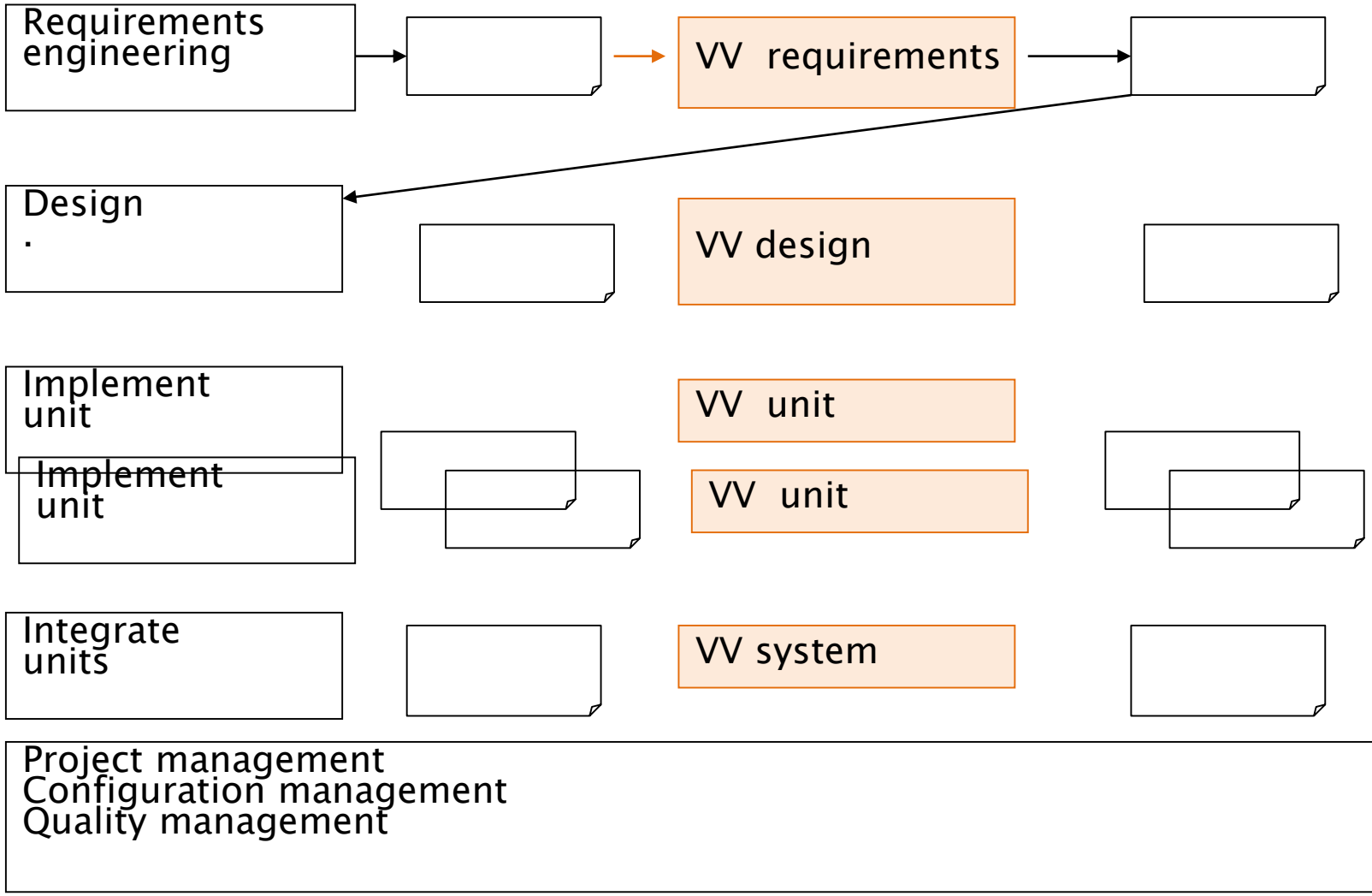
Verification & Validation

Object-Oriented Programming

SoftEng
http://softeng.polito.it

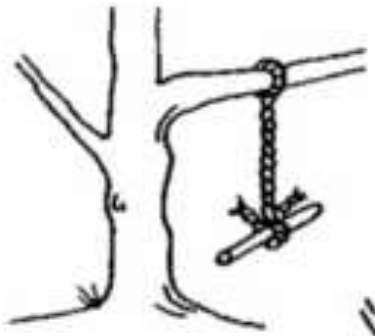
The whole picture

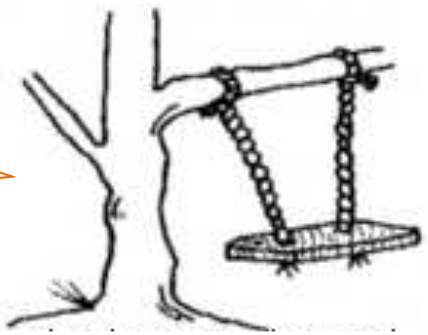


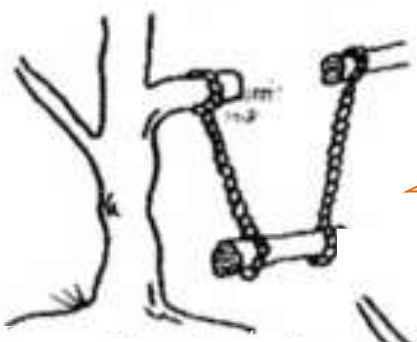
V&V

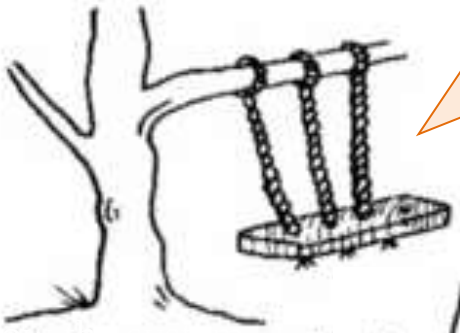
- Validation
  - is it the right software system?
  - effectiveness
  - external (vs user)
  - reliability
- Verification
  - is the software system right?
  - efficiency
  - internal (correctness of vertical transformations)
  - correctness

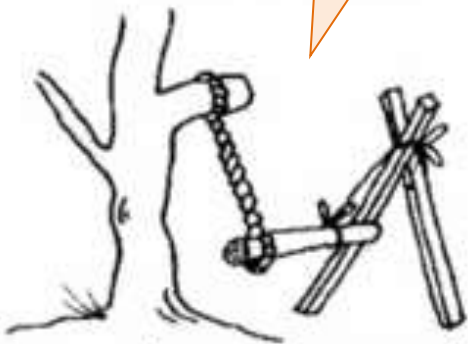
Validation





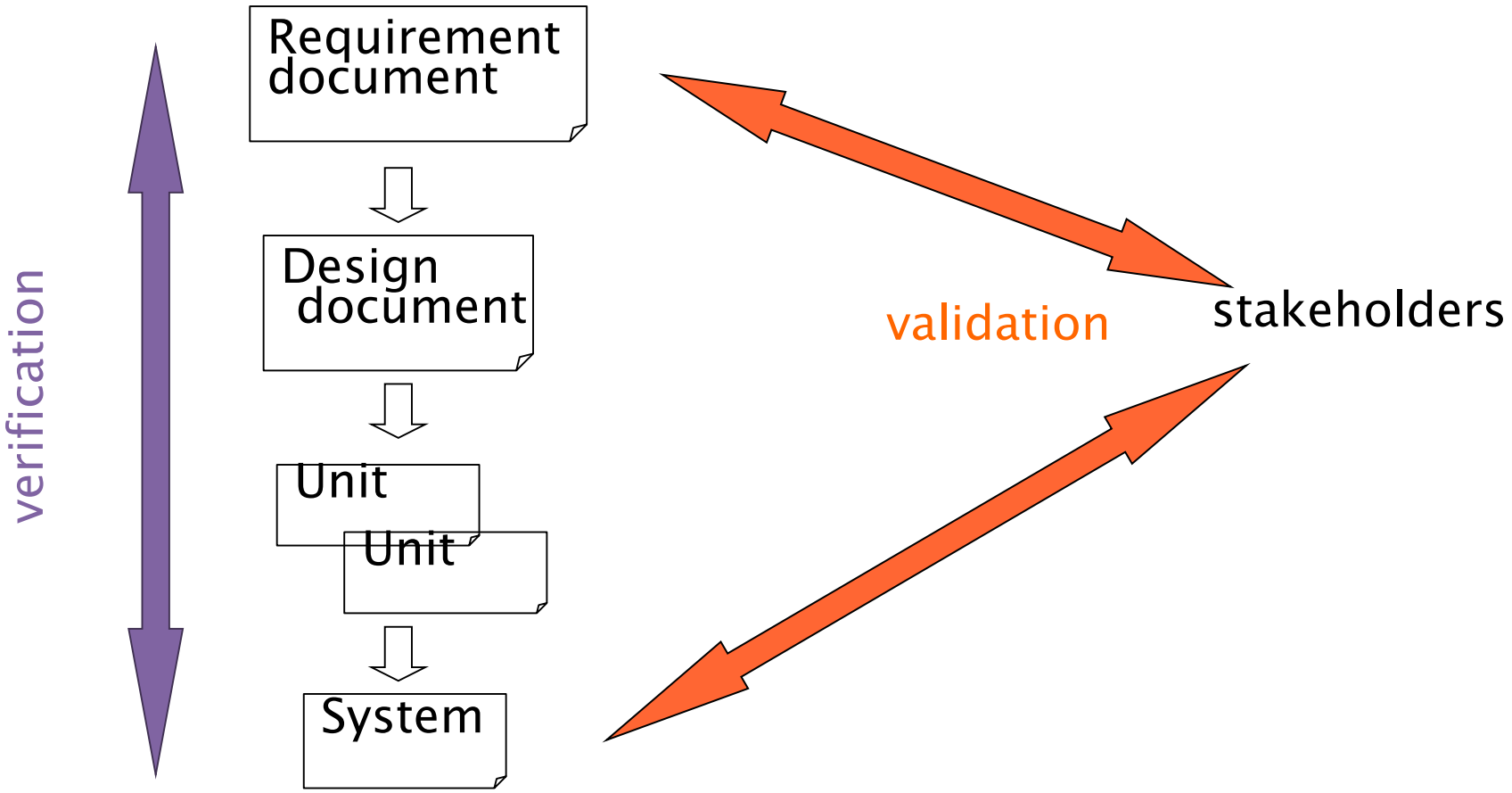






Verification

V & V

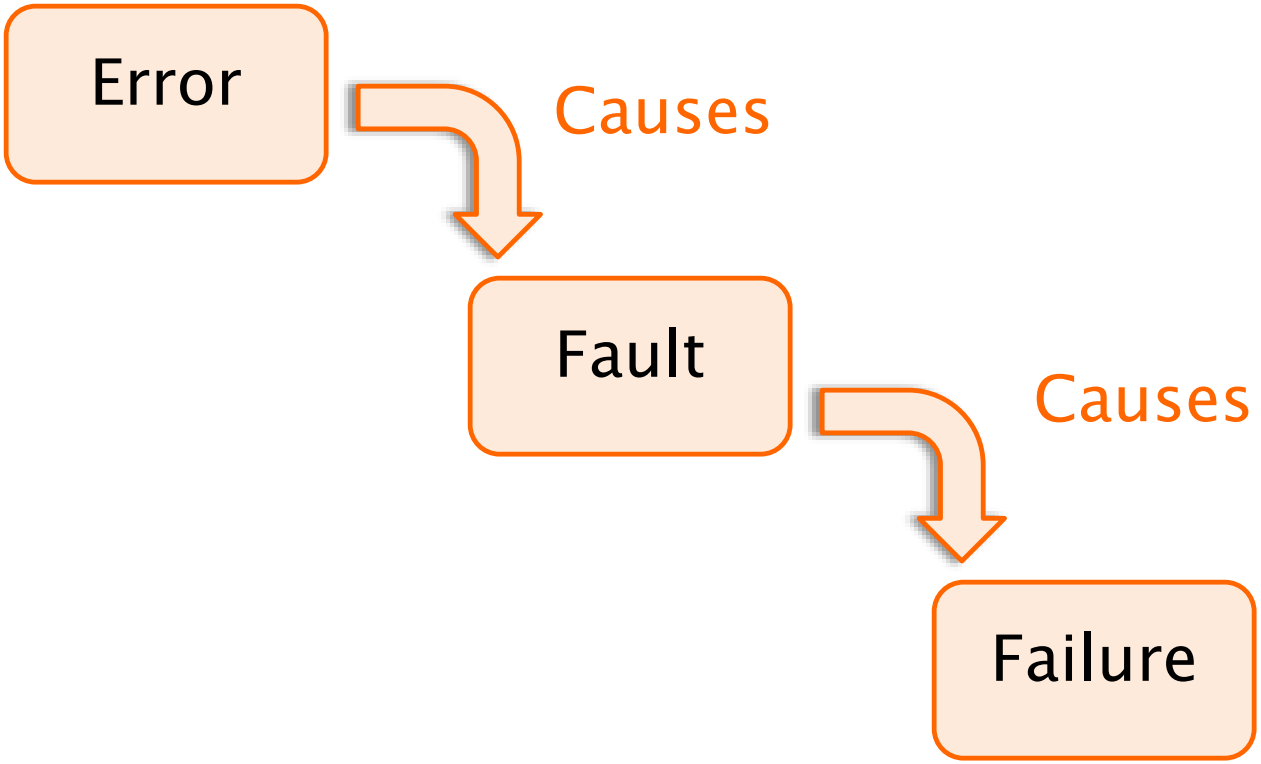

Requirement document
Design document
Unit
Unit
System
verification
validation
stakeholders

TERMINOLOGY

Failure, fault, defect

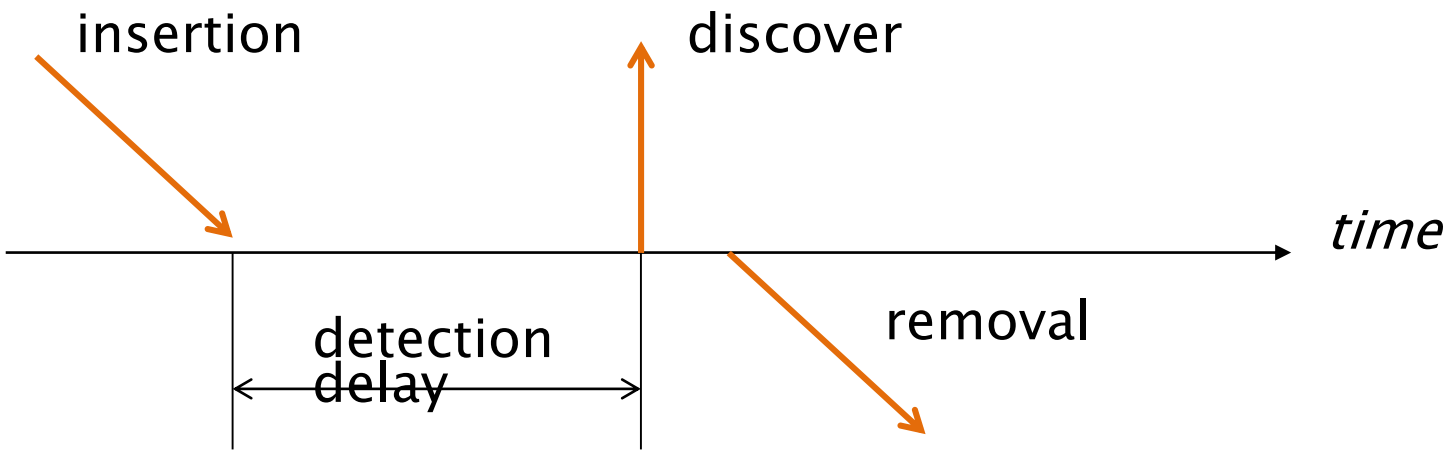
- Error
  - A mistake e.g. committed by a programmer
- Fault (Bug)
  - The feature of software that causes a failure
  - May be due to:
    - An error in software
    - Incomplete/incorrect requirements
- Failure
  - An execution event where the software behaves in an unexpected way
- Defect
  - Typically a fault (sometimes a failure)

Error–Fault–Failure



Insertion / removal

- Defect is characterized by
  - Insertion activity (phase)
  - Removal activity (phase)



Basic goals of VV

- Minimize number of defects inserted
  - Cannot be zero due to inherent complexity of software
- Maximize number of defects discovered and removed
- Minimize detection delay

V&V approaches

- Static
  - inspections
  - source code analysis
- Dynamic
  - testing

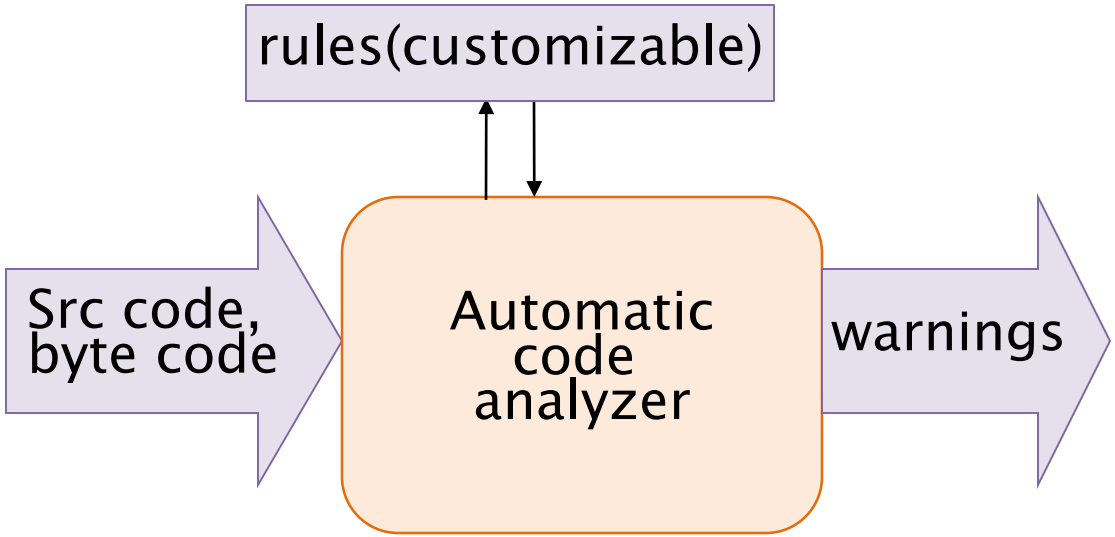
STATIC ANALYSIS

Static analysis techniques

- Compilation static analysis
- Control flow analysis
- Data flow analysis
- Symbolic execution
- Inspections

Automatic code analysis

- It is performed
  - without actually executing programs (at compile time)
  - On source code, or byte code



Code smells

- A code smell is a surface indication that usually corresponds to a deeper problem in the system
- Smells are certain structures in the code that indicate violation of fundamental design principles and negatively impact design quality

Fowler et al., Refactoring, Improving quality of existing code. Addison-Wesley

Technical Debt

- Technical debt reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution
- “Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt.” [W.Cunningham]

TESTING

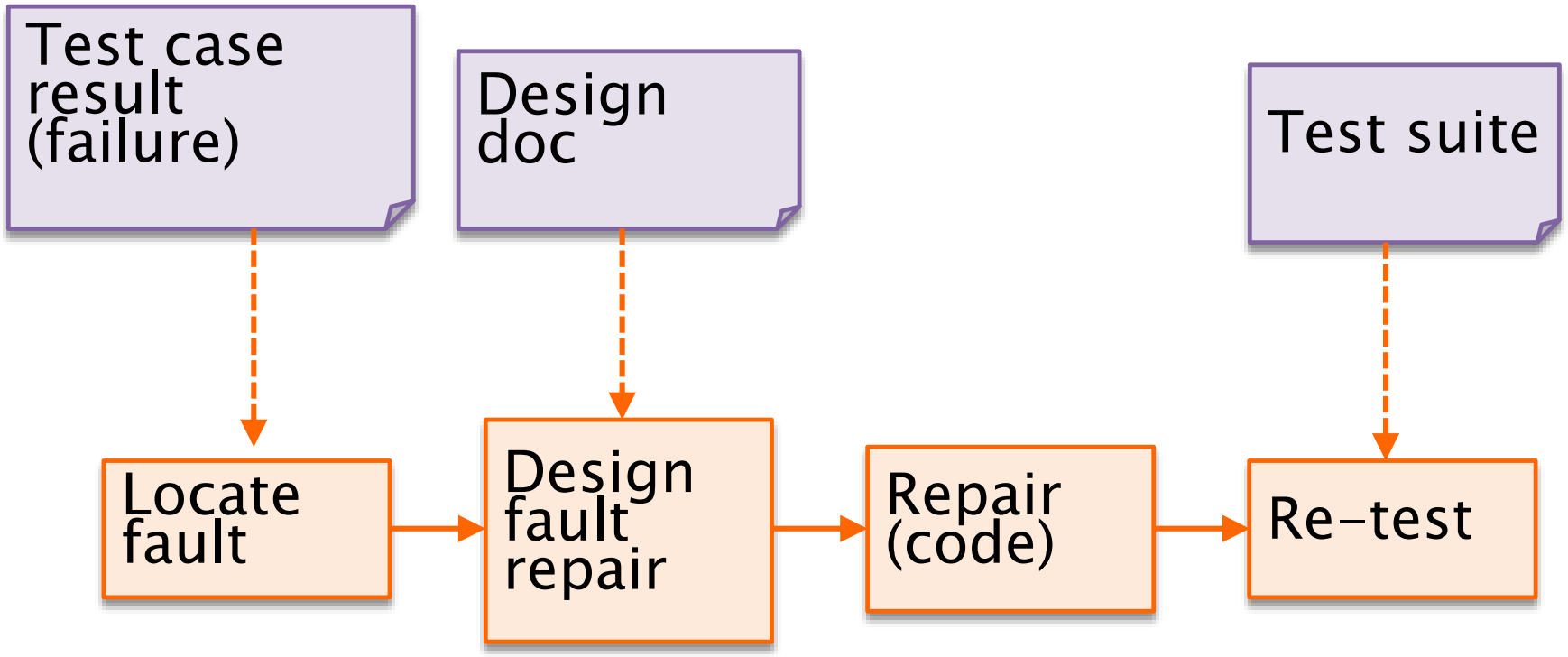
Purpose of test

- The purpose of testing process is to find defects in the software products
  - A test is successful if it reveals a defect
- The process of operating a system or component under specified conditions observing or recording the results to detect the differences between actual and required behavior (= failures)

Testing vs. debugging

- Defect testing and debugging are different activities
  - May be performed by different roles in different times
- Testing tries to find failures
- Debugging searches for and removes the fault

Debugging


Test case result (failure)
Design doc
Test suite
Locate fault
Design fault repair
Repair (code)
Re-test

Test case

- Certain stimulus applied to executable (system or unit), composed of
  - name
  - input (or sequence of)
  - expected output
- With defined constraints/context
  - ex. version and type of OS, DBMS, GUI ..
- A set of related test cases forms a test suite

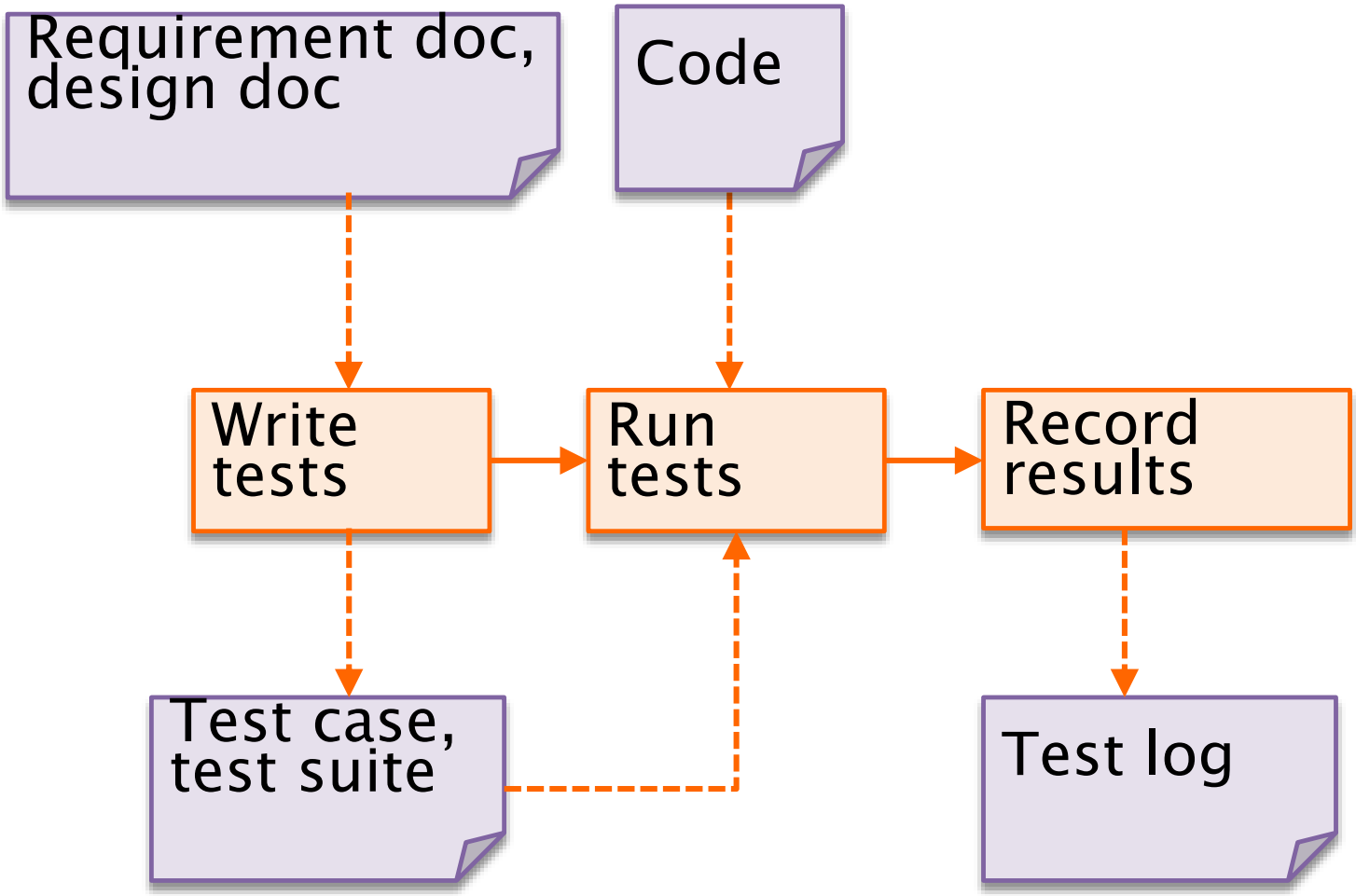
Test case log

- Test case reference +
  - Time and date of application
  - Actual output
  - Result (pass / no pass)

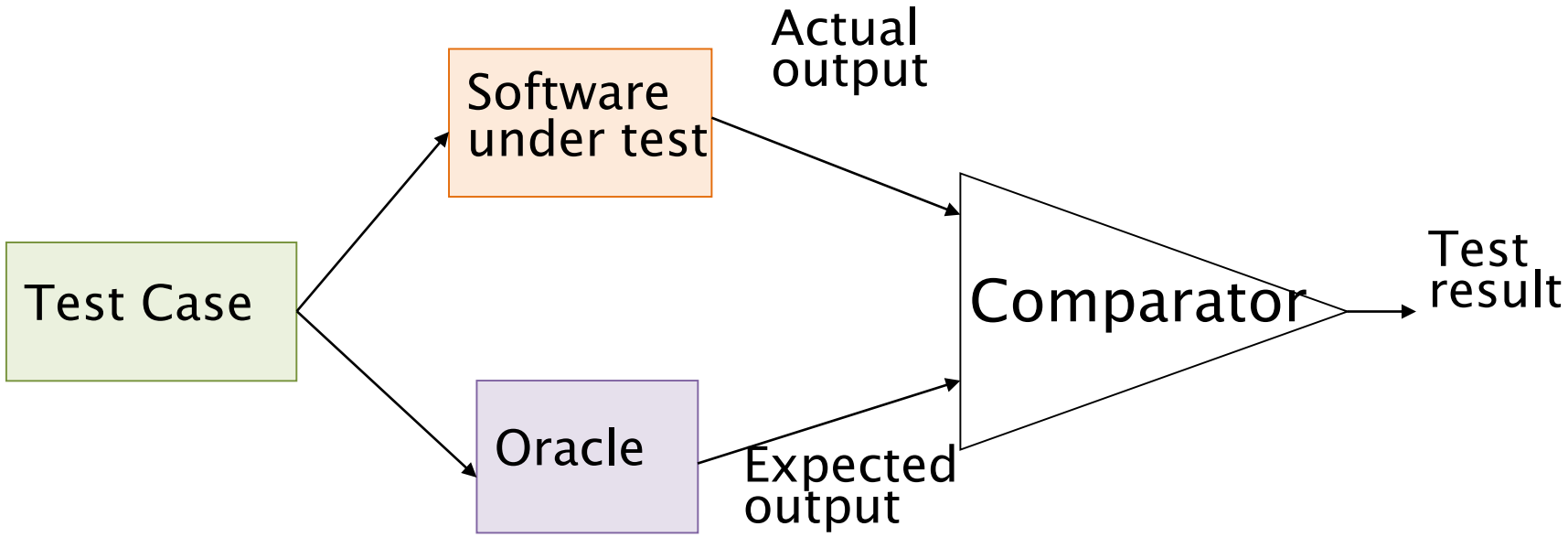
Ex.

- Function **`add(int x, int y)`**
- Test case:
  - T1(1,1; 2)
  - T2(3,5; 8)
- Test suite
  - TS1{T1, T2}
- Test log
  - T1, 16-3-2013 9:31, result 2, success
  - T2, 16-3-2013 9:32, result 9, fail

Test activities



Oracle


Actual output
Software under test
Test Case
Comparator
Test result
Oracle
Expected output

Oracle

- The ideal condition would be to have an automatic oracle and an automatic comparator
  - The former is very difficult to have
  - The latter is available only in some cases
- A human oracle is subject to errors
- The oracle is based on the program specifications (which can be wrong)

Oracle

- Necessary condition to perform testing:
  - Know the expected behavior of a program for a given test case (oracle)
- Human oracle
  - Based on req. specification or judgment
- Automatic oracle
  - Generated from (formal) req. specification
  - Same software developed by other parties
  - Previous version of the program (**regression**)

Software peculiarities

- No ageing
  - If function sum(2,3) works, it works forever
    - Supporting microprocessor will eventually fail for age, not the software
- Not linear, not continuous
  - If sum(2,3) works, may be sum(2,4) does not

Exhaustive test

- function: $Y = A + B$
- A and B integers, 32 bit

- Total number of test cases : $2^{32} * 2^{32} = 2^{64} \approx 10^{20}$
- 1 µs/test $\Rightarrow$ ~600 thousands years

Exhaustive test

- Exhaustive test is not possible
- So, goal of test is finding defects, not demonstrating that systems is defect free
- Goal of test (and VV in general) is assuring a *good enough* level of confidence

Dijkstra thesis

- *Testing can only reveal the presence of errors, never their absence*

E. W. Dijkstra. Notes on Structured Programming.
In *Structured Programming,* O.-J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, Eds. Academic, New York, 1972, pp. 1–81.

Test classification

- Per phase/granularity level
  - Unit, integration, system
  - Regression
- Per approach
  - Black box (functional)
  - White box (structural)
  - Reliability assessment/prediction
  - Risk based (safety security)

Test per granularity level/phase

- Unit tests
  - Individual modules
- Integration tests
  - Modules when working together
- System tests
  - The system as a whole (usable system)

Unit test

- Black box (functional)
  - Random
  - Equivalence classes partitioning
  - Boundary conditions
- White Box (structural)
  - Coverage of structural elements
    - Statement
    - Decision, condition (simple, multiple)
    - Path
    - Loop

Integration test

- Add one unit at a time, test the partial aggregate
  - Defects found, most likely, come by last unit/interaction added

Stub, driver

- Driver
  - Unit (function or class) developed to pilot another unit
- Stub
  - Unit developed to substitute another unit (fake unit)
- Also called mockups

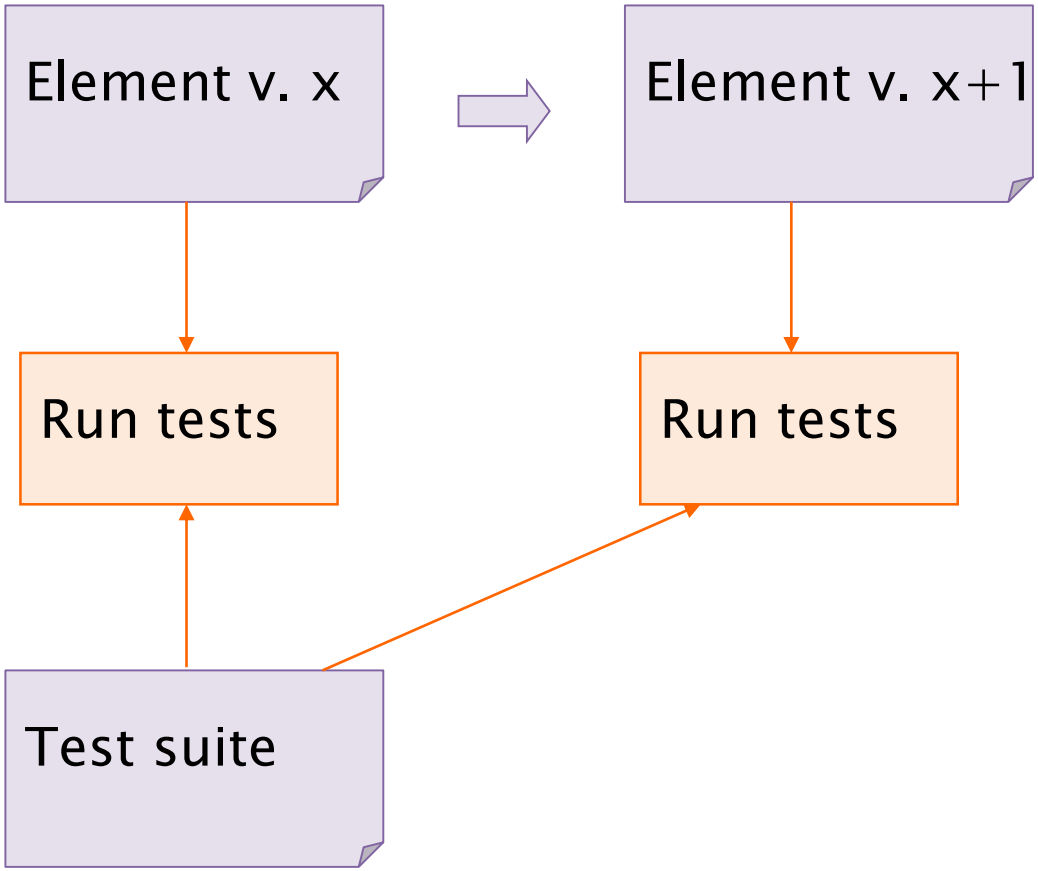
System test

- Is applied to the software system as a whole
  - Aims at verifying the correspondence of the system to the requirements
- Test of functional requirements
  - Coverage of uses cases/scenarios as listed in requirement document
  - Consider usage profile (the most common, typical ways of using the system)
- Test in conditions as far as possible close to working conditions

Regression testing

- Regression testing
  - Tests previously defined are repeated after a change
  - To assure that the change has not introduced defects
    - Time0
      - Element (unit, system) in v0, test set t0 is defined and applied, all tests pass
    - Time1
      - Element is changed to v1
      - Test set t0 is re-applied, do all tests still pass?

Regression testing



References and Further Readings

- IEEE Std 829-2008: **IEEE Standard for Software and System Test Documentation**
- Fowler et al., Refactoring, Improving quality of existing code. Addison-Wesley