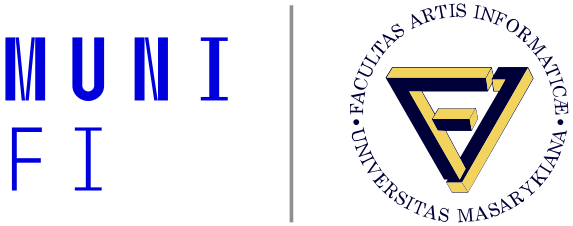


SW Testing

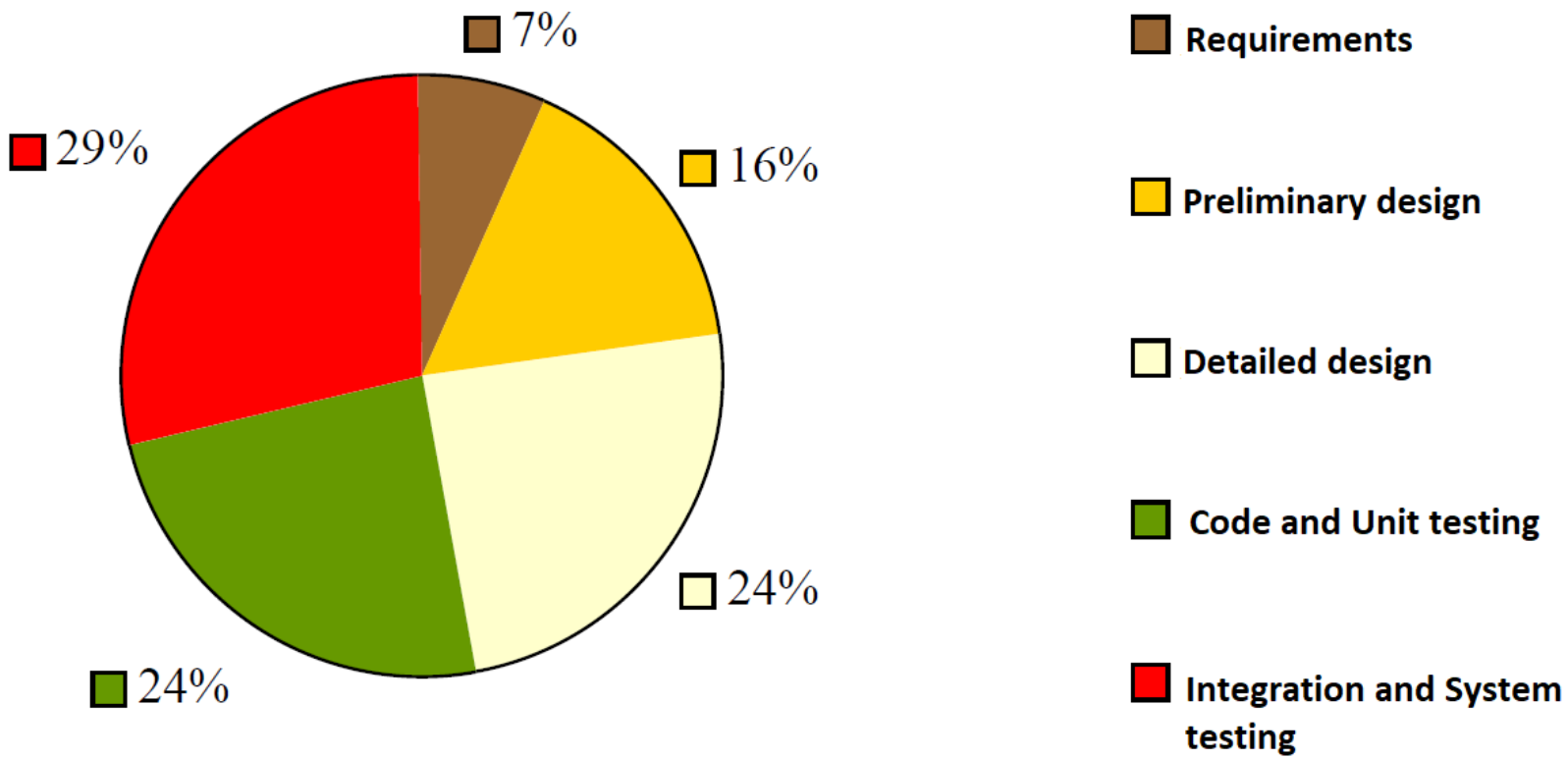
PA017 SW Engineering II → Aspects of SW Development Management

Jaroslav Ráček **Josef Spurný**

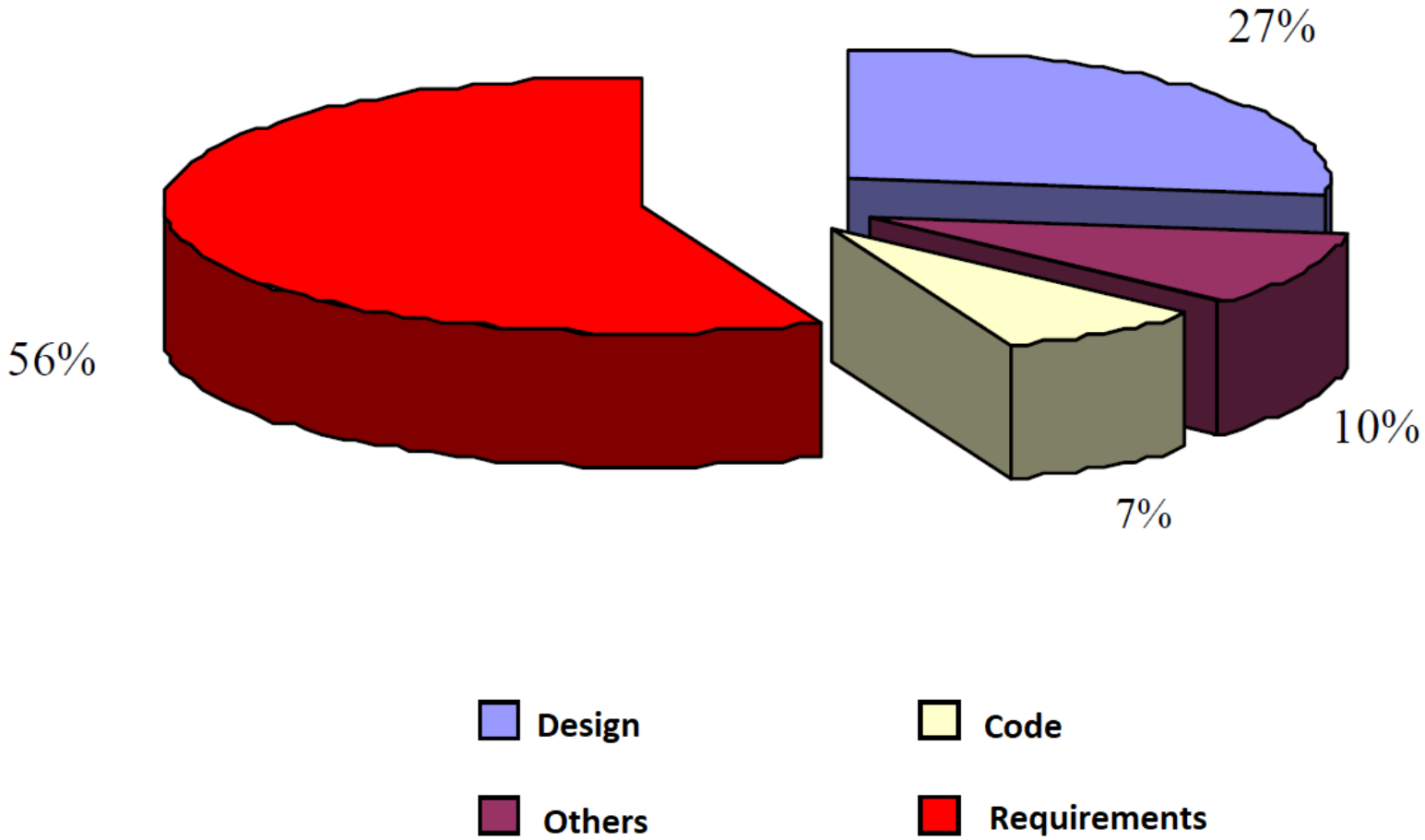
Faculty of Informatics, Masaryk University

November 8, 2022

Price of testing during development



Sources of defects

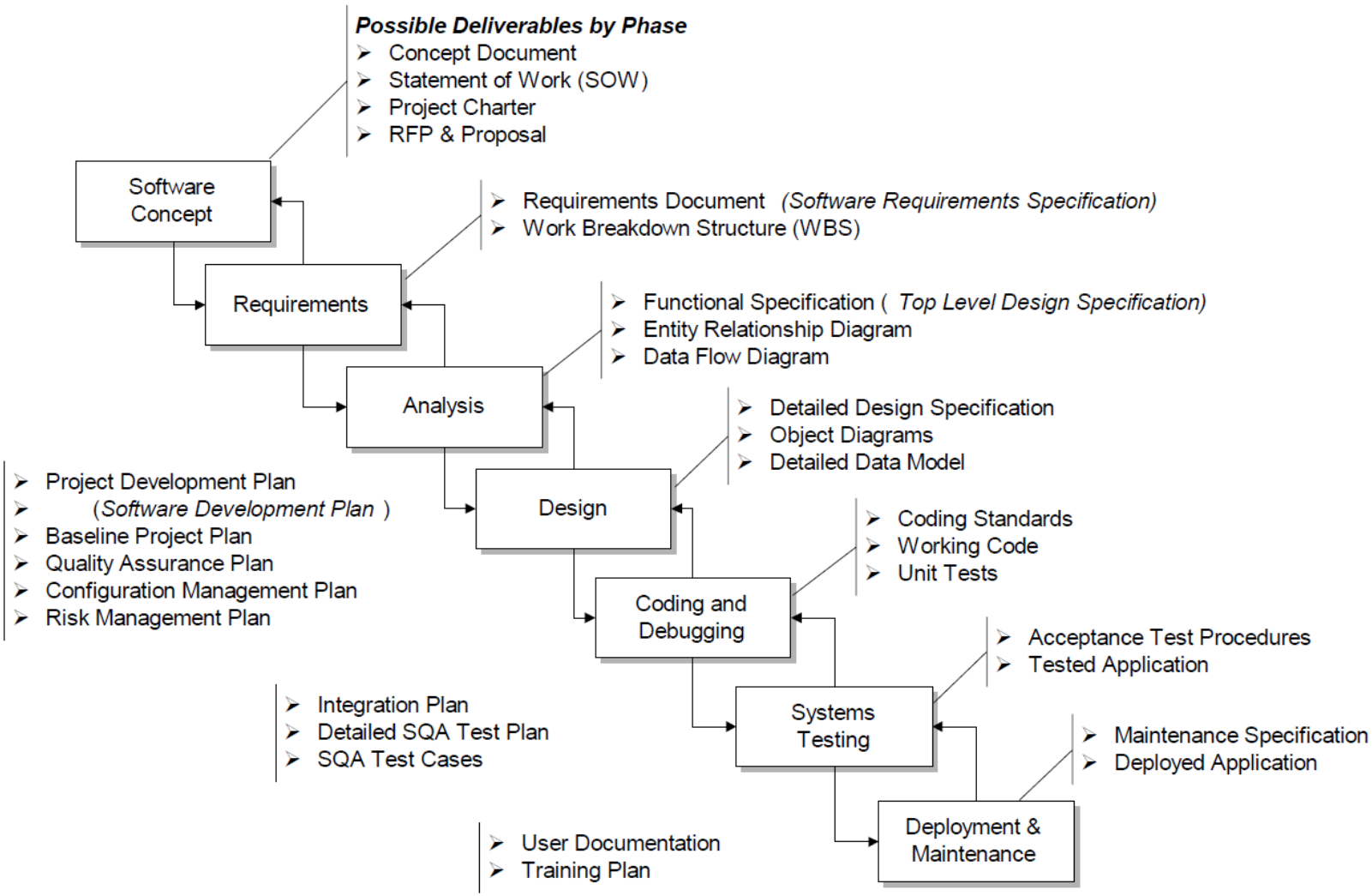


If 99.9 % of functions worked fine...

... then every year:

- CZ: 900,000 bank transfers were incorrectly booked
- CZ: 214,000 letters will be lost
- CZ: 75,000 patients will receive wrong drug prescription
- World: 31,000 flights will crash during landing

Deliverables by phases



What is SW testing?

Testing

is a process of executing a program with the aim of detecting a defect

- Good test case has a high probability of detecting a so-far-undetected defect
- A successful test is one which has detected so-far-undetected defect

Successful test

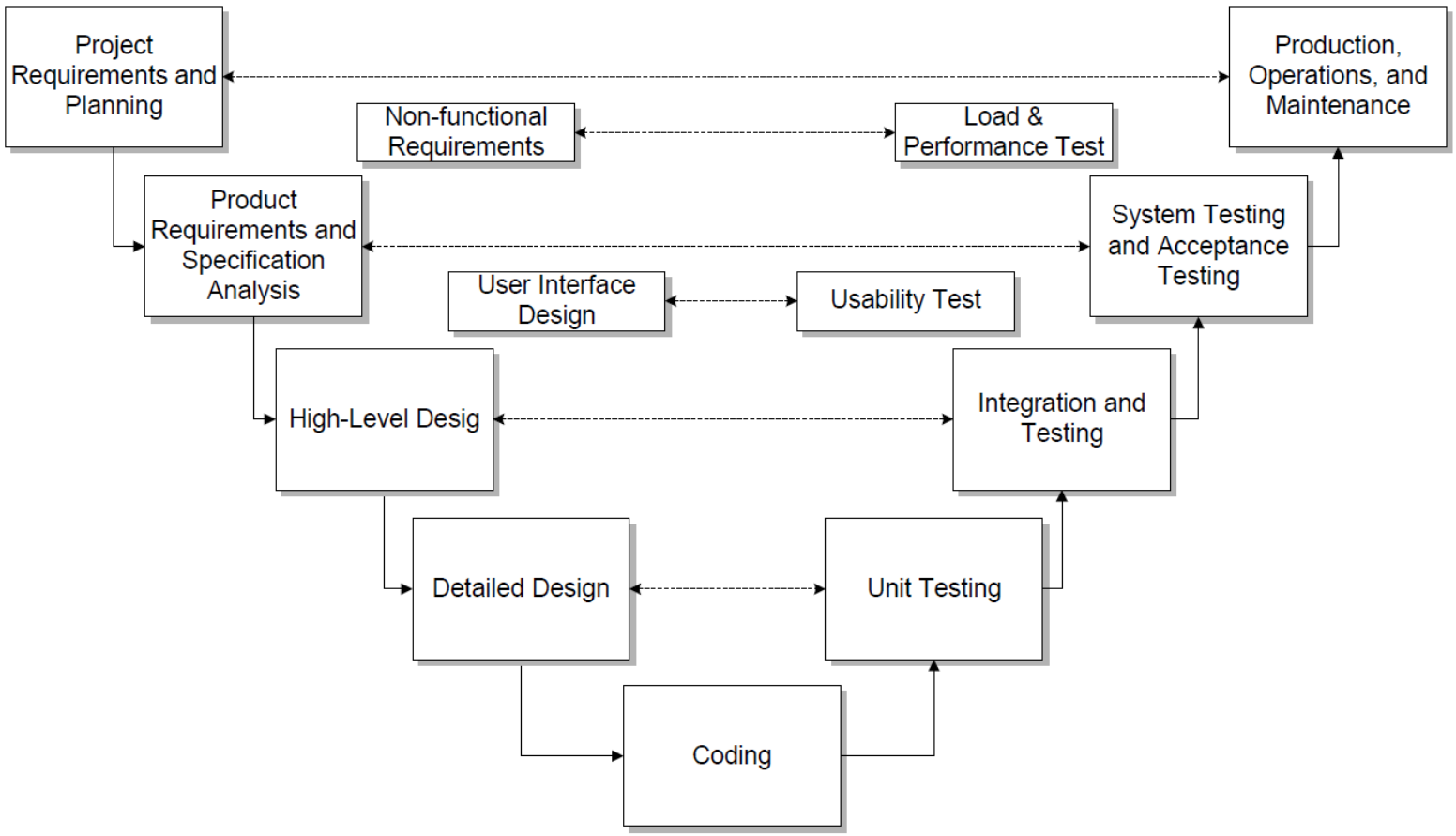
~~Def.: Test is successful if it does not detect any anomaly in program output~~

Definition

Test is successful if it detects one or more defects in the program.

The Art of Software Testing
Glenford J. Myers, 1979

V-Model



Activities done earlier in the project are tested later – the more costly they get.

What does testing reveal?

- Testing cannot reveal absence of defects, it can only reveal their presence
- Testing also shows functionality and performance
- Overall, it is an indicator of SW quality

Verification & Validation

Any engineering product may be tested in a following manner:

- testing against specified functionality = Validation
  "*Do right things*"
- testing against internal procedures = Verification
  "*Do things right*"

Application in V-Model: Validation is typically done by customer, verification is done by internal team

Testing in a team

- Testing is a destructive endeavor
- Programmer is not a good tester of his / her own code
- Detailed knowledge of code structure makes it easier to find and fix a defect
- Cooperation of two independent teams is necessary:
  Developers team & Quality team

Complete testing

- Even for a small programs, the amount of independent logical paths through may be very high
- Code with 100 lines, several embedded cycles, 20 different parameter setups $\rightarrow 10^{14}$ logical paths
- Testing at rate of $\frac{test}{ms}$ would take 3170 years
- Complete testing is not possible!

Selective testing

- Even when complete testing is not realistic to perform (almost every time), the "whitebox" testing shall not be left out
- Important logical paths and cycles shall be tested
- Selective testing validates the interface and increases trust in the internal program structure

Dynamic testing

- Executing the program with pre-defined inputs
- Comparison of obtained results with expected results
- This approach is in fact sampling, it cannot prove absence of defects
- Every software has bugs and testing cannot guarantee their absence

Test cases

- Key items for every testing plan
- May include scripting, data, and control lists
- May be related to *Matrix of requirements coverage* – a tool for requirements tracking

Whiteboxes and Blackboxes

Function

- testing the activities executed by a function
- Blackbox testing

Internal procedure

- testing whether "all engines are running"
- Whitebox testing

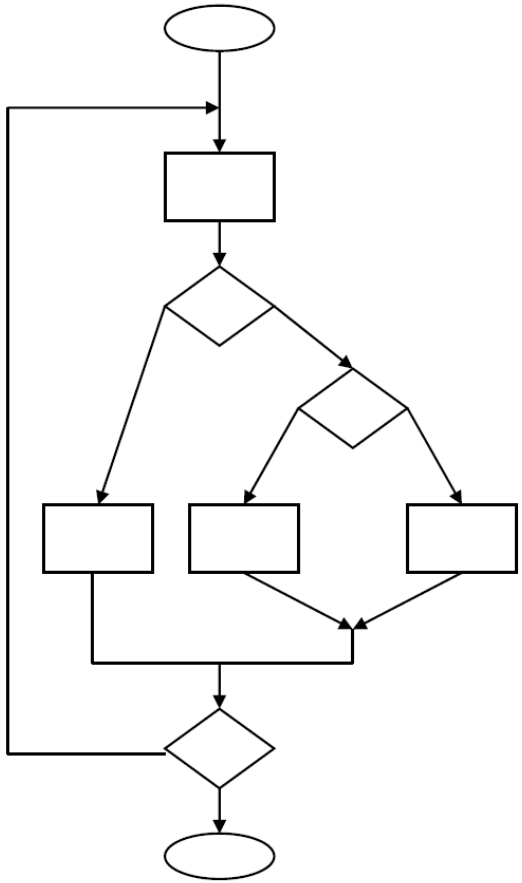
Blackbox testing

- Functional testing
- Program is a blackbox
  - We do not care about internal procedure, we care about the result
  - Focused on inputs & outputs
- Test cases based on functional requirements specification

Whitebox testing

- Considers the internal program structure
- Covers:
  - executed commands
  - independent paths through code
- Test cases based on functional requirements specification

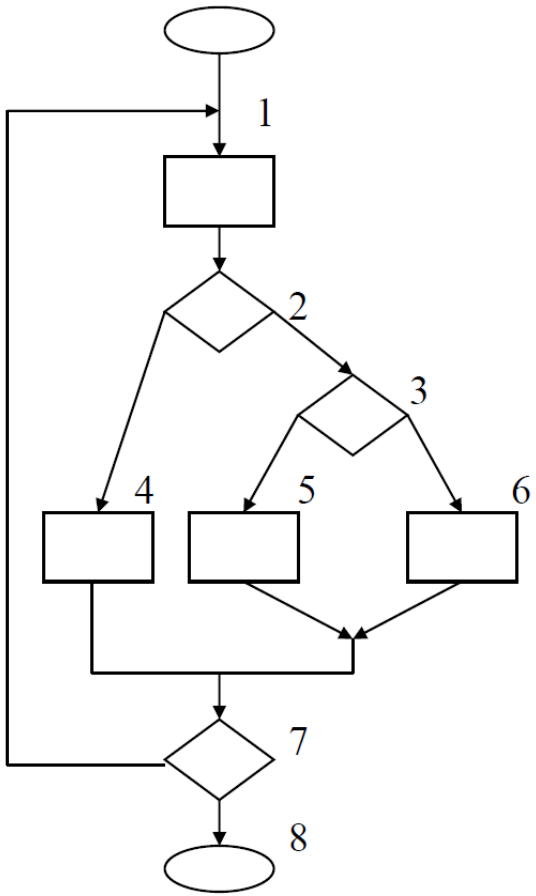
Whitebox testing



Step 1: Calculate cyclomatic complexity

Number of decisions (predicate nodes) + 1
or
Number of regions
or
Edges - Nodes +2

Whitebox testing



Step 2: Determine independent paths

Since cyclomatic complexity is 4, there are 4 independent paths:

path 1: 123678
path 2: 123578
path 3: 12478
path 4: 1247124…78

Whitebox testing

- The flowchart is not necessary, but visual representation might help to determine all paths
- Testing the elementary paths shall be included for critical functionality modules
- However, it does increase cost for development

Unit & Modules testing

- Most commonly whitebox, sometimes blackbox
- Units are tested by programmers
  - unit tests are written in the same language as the module
  - alternative name: "*Test drivers*"
- Individual test may be grouped to "*Test suites*"
- Units are tested continuously, once given module is finished

Integration & Testing

- Development / Integration / Testing
  - the most frequent point where activities overlap
- Sometimes, Integration & Testing is considered as one phase
- Continuous integration of functionality
- QA team works in parallel with developers team

Integration approaches

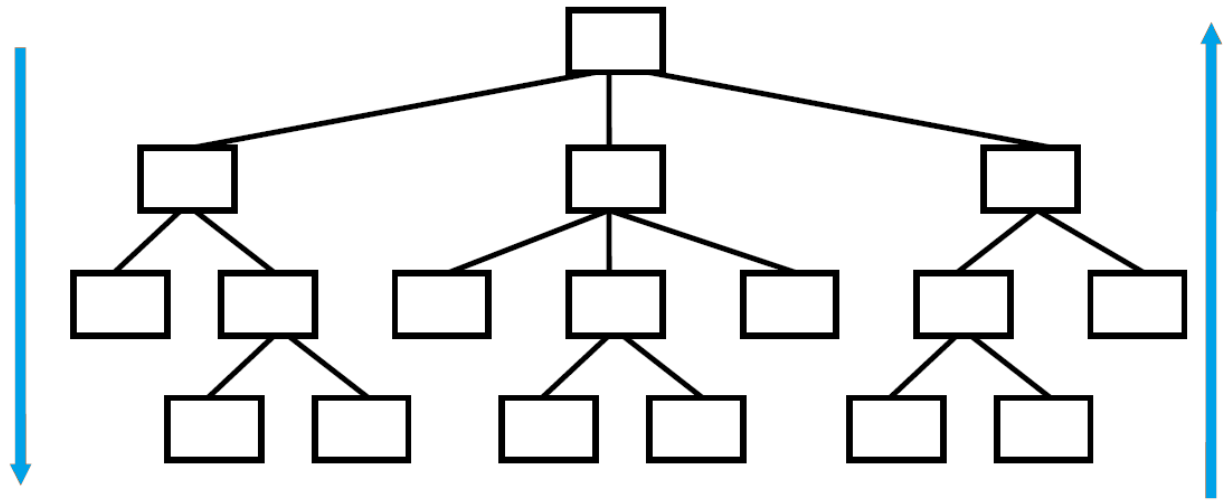
Top-down:

- First, the system core (backbone) is implemented
- Then, core is supplemented with a minimal "*shell*" of the system
- Not-yet-implemented modules are replaced with "*stubs*" (prosthetics), that are later replaced with fully-functional modules

Bottom-up:

- Begins with implementation of individual increments – units
- After unit testing, the units are combined into subsystems
- Subsystems are combined into final system

Top-down testing, Bottom-up testing



TDT: usage of stubs – simple substituent objects with same interface
BUT: classic testing procedure with higher-level testing objects – drivers
TDT reveals errors in analysis and design, and is consistent with prototyping approach

Disadvantages of TDT & BUT

TDT disadvantages

- complex modules cannot be easily replaced with a stub
- Using stubs may cause that results of higher-level tests might not be visible

BUT disadvantages

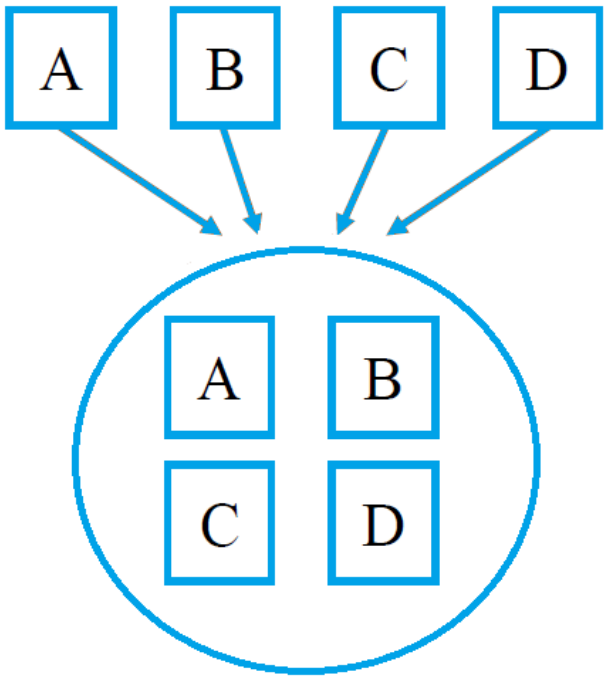
- Costs for drivers construction are typically higher than costs for constructing stubs
- A program suitable for demonstration to the customer is delivered later when compared to TDT

Both approaches have their disadvantages, hence none is the best

Attributes of integration

- Integration testing is performed by QA and/or developers team
- Project budget and number of involved team members are at the peak
- Issues:
  - work under pressure
  - final deadline is approaching
  - unexpected bugs
  - motivational problems
  - conflicts at product handover to the client

Integration testing



Unit testing

Integration testing

How do we find out where is the defect?

→ Incremental integration & testing

MUNI
FACULTY
OF INFORMATICS