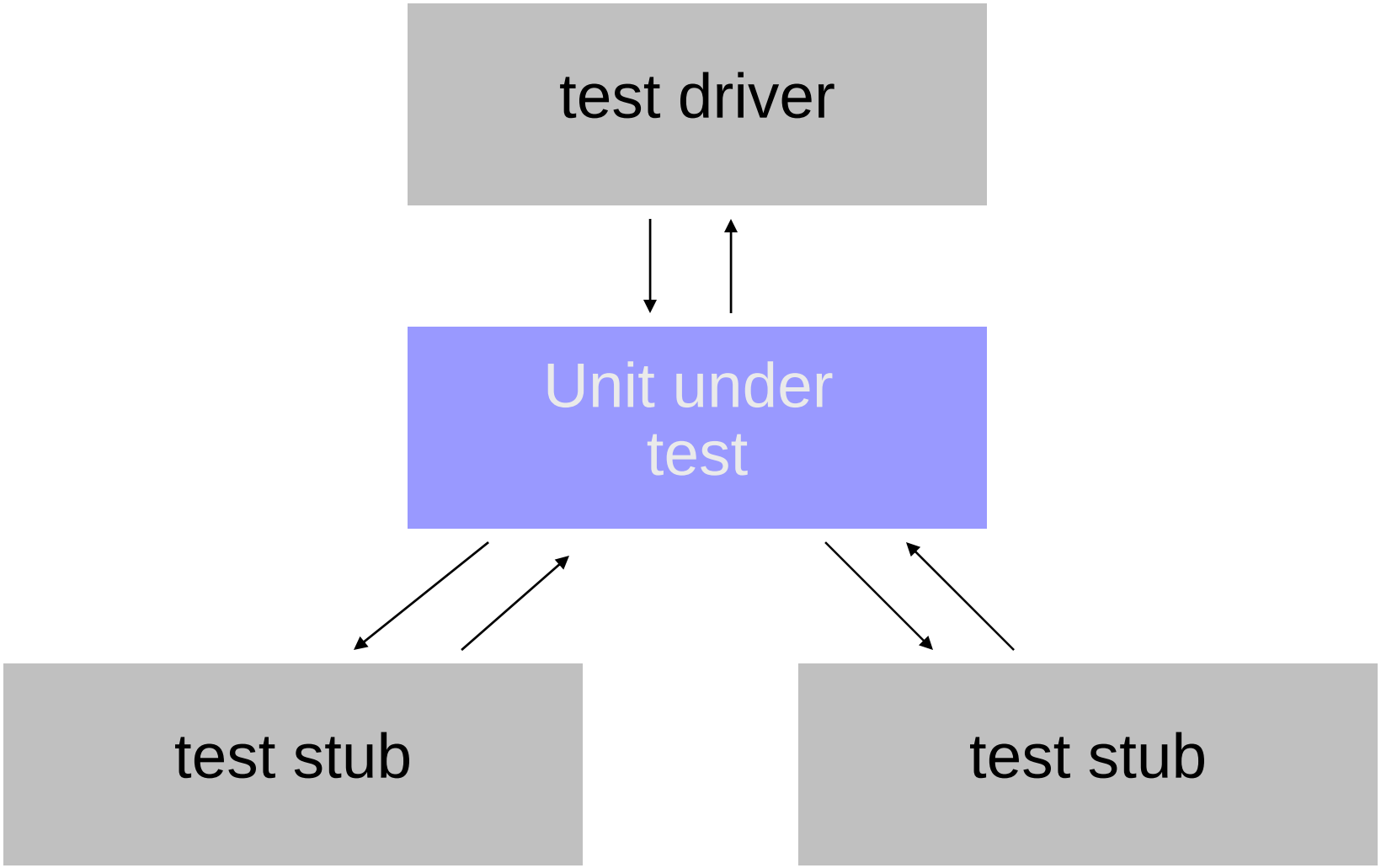
Implemented Behavior
(Whitebox)

V - Model

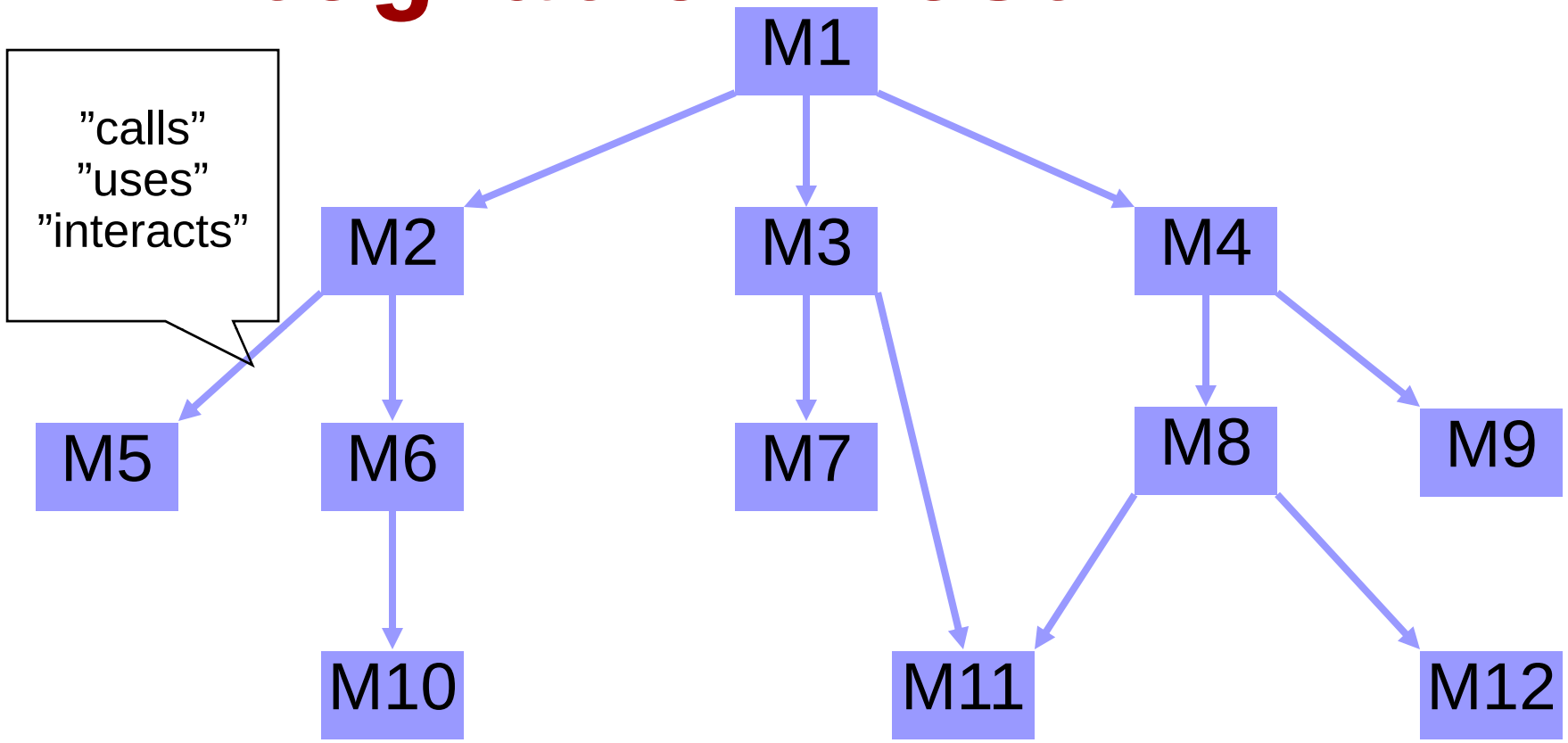

requirements
acceptance test spec
acceptance test
specification
system test spec
system test
architecture spec
integration test spec
integration test
detailed design
module test spec
module test
implementation code
unit test spec
unit-test

CISS

Unit Test



Integration Test



- Top Down
- Buttom Up
- Sandwich
- Depth vs Bredth first

CISS

System test

- 2*CRTG (4 channels) 2 * 200 k€

Test Equipment

- Complete Type Approval Test System (3 M€)

CISS

Acceptance Testing

- By customers
- Of costumer's requirements
- In costumer's environment

CISS

A Self-Assessment Test [Myers]

- "A program reads three integer values. The three values are interpreted as representing the lengths of the sides of a triangle. The program prints a message that states whether the triangle is scalene, isosceles, or equilateral."

- *Write a set of test cases to test this program*

A Self-Assessment Test [Myers]

Test cases for:

1. valid scalene triangle ?
2. valid equilateral triangle ?
3. valid isosceles triangle ?
4. 3 permutations of previous ?
5. side = 0 ?
6. negative side ?
7. one side is sum of others ?
8. 3 permutations of previous ?
9. one side larger than sum of others ?
10. 3 permutations of previous ?
11. all sides = 0 ?
12. non-integer input ?
13. wrong number of values ?
14. for each test case: is expected output specified ?
15. check behaviour after output was produced ?

Challenges of Testing

- Infinity of testing:
  - too many possible input combinations -- infinite breadth
  - too many possible input sequences -- infinite depth
  - too many invalid and unexpected inputs
- Exhaustive testing never possible:
  - when to stop testing ?
  - how to invent effective and efficient test cases with high probability of detecting errors ?
- Optimization problem of testing yield and invested effort
  - usually stop when time is over ……
- What is an effective method to measure coverage ?

Challenges of Testing

- Many operating environments and contexts
  - Impact of platform capabilities – OS, HW, Remote systems
  - Many versions of co-installed software
  - Typical and rare use patterns
- How to translate in to effective tests?

Challenges of Testing

- How can software fail ?
  - Typical programming errors
  - Typical wrongly implemented features
  - Exceptional cases
  - No realistic **reliability models** for software

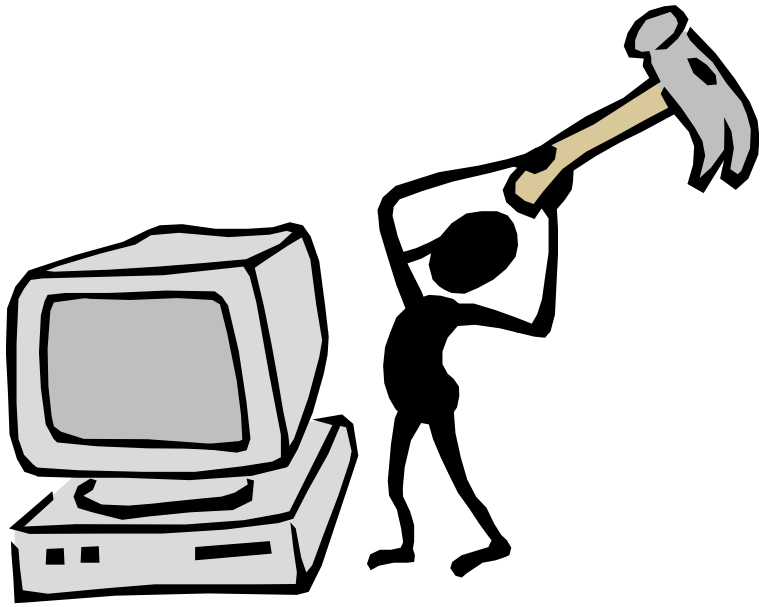
Challenges of Testing

- Test oracle problem
  - Bad specification or no specification at all
  - Requirements change
  - Requirements elucidation is a process

Challenges of Testing

- Regression testing:
  - very important
  - very boring and expensive
  - must be automated

Challenges: Who Should Test?



- Developer
  - Understands the system
  - But, will test gently
  - And, is driven by deadlines
- Independent tester
  - Must learn system
  - But, will attempt to break it
  - And, is driven by “quality”

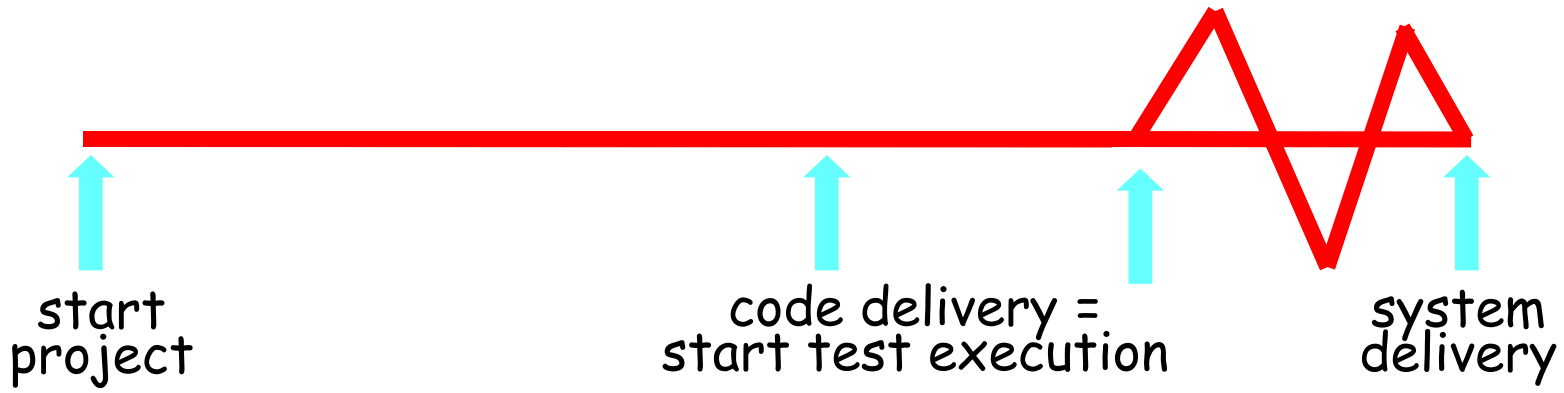
- Destructive, Unprestigious??

CISS

Challenges of Testing

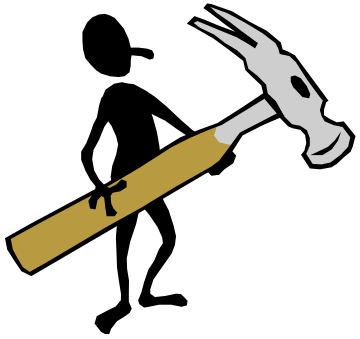
Moving implementation deadlines

..... but fixed delivery deadlines



Challenges of Testing

- Lack of appropriate tools
  - Diversified fields
  - Experts dispersed, but also doesn't talk across application domain.
  - Tools are specialized, sells in low volume
  - Tools are expensive,
  - Tools are immature
  - *No money available for test tools*



CISS

Testing Support Tools

- Test generation
  - analysis of system under test,
    its specifications, its environment and its interfaces
  - determination of test strategy
  - construction and specification of set of test cases
- Test execution
  - implementation of means for execution of specified tests
  - execution of specified tests
  - analysis and verdict assignment for executed tests
- Test organization
  - management and planning of test process
  - allocation of resources
  - testware management and consolidation for regression testing

Challenges of Testing

- New embedded systems
  - more functionality
  - increasingly advanced
  - faster time-to-market
  - higher quality
- Testing
  - more to be tested
  - more complicated
  - in less time
  - more thorough

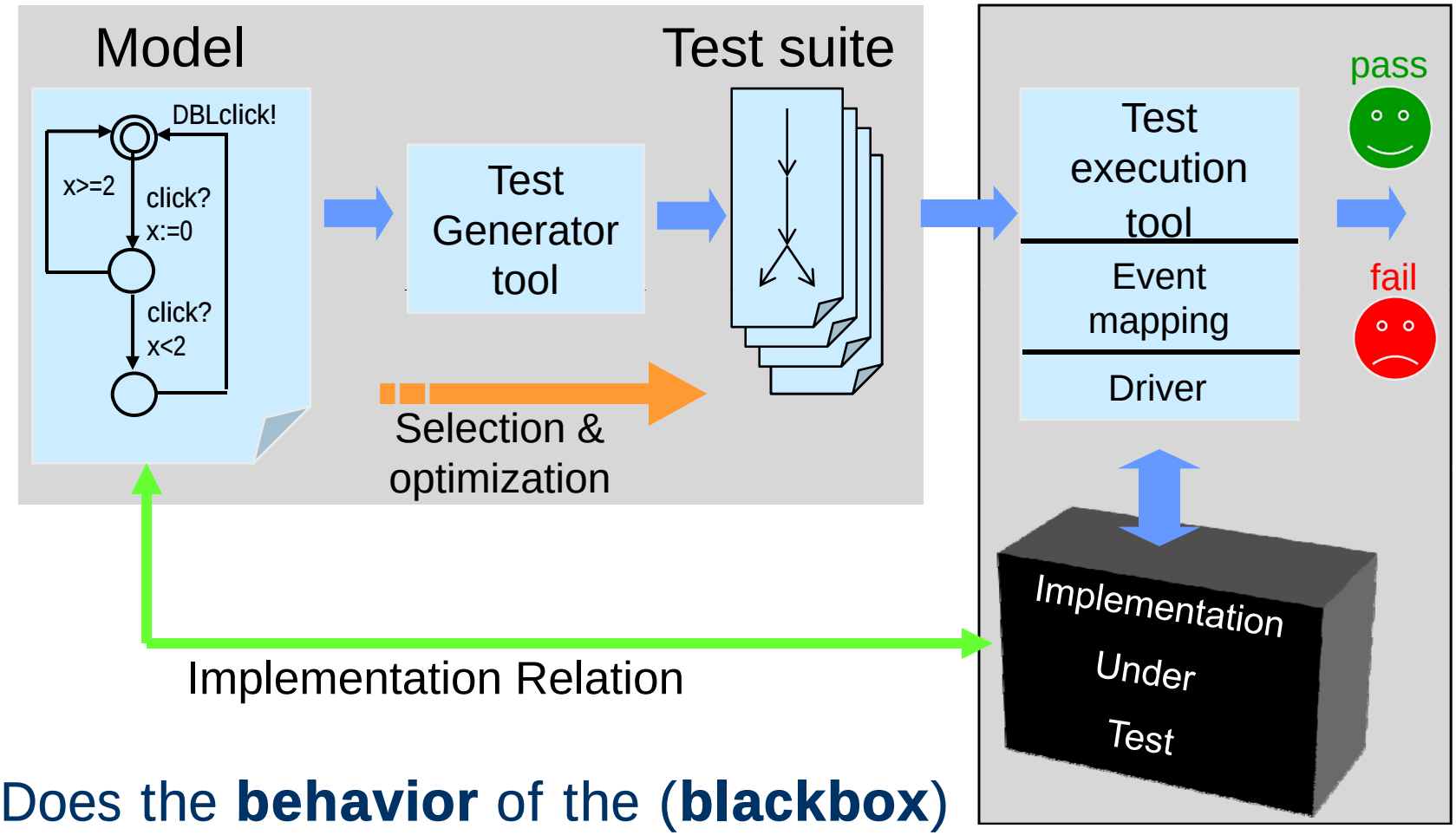
- skilled developers and testers
- advanced testing tools and techniques
- well organized
- using solid development method

Manual Testing

1. Figure out what to test?!
2. Design good (abstract) test cases
3. Implement as test programs, scripts (or manual execution and analysis)
4. Execute test cases
5. Analyze results
6. Goto 1

- Change
  - Requirements
  - System understanding evolve
  - Implementation change
- Regression testing only practical when automated

Automated Model Based Conformance Testing



Does the **behavior** of the (**blackbox**) implementation ***comply*** to that of the specification?

CISS

Summary

BRICS
Basic Research
in Computer Science

Some Testing Principles

- Testing starts during the requirements phase
- The programmer shall not be the (only) tester
- A test case specifies the test inputs
  and the expected outputs
- Test cases shall also cover invalid and unexpected inputs
- Test cases shall test that the program does what it should do
  and that it does not do what it should not do
- Test cases shall be recorded for reuse
- A test is successful when it detects an error
  (but the project manager thinks differently !)
- No risk, no test

END

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER