Advanced Topics in Object Technology

Bertrand Meyer

Contact

- Chair of Software Engineering:
  - http://se.inf.ethz.ch
- Course assistant:
  - Karine Arnout
  - http://se.inf.ethz.ch/people/arnout

Lecture 1:

Introduction, Quality issues, Lifecycle

Agenda for today

- Introduction
- Quality issues
- Lifecycle

Agenda for today

- Introduction
- Quality issues
- Lifecycle

Introduction

- Course objectives
- Topics
- Technologies
- Guest lectures
- Textbook
- Grading
- Practical setup

Course objectives

- Provide you with solid knowledge of:
  - Object technology principles and methods
  - The practice of object-oriented analysis, design and implementation
  - Some open issues
  - Some recent developments
  - Two specific technologies

Topics

- Quality issues
- Lifecycle
- Abstract Data Types
- Object model choices
- Inheritance techniques
- Design patterns
- Concurrent object-oriented computation
- Language mechanisms
- Persistence and O-O database
- Project management
- Genericity, typing issues, covariance
- …

Technologies

- Eiffel
- .NET

Guest lectures (tentative)

- Philippe Lahire, University of Nice (France):
  - Aspect-Oriented Programming (16 April 2003)
- …
- ETH assistants
  - Karine Arnout

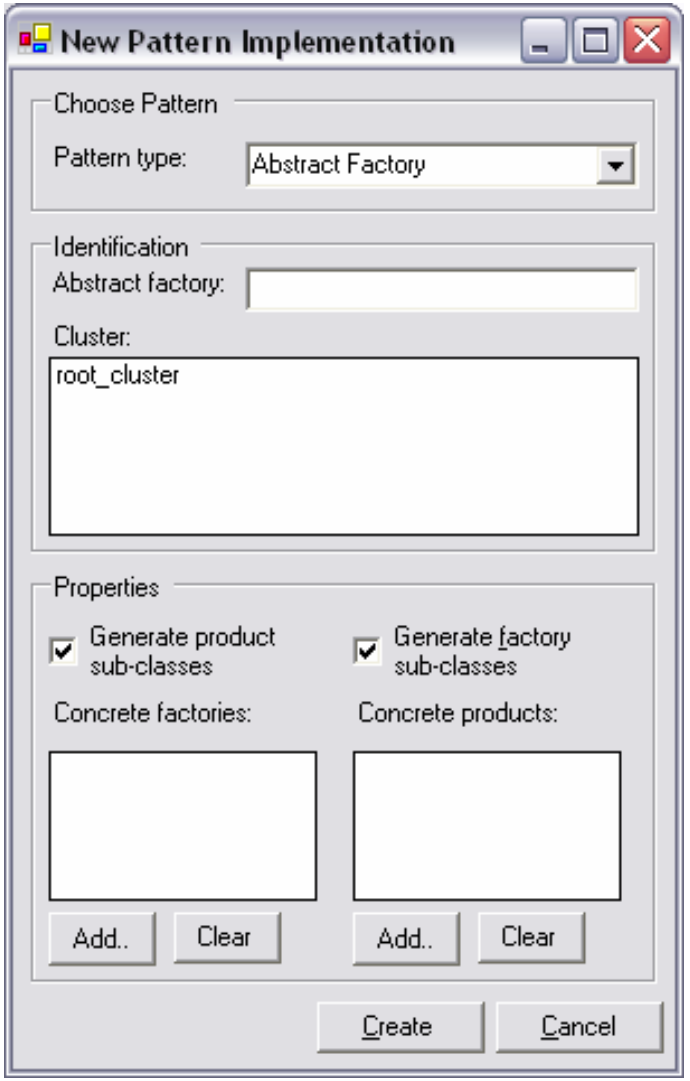
Textbook

- Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*. Prentice Hall, 1997.
  - Available from Ruth Bürkli, RZ-F8
  - Price: CHF 81.00

- Recommended:
  - Erich Gamma et al.: *Design Patterns*. Addison-Wesley, 1995.

Grading

- Exam (2h): 40%
  - 2 July 2003
- Project: 60%
  - Development of a "Pattern Wizard"
  - Deadline: 18 June 2003



Practical setup

- Course page:
  - http://se.inf.ethz.ch/teaching/ss2003/atot_ss2003.html
- Slides:
  - http://se.inf.ethz.ch/teaching/ss2003/atot_ss2003.html#slides

Practical setup (cont'd)

- Please send an email:
  - To: atot-course@se.inf.ethz.ch
  - Subject: ATOT course participant
  - Content:
    - Your name
    - Preferred email address
    - Status
      - Diplom student (semester), Ph.D. student, other.
    - Taking the course for credit or not.
  - Attach a picture (JPEG, GIF, PNG) if you wish

Practical setup (cont'd)

- If any questions / problems, contact:
  - Karine Arnout
  - http://se.inf.ethz.ch/people/arnout
  - Office: RZ-F7
  - Phone: 01 632 47 23

Before getting started…

- Please fill in the questionnaire:
  - Anonymous!
  - You have 10 minutes

Some words of warning

- Steps in reacting to O-O (from the preface to *Object-Oriented Software Construction*):
  - "(1) It's trivial;
  - (2) It's wrong;
  - (3) That's how I did it all along anyway."

- Beware of the "mOOzak" phenomenon.

Some words of warning (cont'd)


```
benefit_from_course is
        -- Make students succeed.
    require
        some_humility
    do
        all_exercises
    ensure
        OO_mastery_for_fun_and_profit
    end
```

Terminology

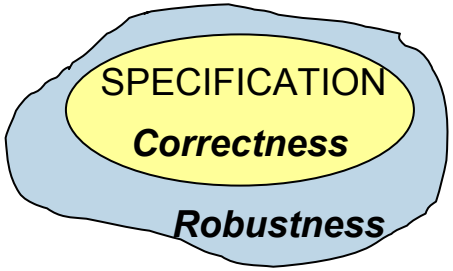
- I will be strict about terminology:
  - Endless confusions in the literature and in discussions.
  - Basic concepts have precise definitions — no justification whatsoever for such confusions.
  - Object technology is (in part) about bringing rational, scientific principles to software. No excuse for sloppy terminology.
- Alternative conventions will be mentioned when necessary.
- CHF 5 fine for saying “object” when meaning “class” (after lecture 4)

Agenda for today

- Introduction
- Quality issues
- Lifecycle

The goal: Software quality

- REUSABILITY
- EXTENDIBILITY
- RELIABILITY (Correctness + Robustness)
- PORTABILITY
- EFFICIENCY
- INTEGRITY
- …



- Correctness:
  - The ability of a software system to perform according to specification, in cases defined by the specification.
- Robustness:
  - The ability of a software system to react in a reasonable manner to cases not covered by the specification.

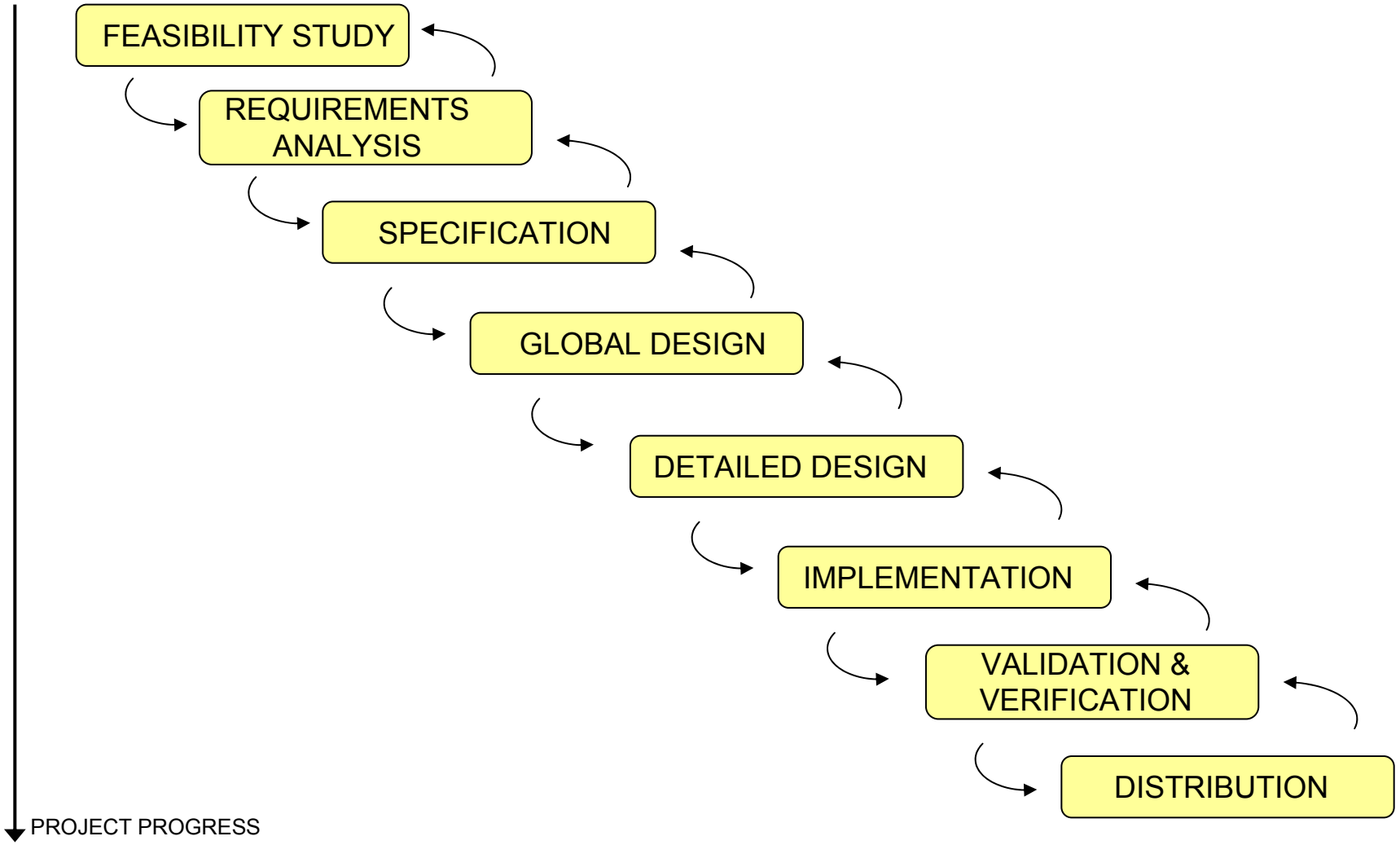
The challenge of software quality

- Reliability [correctness + robustness]:
  - It should be easier to build software that functions properly, and easier to guarantee what it does.
- Modularity [reusability + extendibility]:
  - We should build *less* software!
  - Software should be easier to modify.

Agenda for today

- Introduction
- Quality issues
- Lifecycle

The waterfall model of the lifecycle

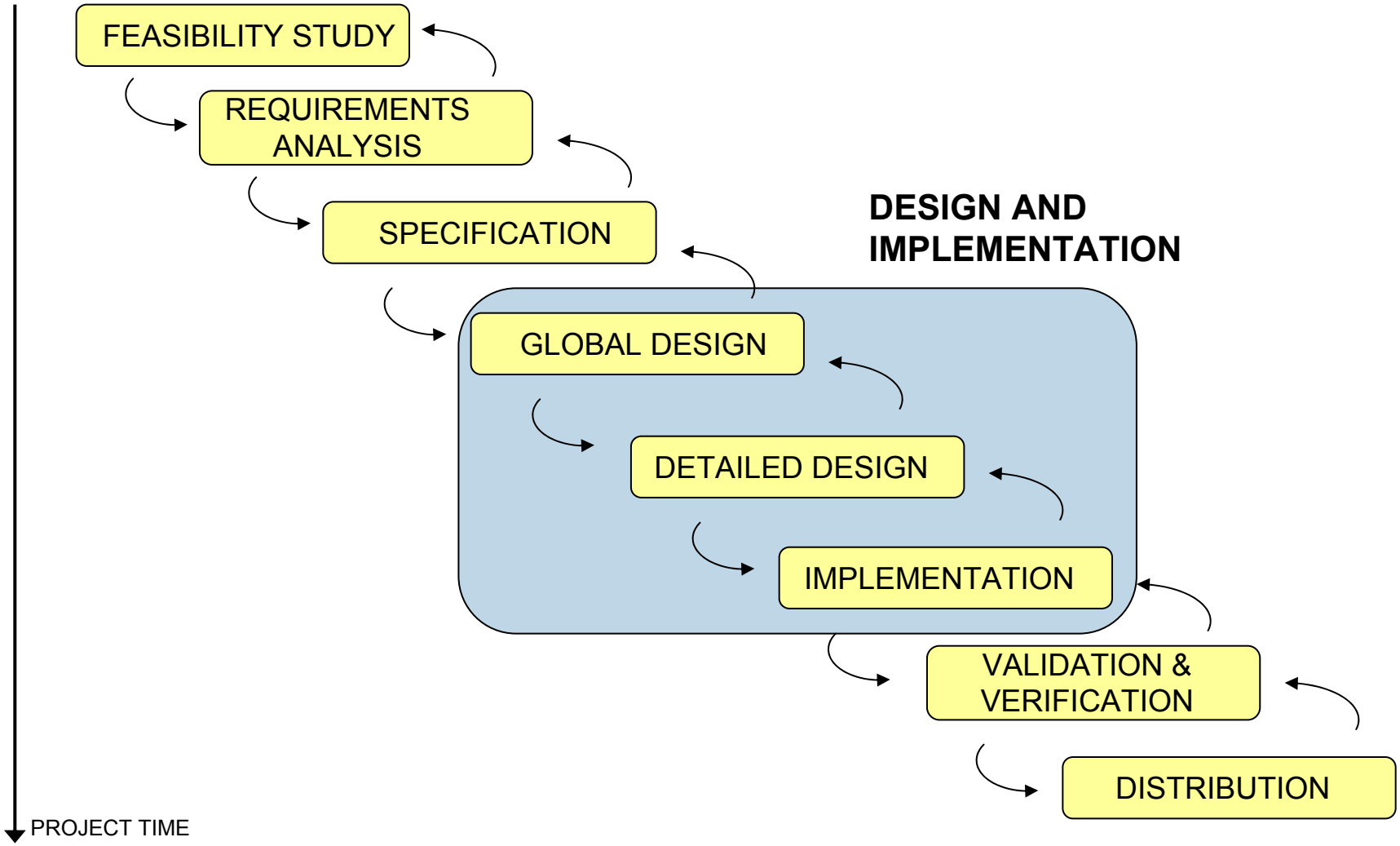


Arguments for the waterfall

(After B.W. Boehm: *Software engineering economics*)

- The activities are necessary.
  - (But: merging of middle activities.)
- The order is the right one.

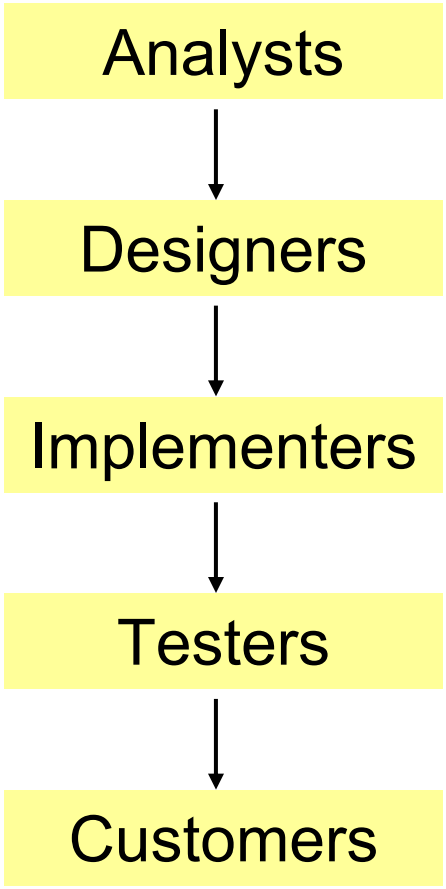
The waterfall model of the lifecycle



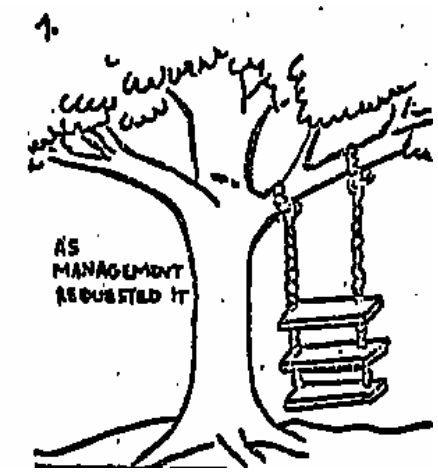
Problems with the waterfall

- Late appearance of actual code.
- Lack of support for requirements change — and more generally for extendibility and reusability.
- Lack of support for the maintenance activity (70% of software costs?).
- Division of labor hampering Total Quality Management.
- Impedance mismatches.
- Highly synchronous model.

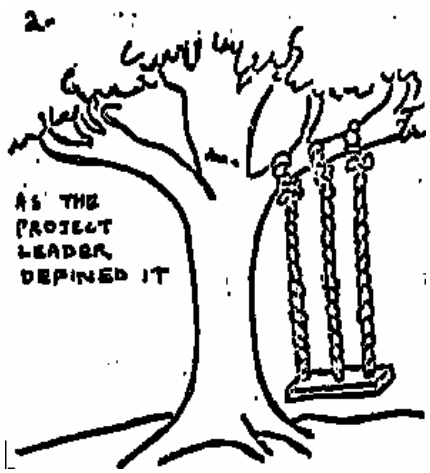
Quality control?



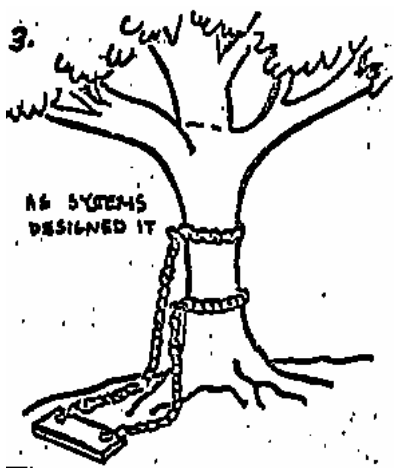
Impedance mismatches



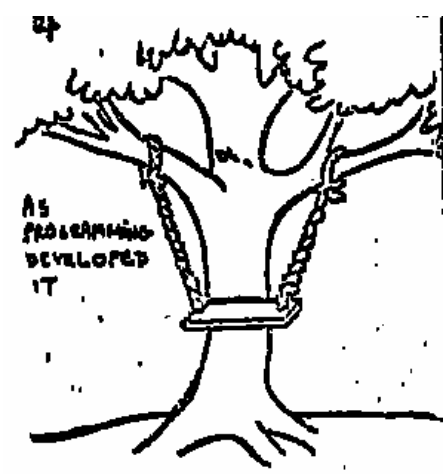
As Management requested it.



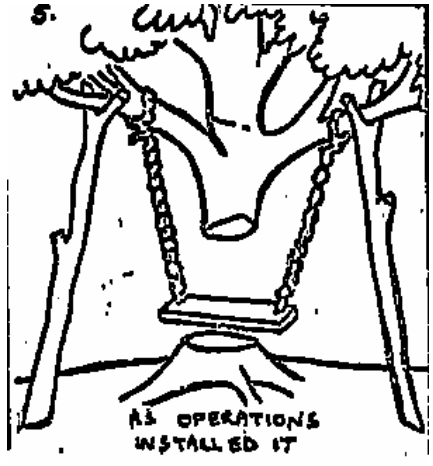
As the Project Leader defined it.



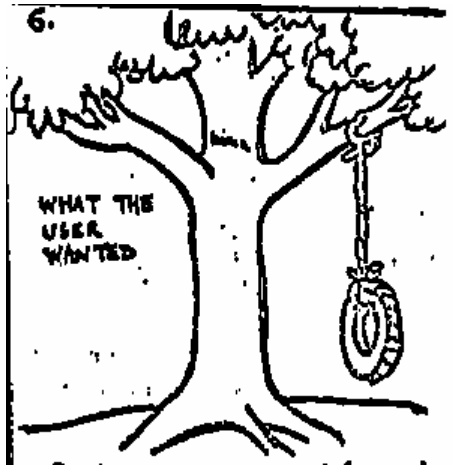
As Systems designed it.



As Programming developed it.



As Operations installed it.



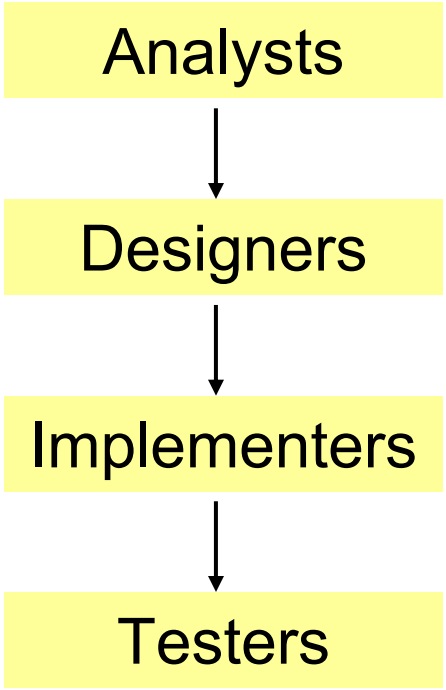
What the user wanted.

(Pre-1970 cartoon; origin unknown)

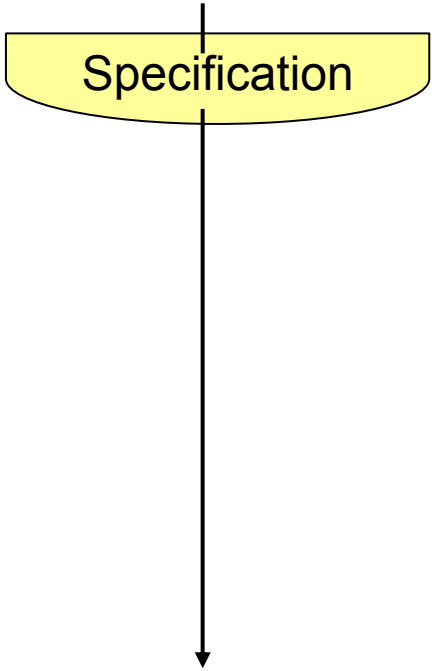
The escherfall (Spiral)

M.C Escher:
Waterval

Tasks



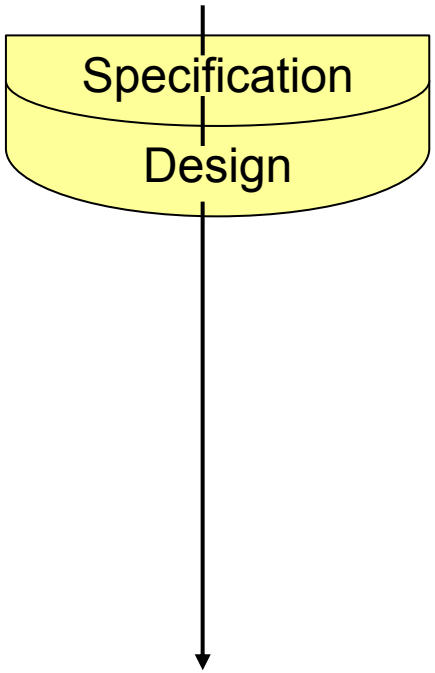
Seamless development



TRANSACTION, PLANE, CUSTOMER, …

Example classes

Seamless development

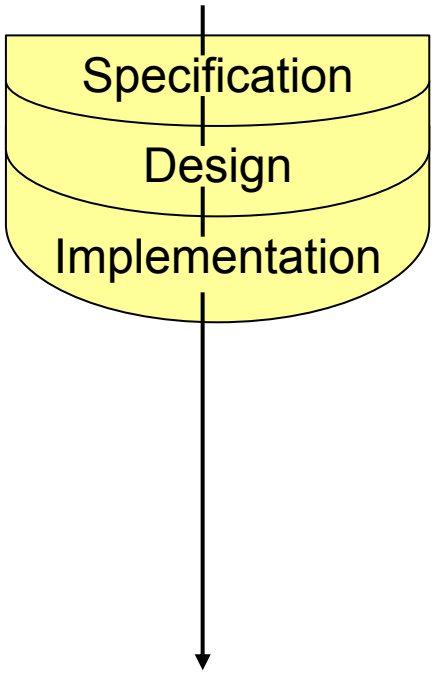


TRANSACTION, PLANE, CUSTOMER, …

STATE, USER_COMMAND, …

Example classes

Seamless development



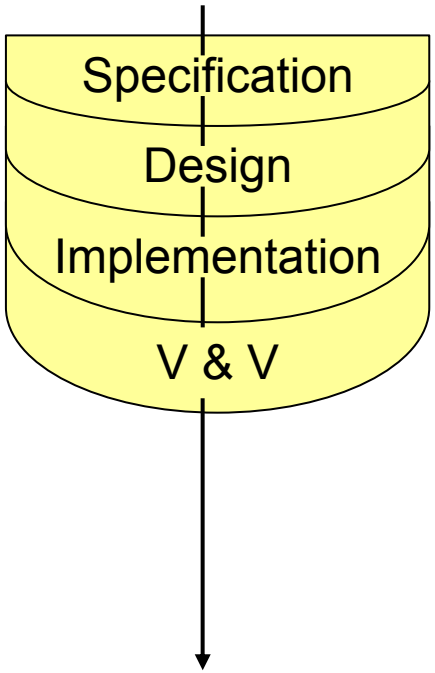
TRANSACTION, PLANE, CUSTOMER, …

STATE, USER, …

HASH_TABLE, LINKED_LIST…

Example classes

Seamless development



TRANSACTION, PLANE, CUSTOMER, …

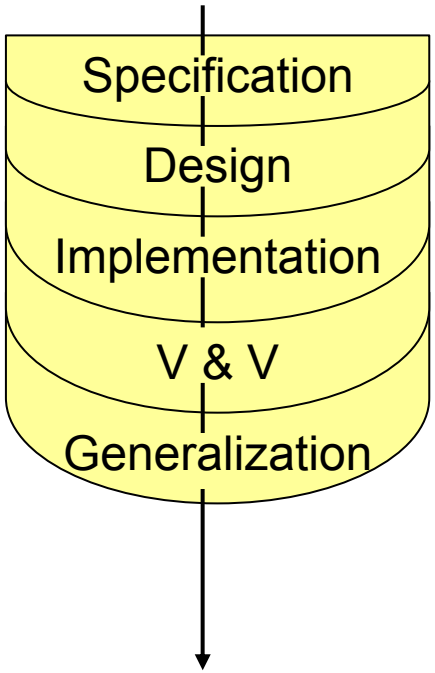
STATE, USER, …

HASH_TABLE, LINKED_LIST…

TEST_DRIVER, …

Example classes

Seamless development



TRANSACTION, PLANE, CUSTOMER, …

STATE, USER, …

HASH_TABLE, LINKED_LIST…

TEST_DRIVER, …

Example classes

Analysis classes


```
deferred class VAT
inherit
    TANK
feature
    in_valve, out_valve: VALVE
    fill is
                    -- Fill the vat.
            require
                    in_valve.open
                    out_valve.closed
            deferred
            ensure
                    in_valve.closed
                    out_valve.closed
                    is_full
            end
    empty, is_full, is_empty, gauge, maximum, ... [Other features] ...
invariant
    is_full = (gauge >= 0.97 * maximum)  and  (gauge <= 1.03 * maximum)
end
```

Precondition

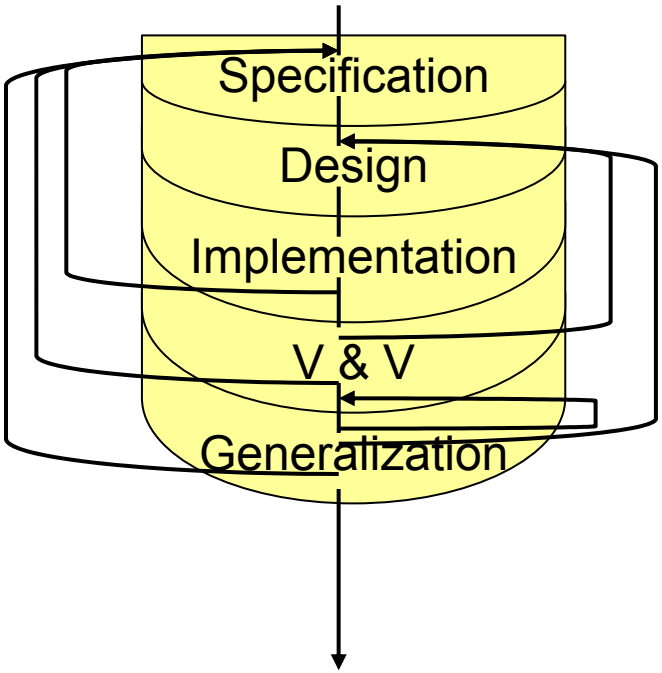
-- i.e. specified only.

-- not implemented.

Postcondition

Class invariant

Reversibility



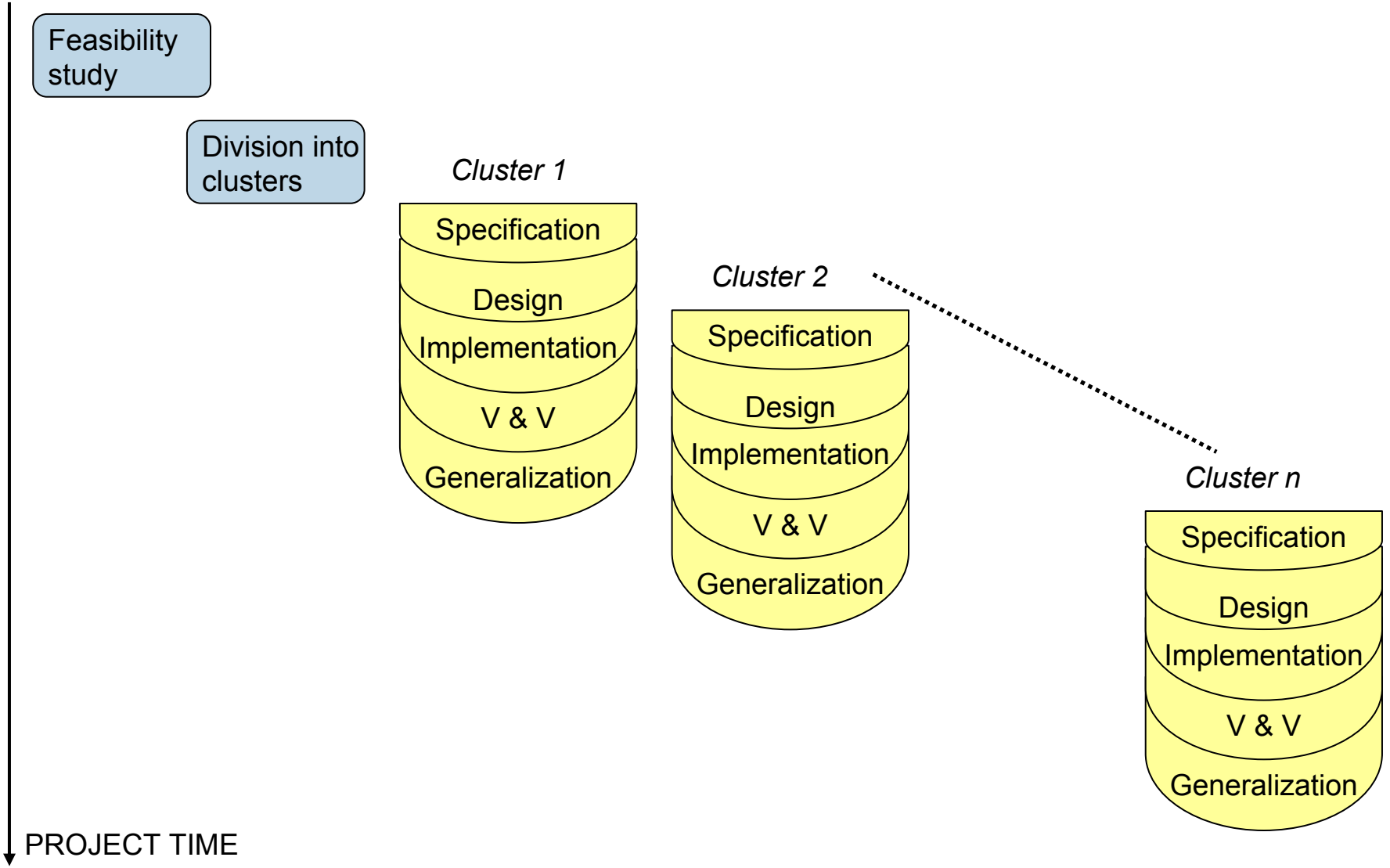
Seamless development

- Use consistent notation from analysis to design, implementation and maintenance.
- Advantages:
  - Smooth process. Avoids gaps (improves productivity, reliability).
  - Direct mapping from problem to solution, i.e. from software system to external model.
  - Better responsiveness to customer requests.
  - Consistency, ease of communication.
  - Better interaction between users, managers and developers.

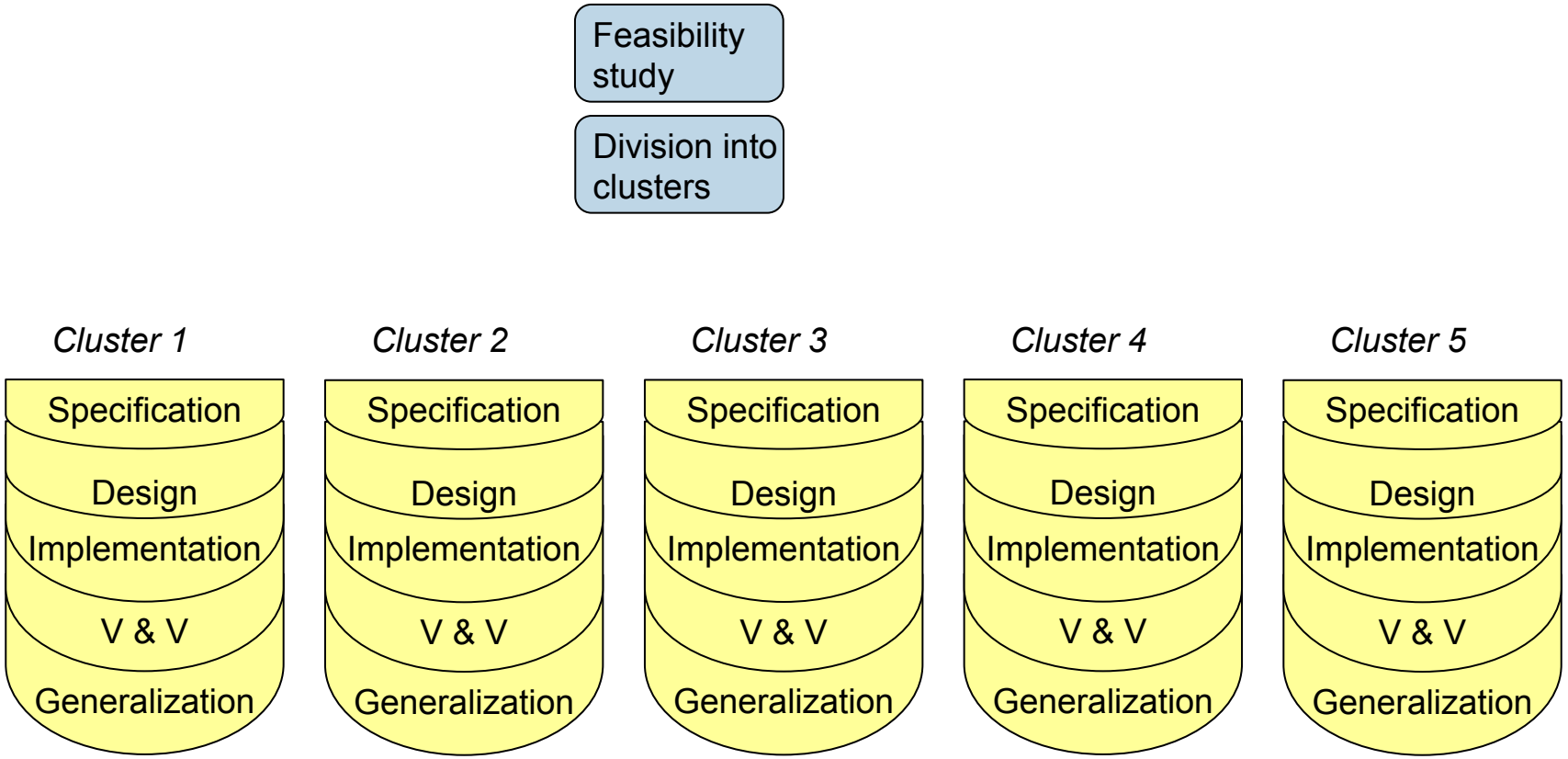
Single model principle

- Use a single base for everything: analysis, design, implementation, documentation...
- Use tools to extract the appropriate views.

The cluster model

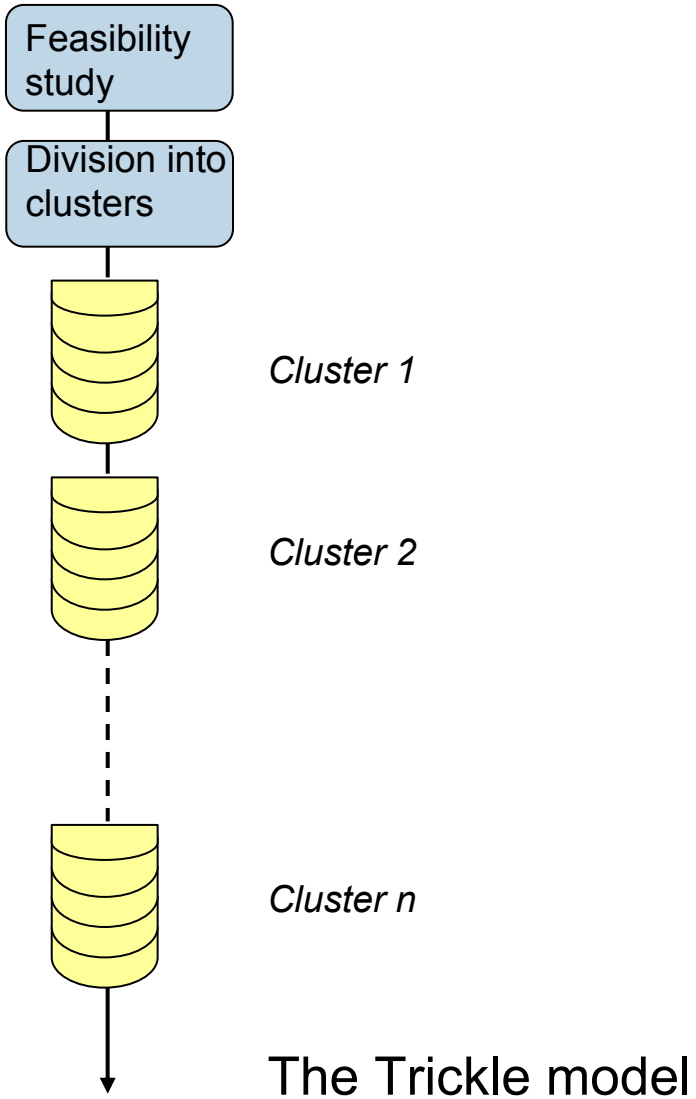


The cluster model: extreme variants (1)

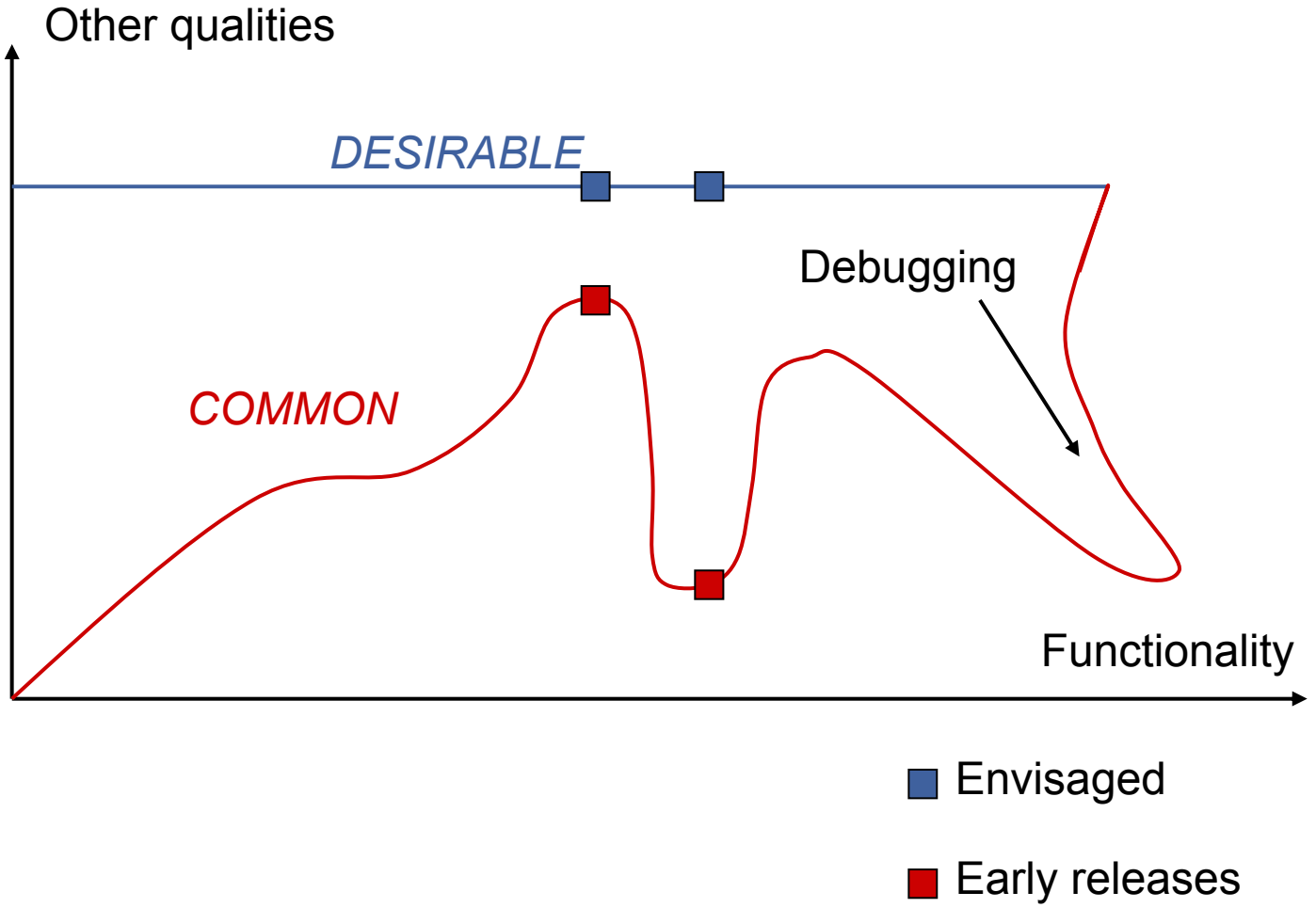


"Clusterfall"

The cluster model: extreme variants (2)



Quality goals: the Osmond curves



Cluster development

- Bottom-up development: from the most general clusters (providing utility functions) to the most application-specific ones.
- Flexible scheduling of clusters – depending on resources, team experience, customer and management demands. Waterfall is one extreme; "trickle" is the other.
- Sub-lifecycle sequencing: specification, design and implementation, validation, generalization.
- Relations between clusters: each cluster may be a client of lower-level ones.

Reading assignment

- For Monday 7 April 2003: OOSC2 chapters
  - Chapter 1: Software quality
  - Chapter 28: The software construction process

End of lecture 1