Chair of Software Engineering

# Software Architecture

Bertrand Meyer, Carlo A. Furia, Martin Nordio

ETH Zurich, February-May 2011

## Lecture 2: The software lifecycle

## Software lifecycle models

Describe an overall distribution of the software construction into tasks, and the ordering of these tasks

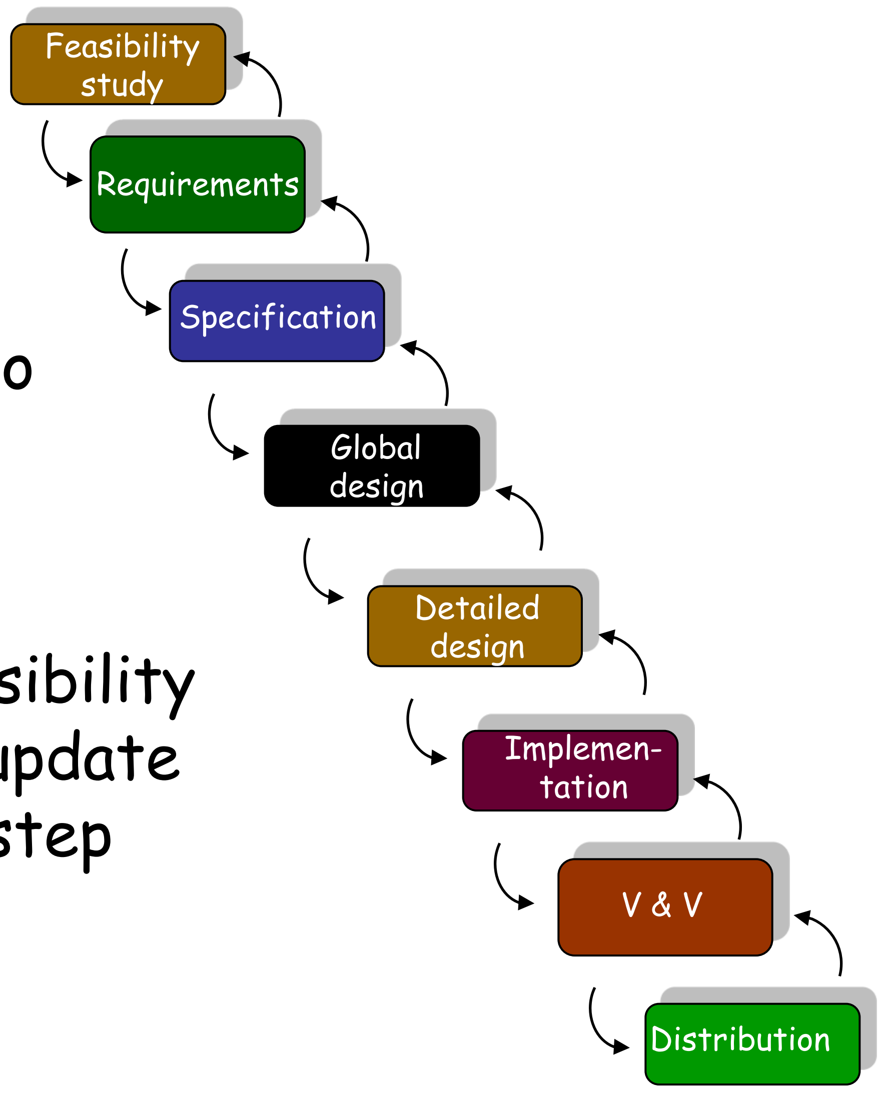
They are models in two ways:

- Provide an abstracted version of reality
- Describe an ideal scheme, not always followed in practice

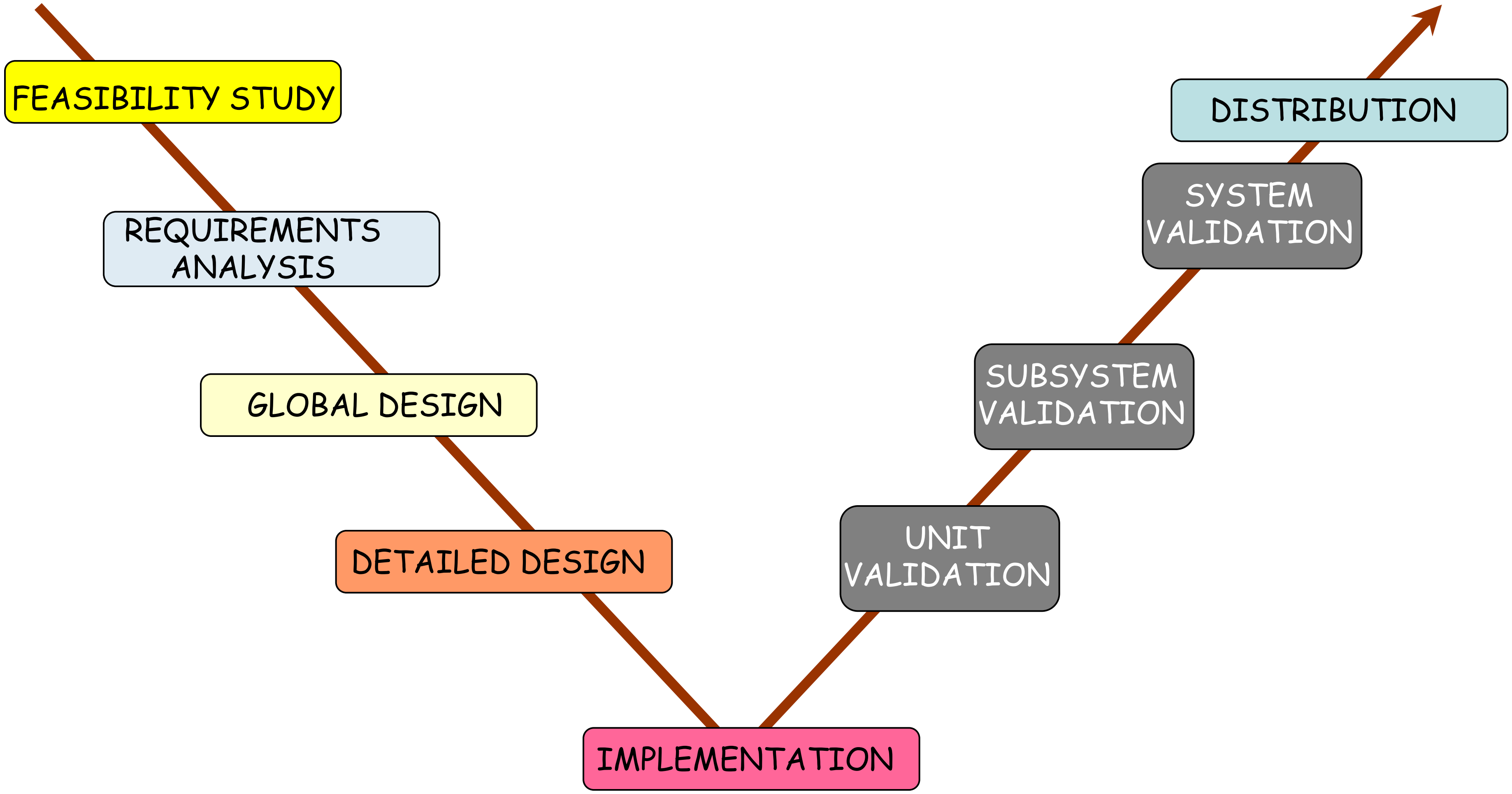
## Lifecycle: the waterfall model

Royce, 1970 (original article actually presented the model to *criticize* it!)

Succession of steps, with possibility at each step to question and update the results of the preceding step



# A V-shaped variant



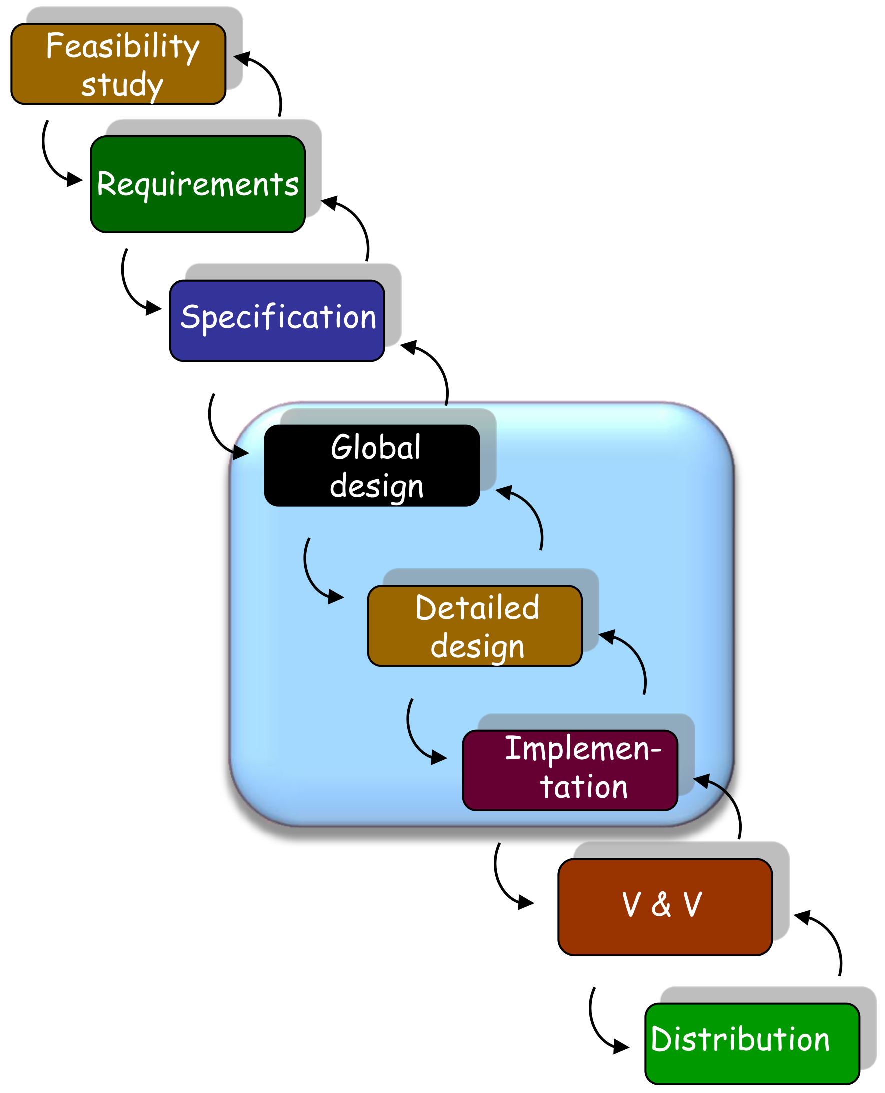
## Arguments for the waterfall

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

- The activities are necessary
  - (But: merging of middle activities)

- The order is the right one.

## Merging of middle activities



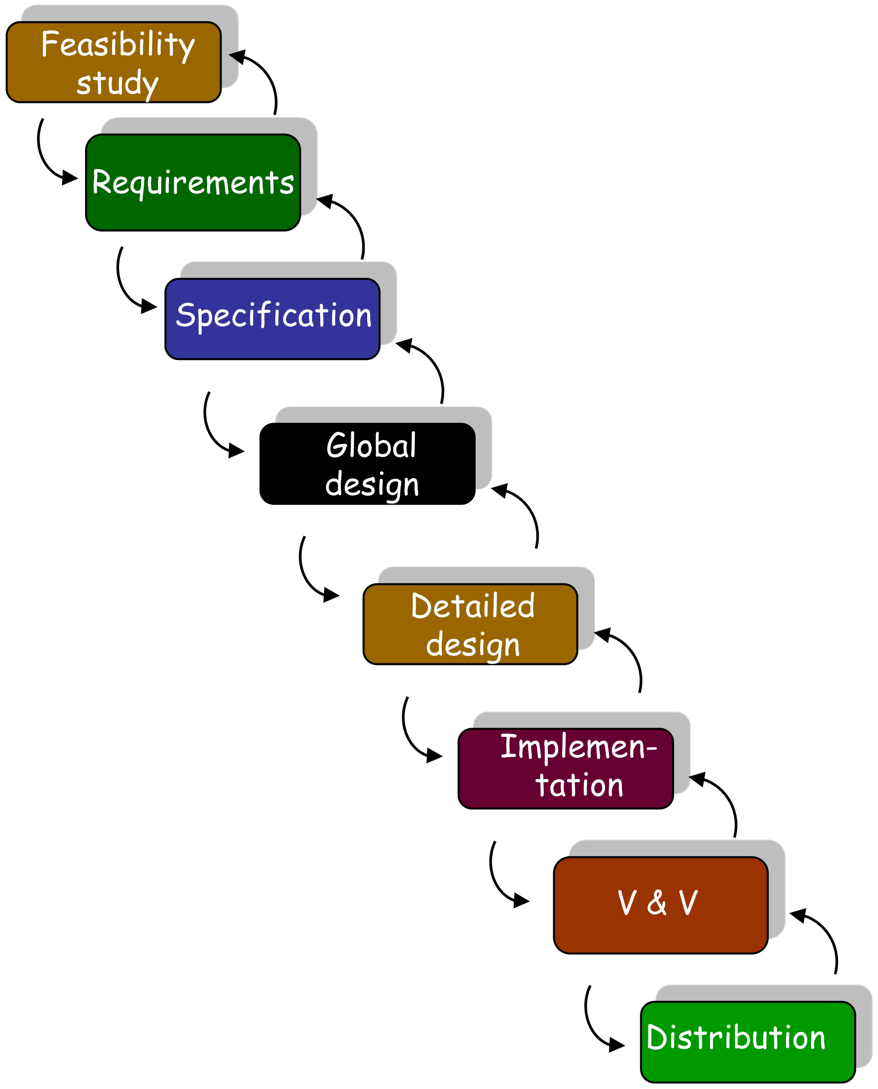
## Arguments for the waterfall

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

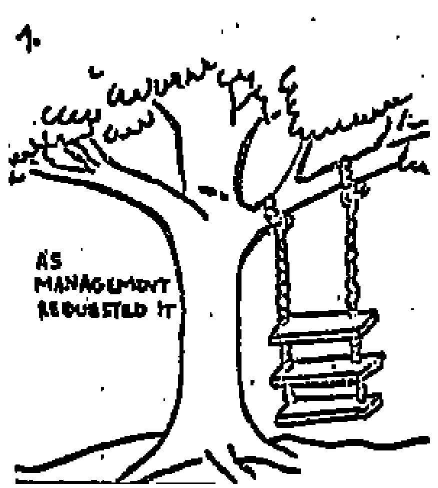
- The activities are necessary
  - (But: merging of middle activities)
- The order is the right one.

## Problems with the waterfall

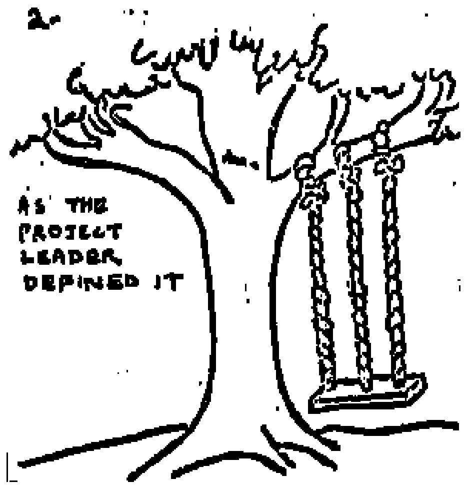
- Late appearance of actual code.
- Lack of support for requirements change — and more generally for extendibility and reusability
- Lack of support for the maintenance activity (70% of software costs?)
- Division of labor hampering Total Quality Management
- Impedance mismatches
- Highly synchronous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



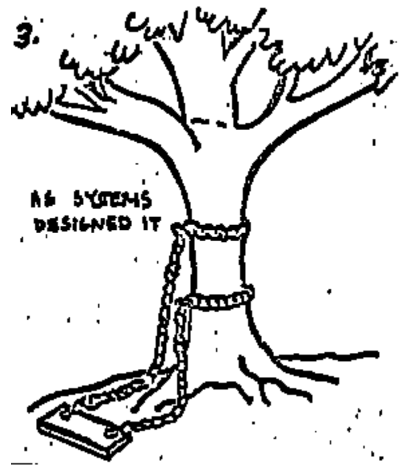
# Lifecycle: “impedance mismatches”



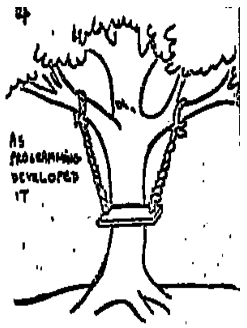
*As Management requested it*



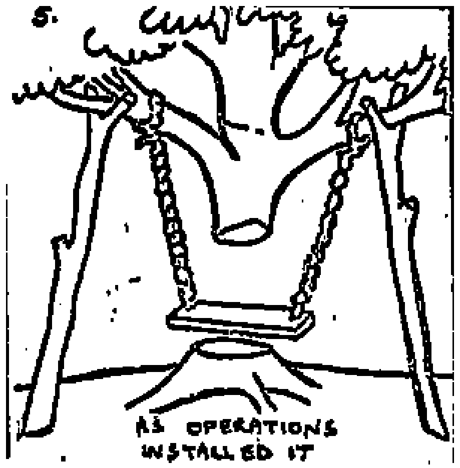
*As the Project Leader defined it*



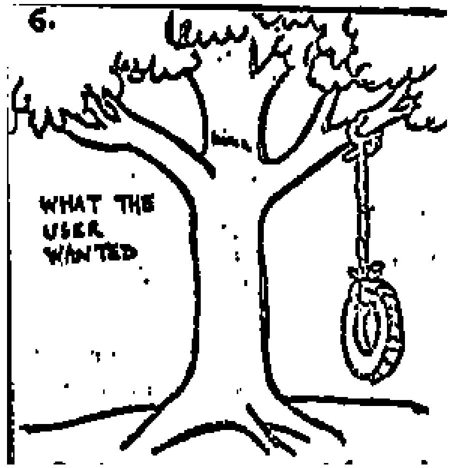
*As Systems designed it*



*As Programming developed it*



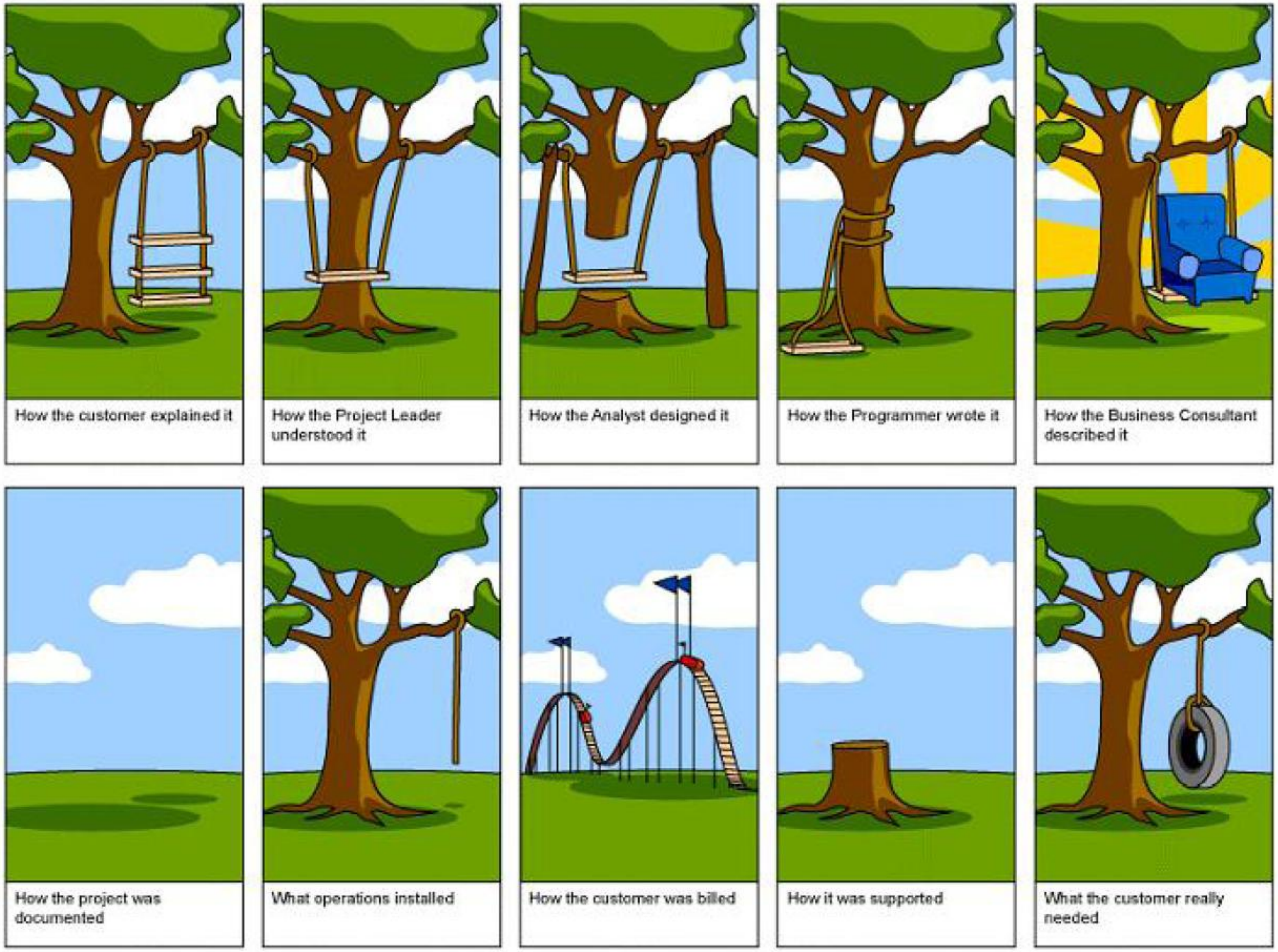
*As Operations installed it*



*What the user wanted*

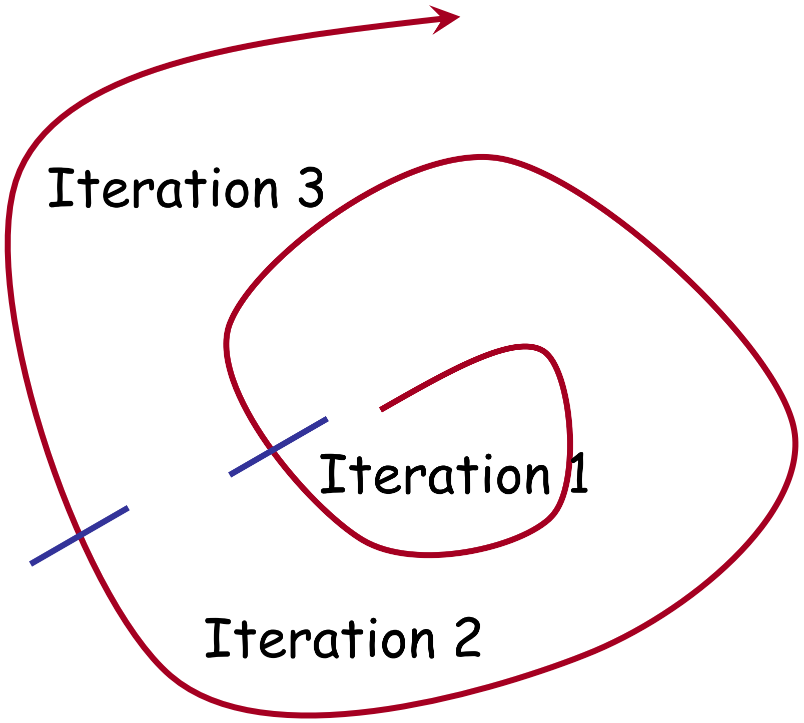
(Pre-1970 cartoon; origin unknown)

# A modern variant

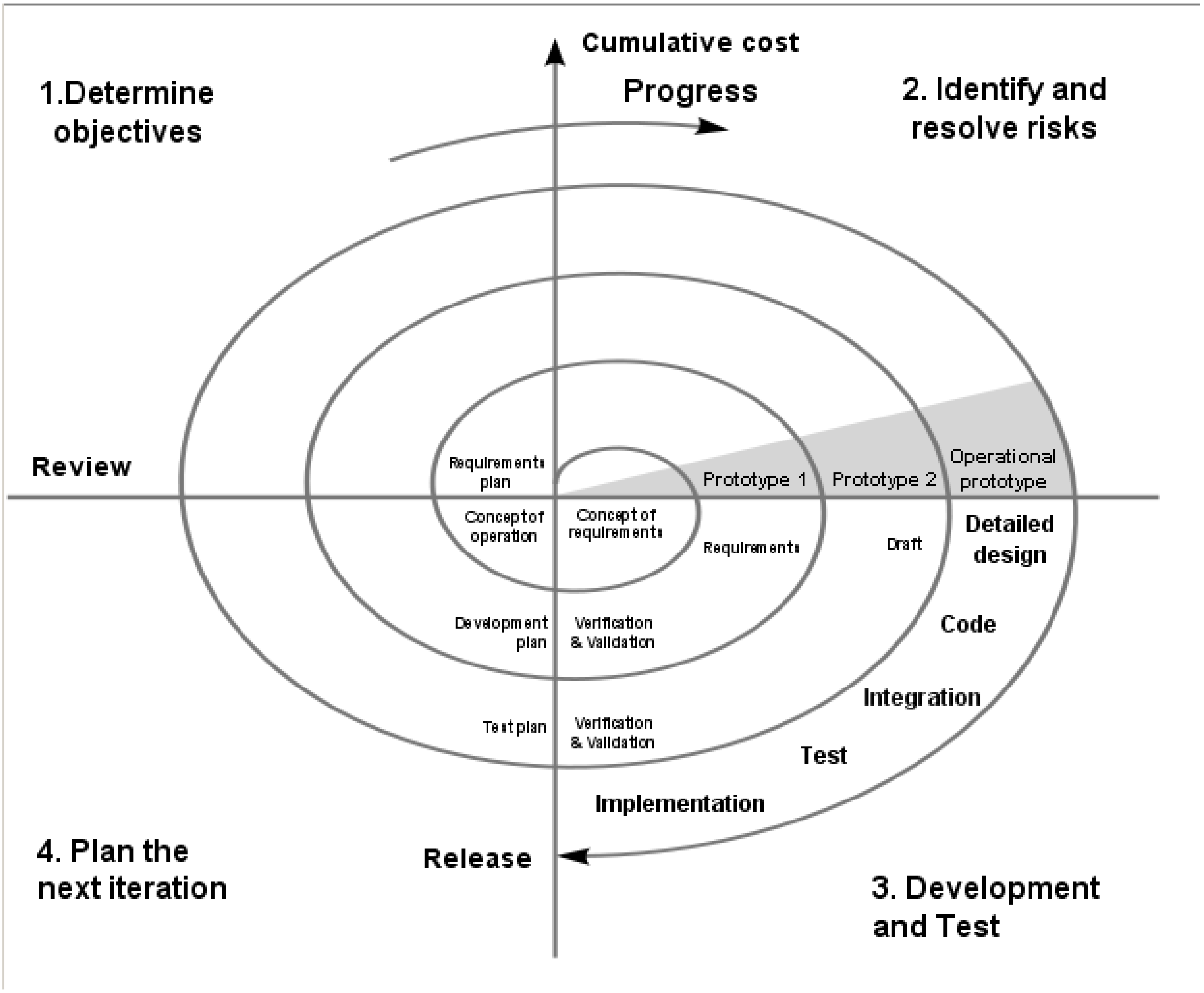

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

## The spiral model (Boehm)

Apply a waterfall-like approach to successive prototypes



## The Spiral model

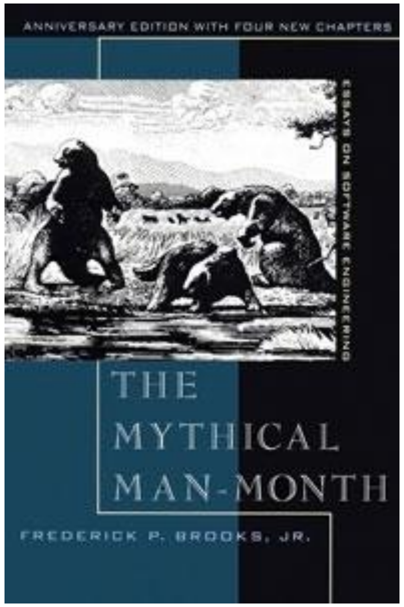


## “Prototyping” in software

The term is used in one of the following meanings:

- 1. Experimentation:
  - Requirements capture
  - Try specific techniques: GUI, implementation (“buying information”)
- 2. Pilot project
- 3. Incremental development
- 4. Throw-away development

(Fred Brooks, *The Mythical Man-Month*, 1975: “Plan to throw one away, you will anyhow”).


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE
MYTHICAL
MAN-MONTH
FREDERICK P. BROOKS, JR.

## The problem with throw-away development

Software development is hard because of the need to reconcile conflicting criteria, e.g. portability and efficiency

A prototype typically sacrifices some of these criteria

Risk of shipping the prototype

In the 20th-anniversary edition of his book (1995), Brooks admitted that "plan to throw one away" is bad advice

## Seamless, incremental development

Seamless development:

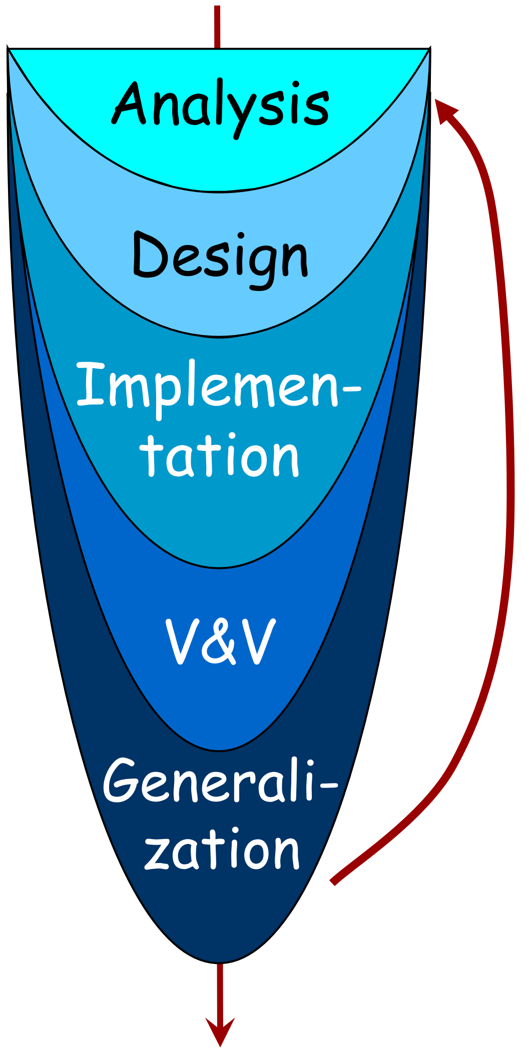
- Single set of notation, tools, concepts, principles throughout
- Continuous, incremental development
- Keep model, implementation and documentation consistent

Reversibility: can go back and forth

These are in particular some of the ideas behind the Eiffel method

## Seamless development

- Single notation, tools, concepts, principles
- Continuous, incremental development
- Keep model, implementation and documentation consistent
- Reversibility: go back and forth



Example classes:

*PLANE, ACCOUNT, TRANSACTION…*

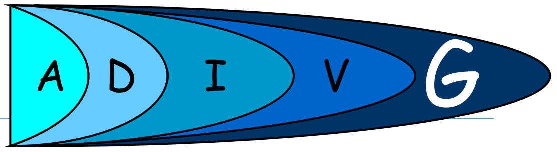
*STATE, COMMAND…*

*HASH_TABLE…*

*TEST_DRIVER…*

*TABLE…*

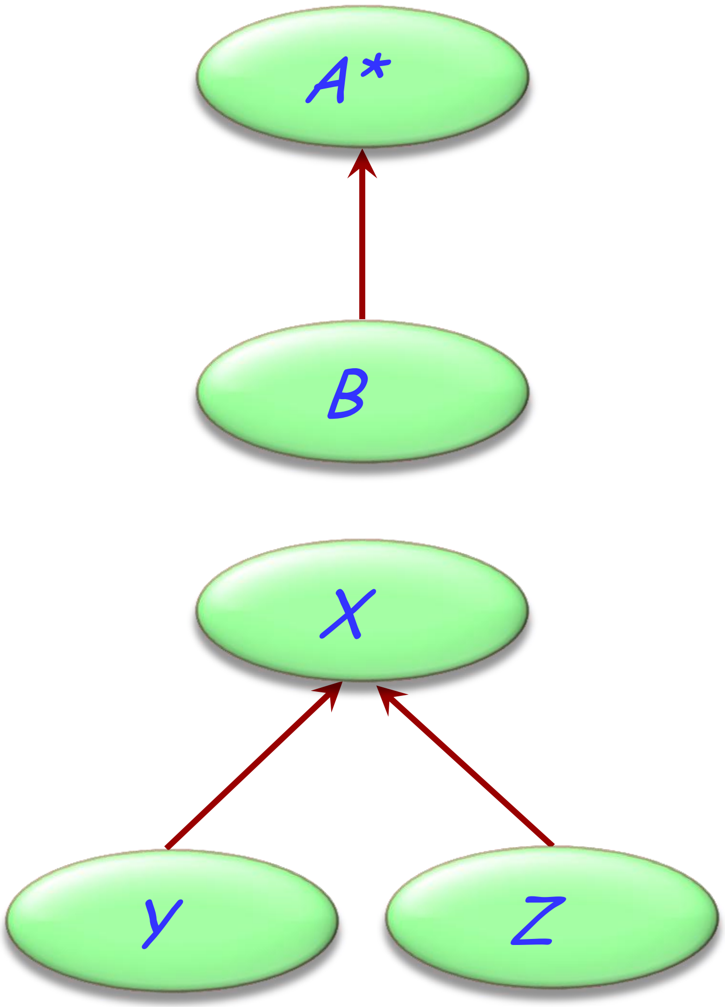
## Generalization



Prepare for reuse. For example:

- Remove built-in limits
- Remove dependencies on specifics of project
- Improve documentation, contracts...
- Abstract
- Extract commonalities and revamp inheritance hierarchy

Few companies have the guts to provide the budget for this

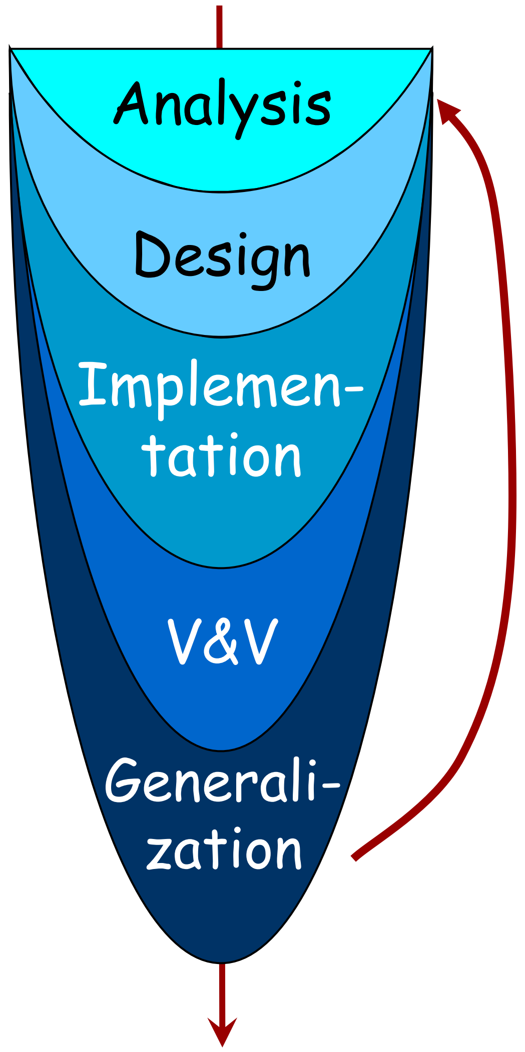

$$\begin{tikzcd} & A^* & \\ & B \arrow[u] & \\ & X & \\ Y \arrow[ur] & & Z \arrow[ul] \end{tikzcd}$$

## Finishing a design

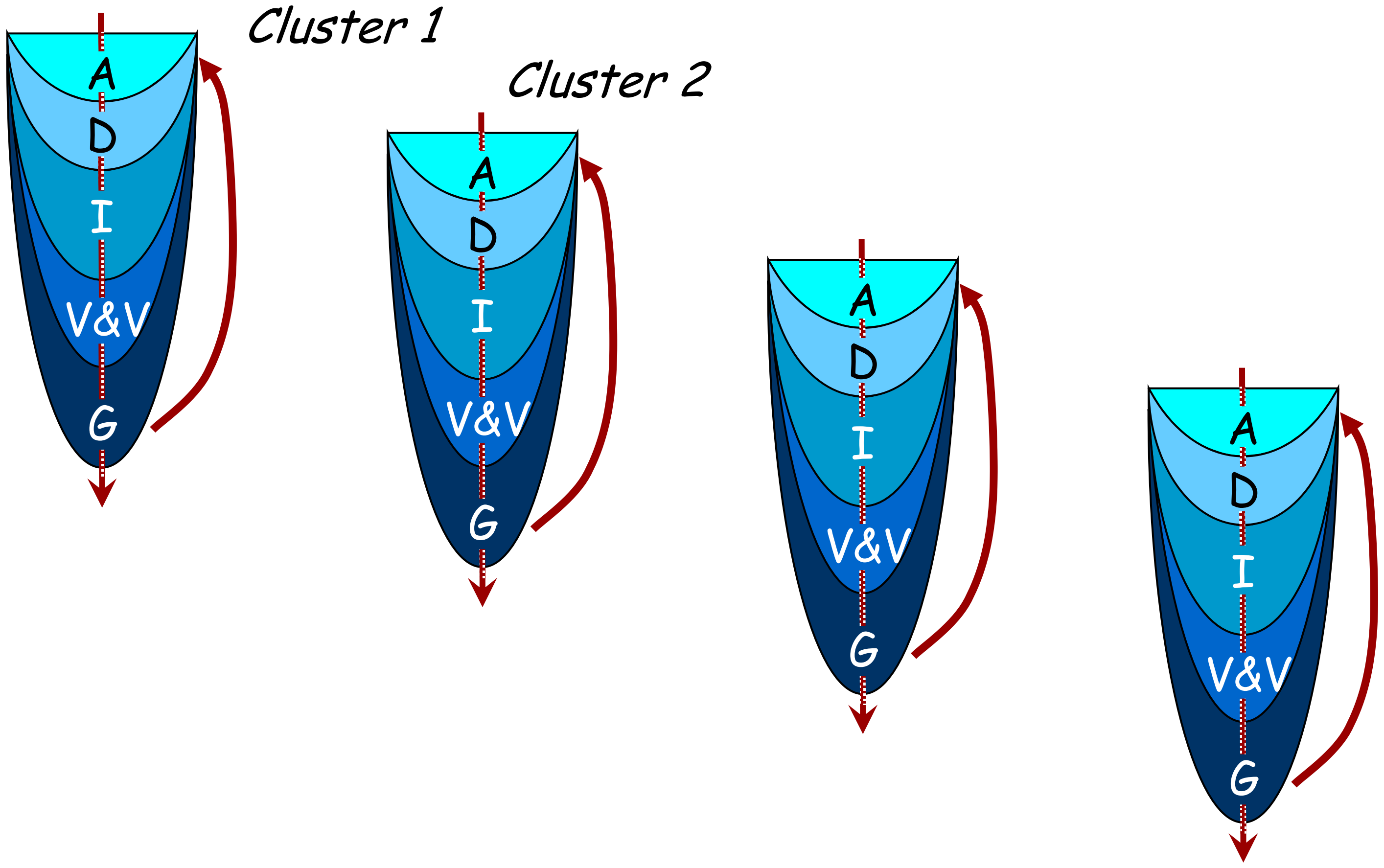
*It seems that the sole purpose of the work of engineers, designers, and calculators is to polish and smooth out, lighten this seam, balance that wing until it is no longer noticed, until it is no longer a wing attached to a fuselage, but a form fully unfolded, finally freed from the ore, a sort of mysteriously joined whole, and of the same quality as that of a poem. It seems that perfection is reached, not when there is nothing more to add, but when there is no longer anything to remove.*

(Antoine de Saint-Exupéry,
*Terre des Hommes*, 1937)

## Reversibility

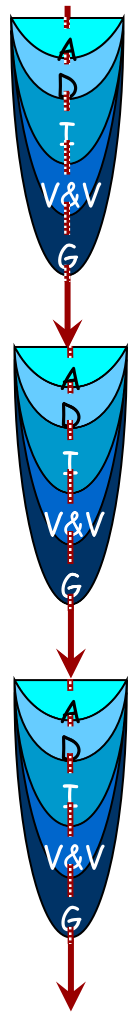


# The cluster model



# Extremes

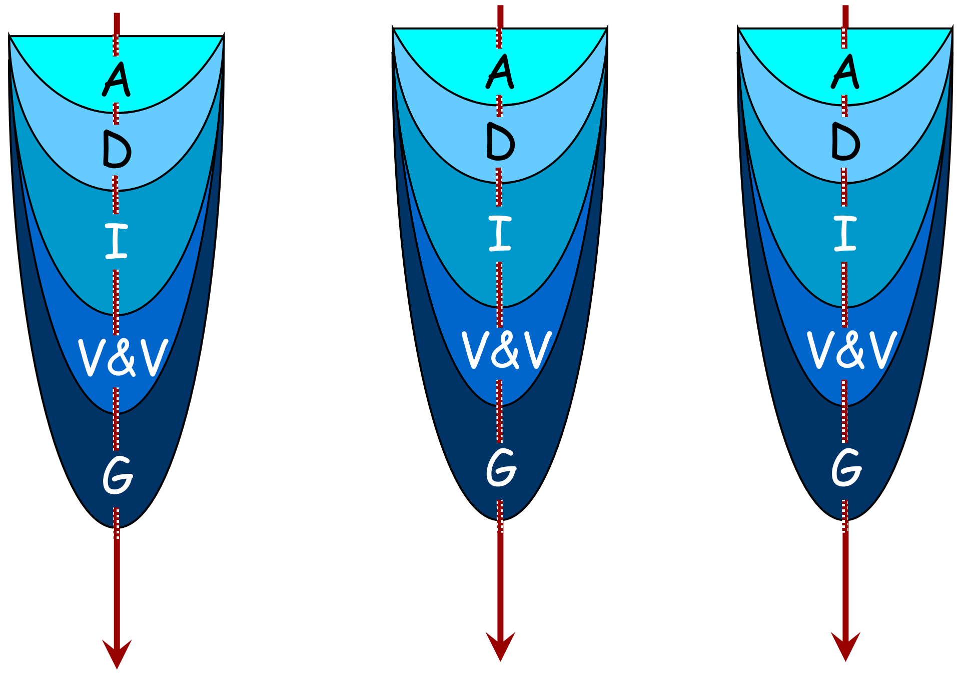
"Trickle"



Cluster 1

Cluster 2

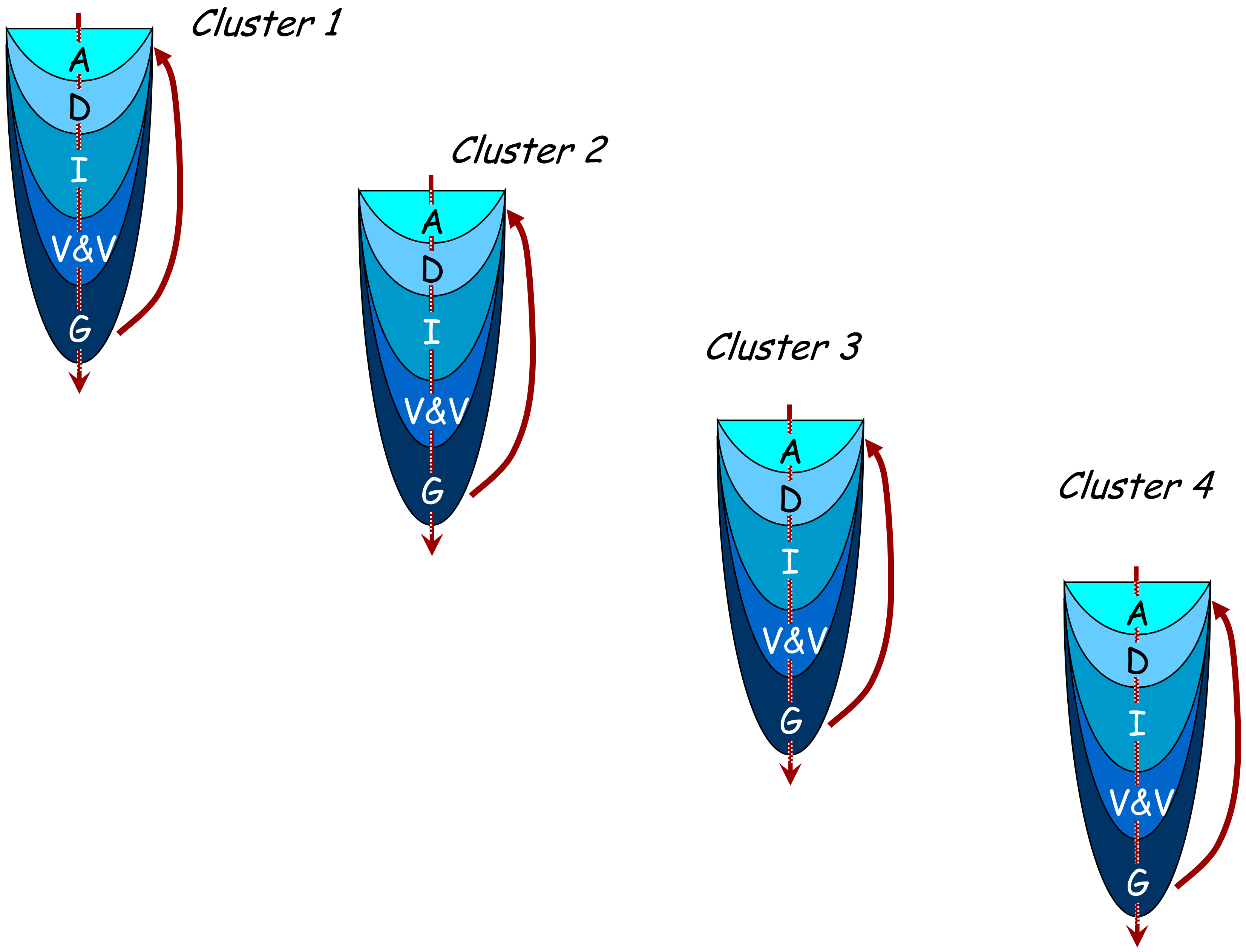
"Clusterfall"



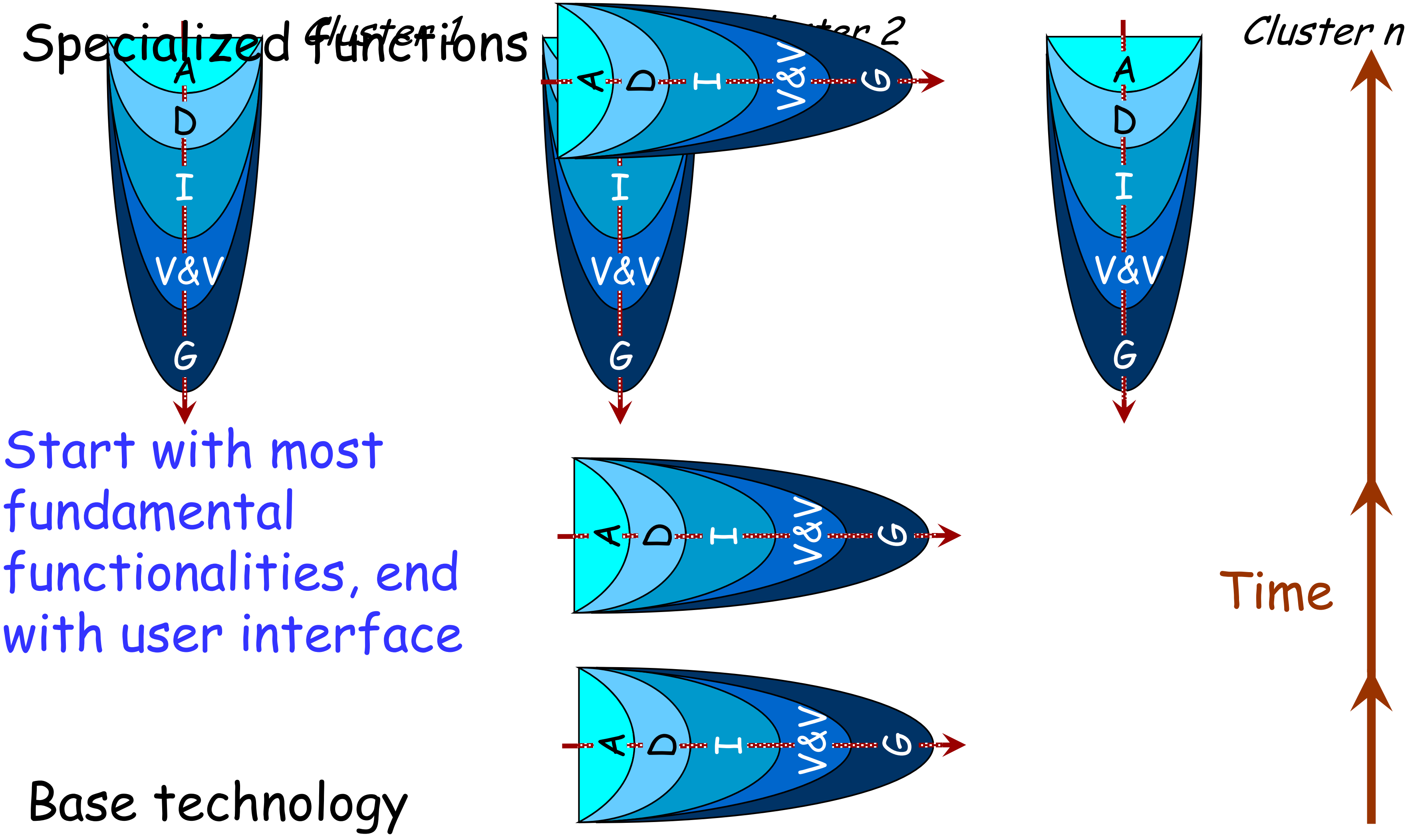
Cluster 1

Cluster 2

## Dynamic rearrangement



## Bottom-up order of cluster development



Start with most fundamental functionalities, end with user interface

## Seamless development with EiffelStudio

Diagram Tool

- System diagrams can be produced automatically from software text
- Works both ways: update diagrams or update text – other view immediately updated

No need for separate UML tool

Metrics Tool

Profiler Tool

Documentation generation tool

...

## Summary

Software development involves fundamental tasks such as requirements, design, implementation, V&V, maintenance…

Lifecycle models determine how they will be ordered

The Waterfall is still the reference, but many variants are possible, e.g. Spiral, Cluster

Seamless development emphasizes the fundamental unity of the software process