

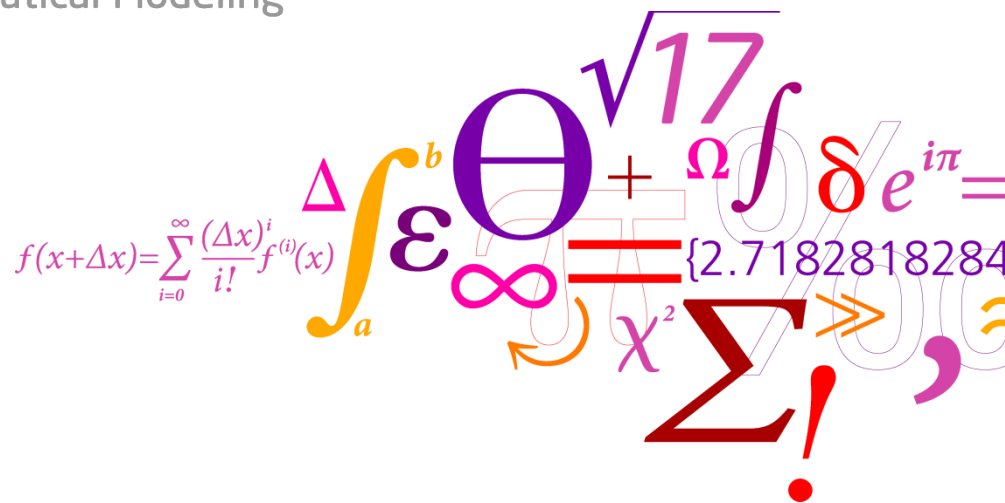
Software Engineering 2

A practical course in software engineering

Ekkart Kindler

DTU Informatics

Department of Informatics and Mathematical Modeling



IV. Working together

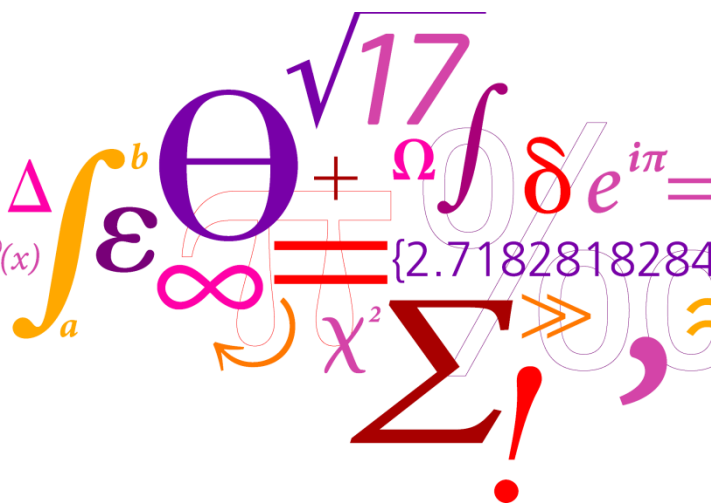
IV.1. Management

DTU Informatics

Department of Informatics and Mathematical Modeling

Some slides repeated from
lecture 4!

On the side: Discussion of
issues (based on project)

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$


(Software) Management

Planning, organizing, leading, monitoring and controlling the software development process.

Define goals and make sure that they will be achieved

General goals:

- Increase productivity

decrease
development costs

- Increase quality

Increase
product value

In short:

Make a profit and increase it!

Software Management requires

- short-term
(within a project or even within a phase)
and
- long-term
(spanning more projects)

measures

Short- and long-term
measures often have
almost opposite effect.

■ Planning

- set goal
- set dates
- define course of action
- assign resources
- ...

■ Organization

- assign tasks
- define organisation structures
- assign responsibilities
- ...

■ **Leading**

- lead and motivate team members
- improve communication
- solve conflicts
- ...

■ **Monitoring and controlling**

- check progress
- identify problems (early)
- produce relief
- ...

- Planning
- Organization
- Leading
- Monitoring & controlling



On the following slides some management "issues" are raised, for triggering a discussion!

The slides do not cover "the answers"! Sometimes "THE answer" does not exist.

- Measure / picture / control progress of project
- Predict cost / time

- Minimise risk
(minimise negative impact of risk)

- Manage size and complexity
- Minimize complicatedness (see next slide)

- Balance workload

- Complexity is inherent to the problem solved
- "Complicatedness is difficulty that serves no purpose ..."
<http://picture-poems.com/week4/complexity.html>

- Understand the problem
- Define the solution
- Design the solution
- Implement the solution

”WHY”

”WHAT” C

D

”HOW”

I

O

- Conceiving
- Designing
- Implementing
- Operating

From lecture 4.

Process models are the distilled experience of project plans of successful software projects

they are idealized and abstract from (many) details

this is their strength (and maybe also their weakness)

- adaptive (e.g. agile, Scrum, evolutionary, ...)

vs.

- predictive (e.g. waterfall, V-model, RUP, ...)

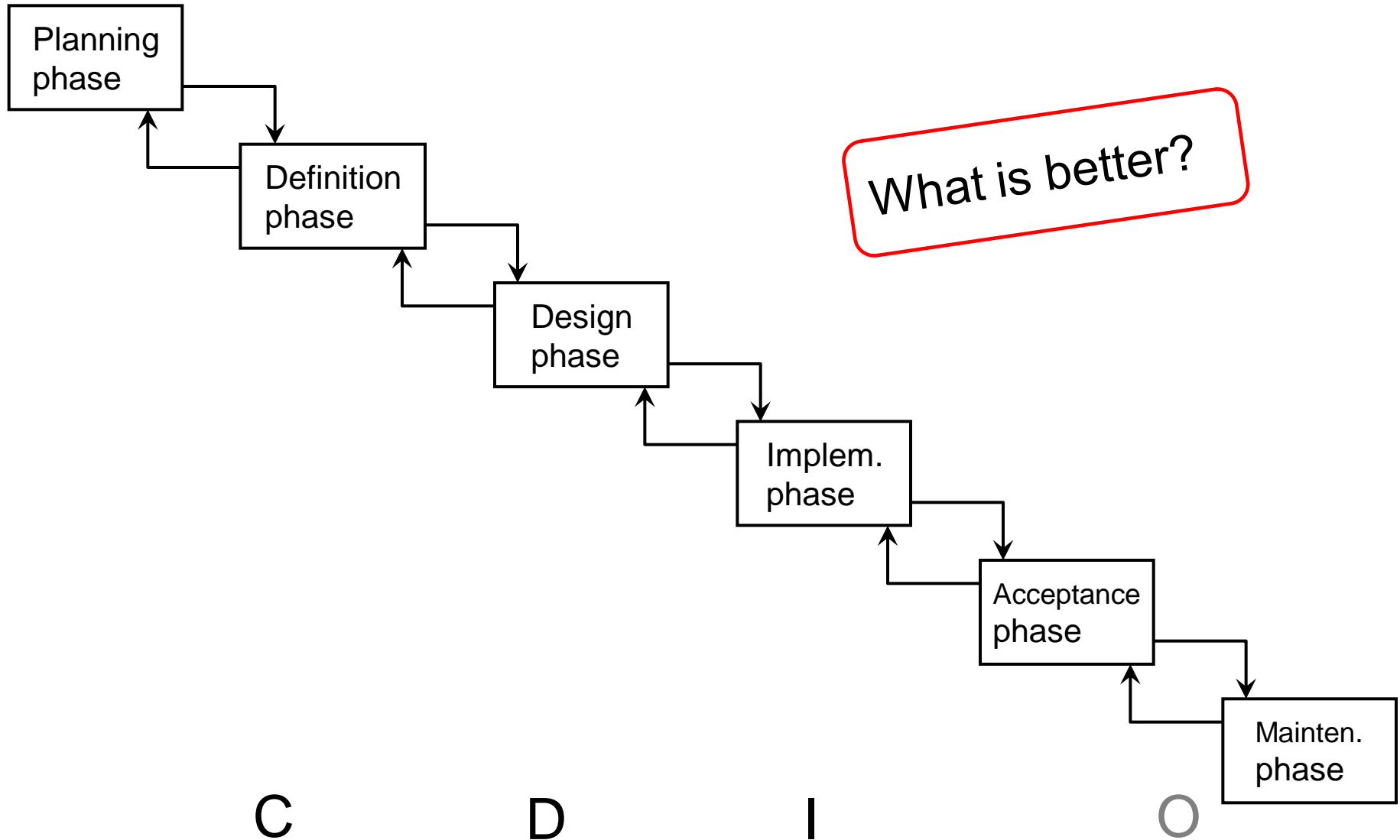
„We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.“

Kent Beck et al. 2001

Waterfall Model



- I without C/D
- C/D goes on forever (I never starts)
→ requirements are illusive

Observation:

- Often C/D needs or is inspired by I
(only an first implementation reveals what we really wanted)
→ "co-evolution" of understanding of problem
solution

We cannot afford to be
dogmatic about methods!

- Understand the problem why
- Define solution what
- Design & implement solution how

The difference is in how to slice it / work through it.

Why?

- Understand what there is already!
- Understand **why** this is not sufficient!

What?

- What can be done about it!
→ define (better) solution

On the dimensions of software documents

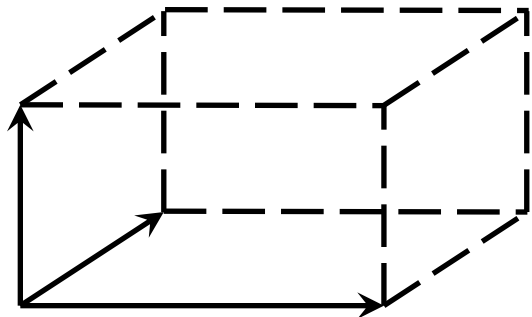
An idea for framing the software engineering process

Ekkart Kindler, Joseph Kiniry,
Anne Haxthausen, Hubert Baumeister

Talk at GTSE 2012
(Stockholm, Nov. 2012)

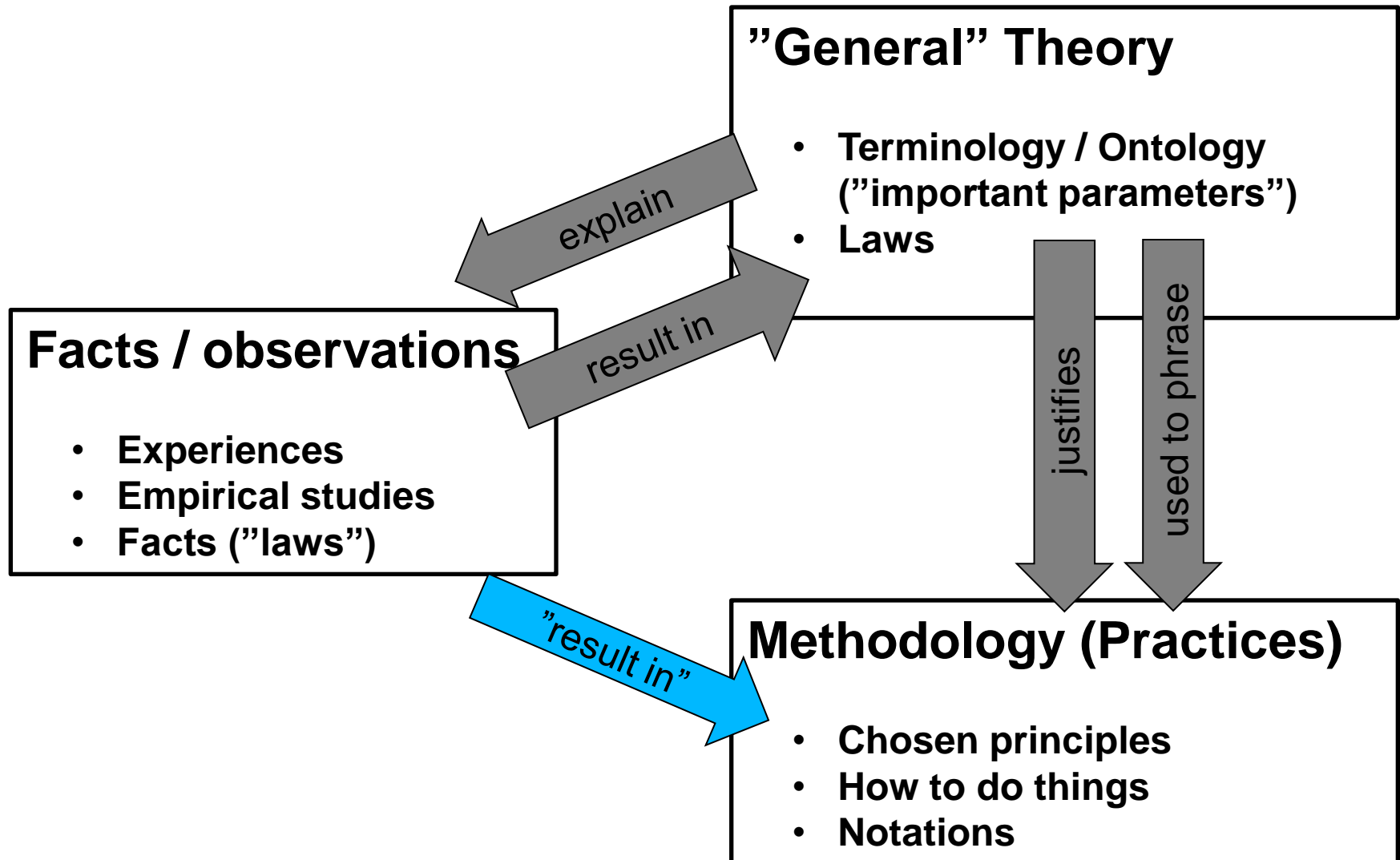
DTU Informatics

Department of Informatics and Mathematical Modeling



$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

Δ \int_a^b ε Θ $\sqrt{17}$ $+$ Ω \int δ $e^{i\pi} =$
 ∞ $\{2.7182818284$
 χ^2 Σ $!$ \gg \approx





what

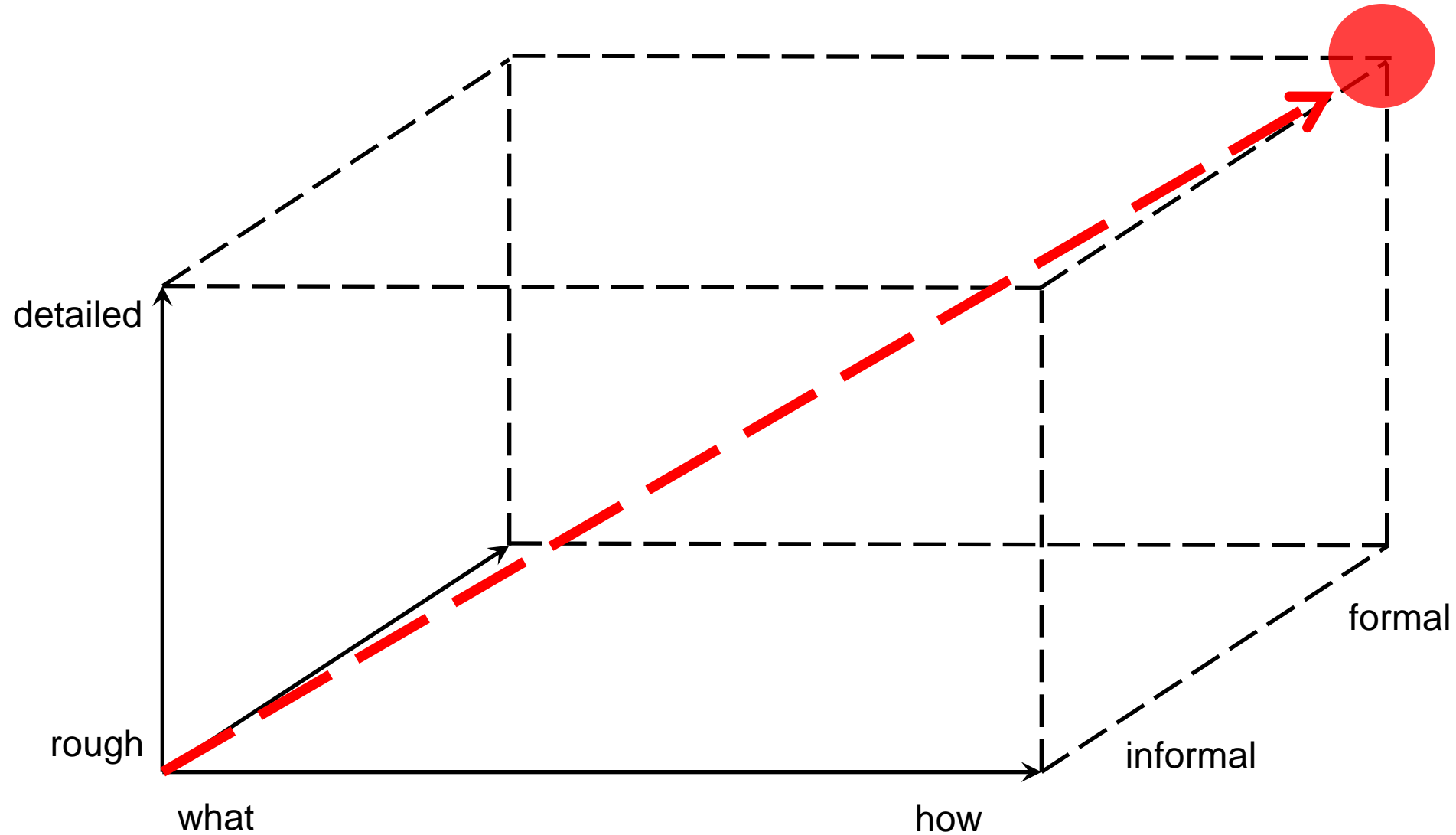
rough



how

detailed

Some of the repeating
"slogans" in my course
SE 2.



Name:

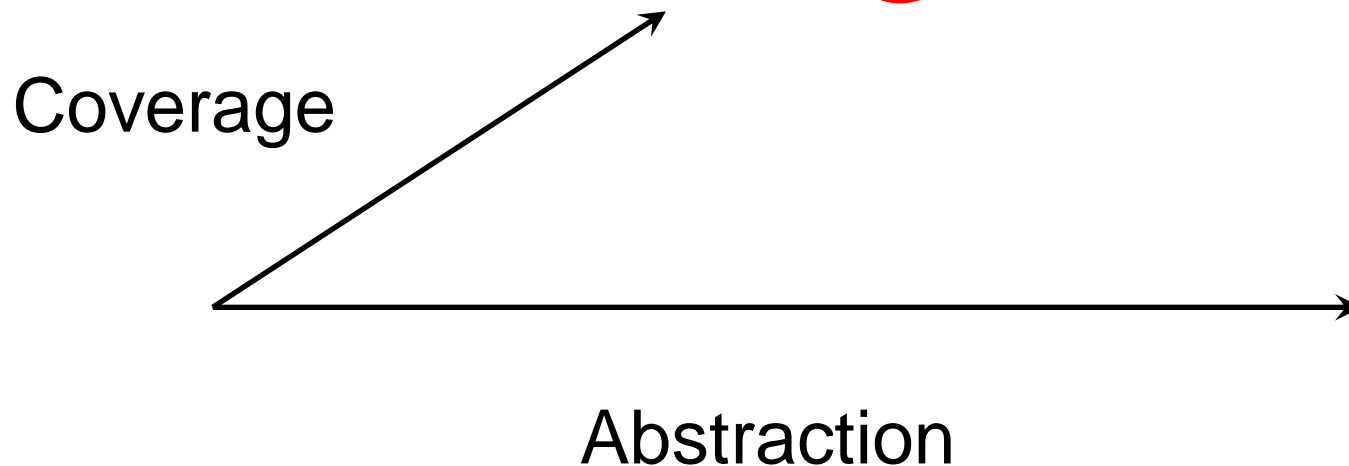
Definition:

”Litmus test”:

Examples:

*Can we fill that for the
”What/how”-dimension
and some others?*

- Level of detail, Abstraction, Composition, ...

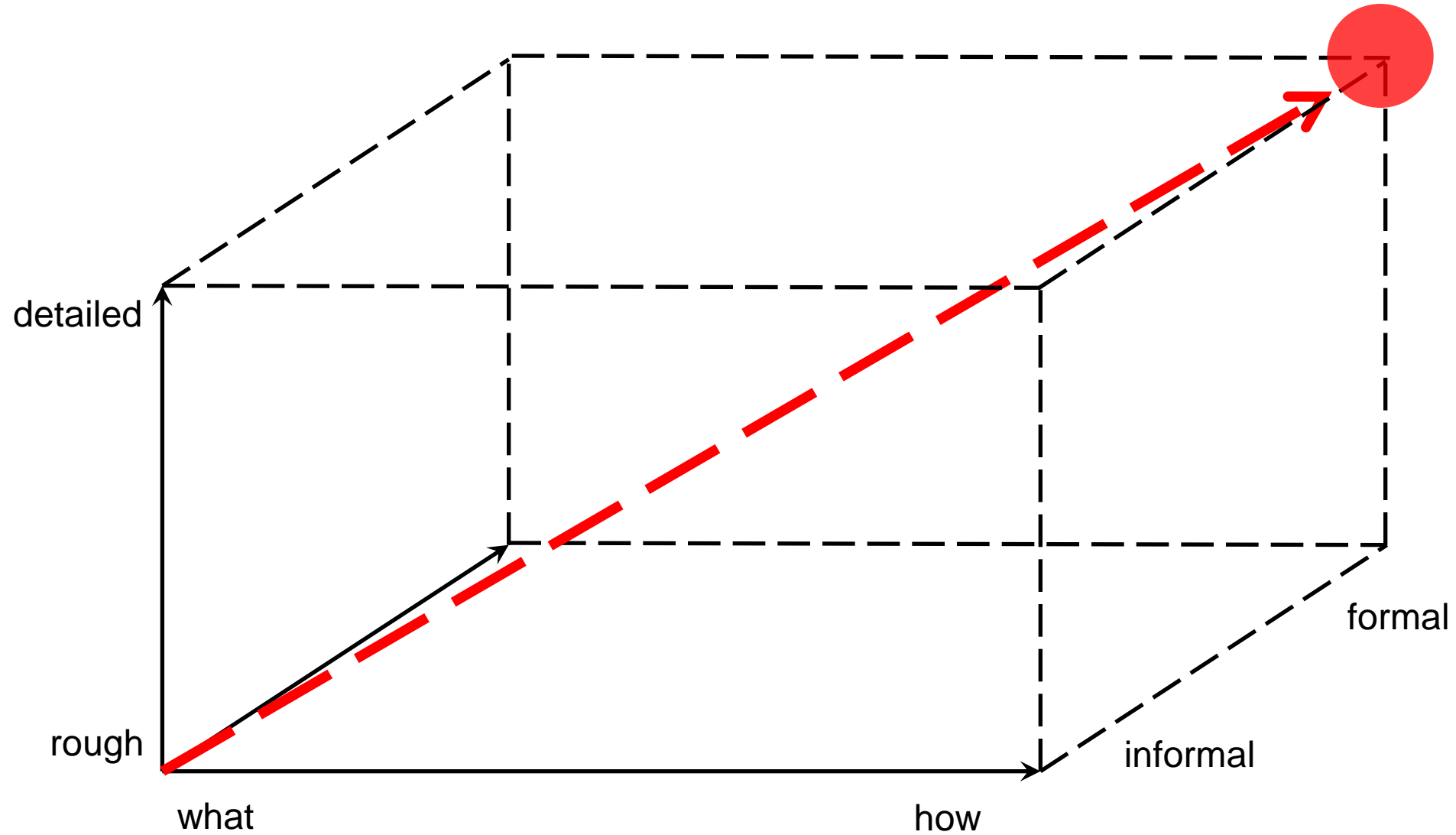


- level of detail
- declarative / executable

- informal / formal
- textual / graphical

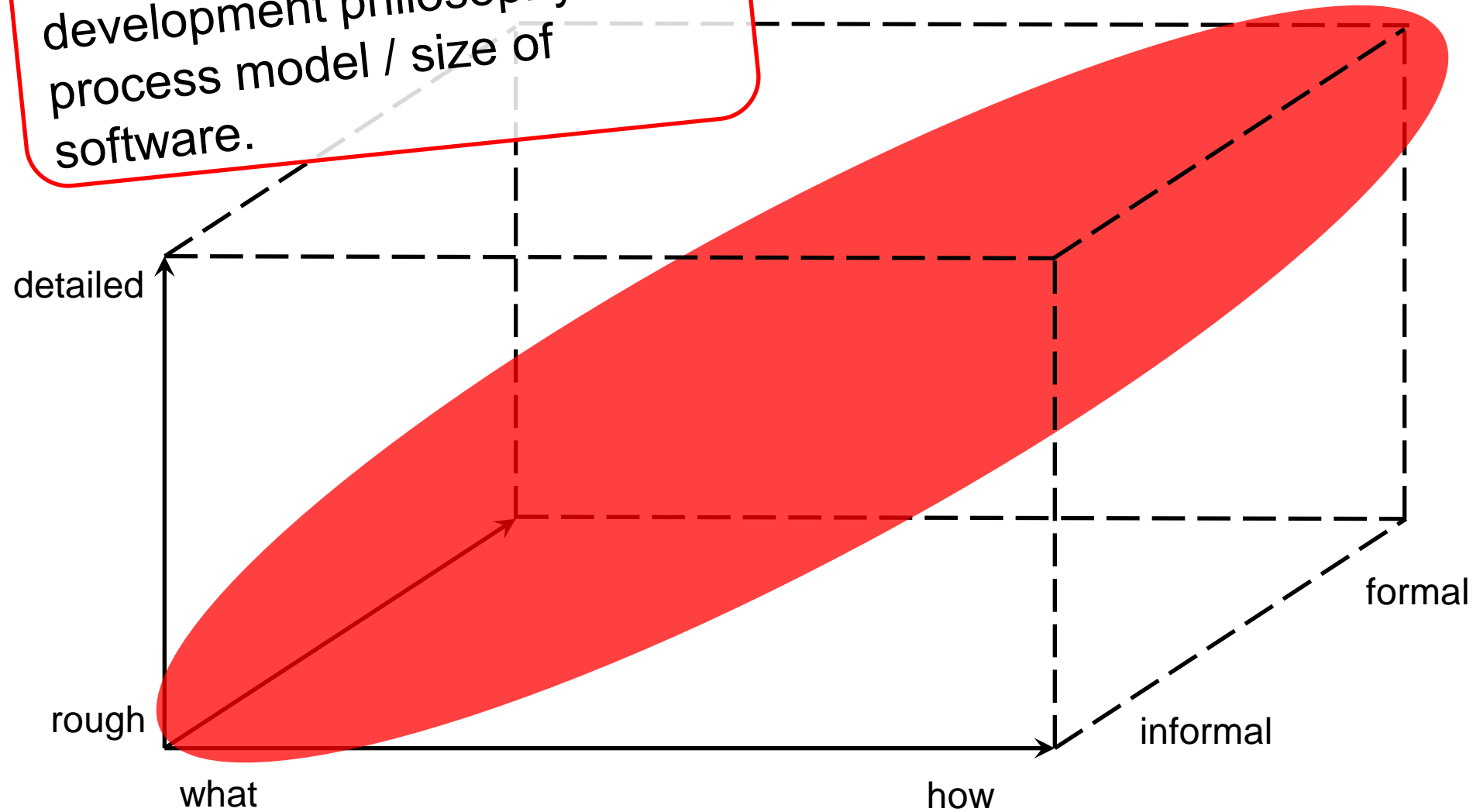
- "imprecise" (loose) / precise

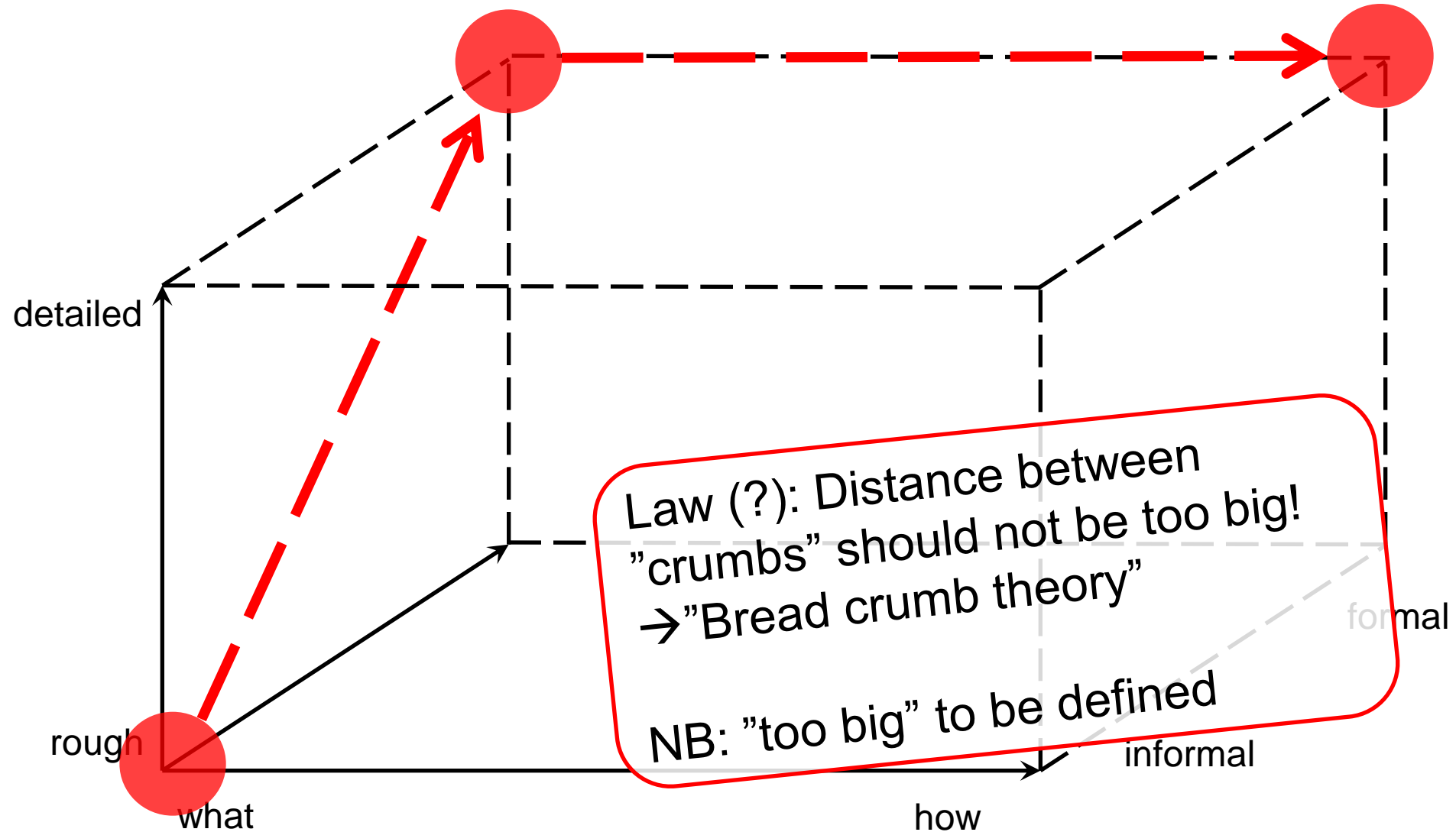
The goal?



"Waterfall model"

Different shapes dependent on development philosophy / process model / size of software.





- Product objective
- Product use
- Use cases
- User story
- Domain model
- Code (implementation)
- Prototype
- GUI definition
- GUI mockup
- ...
- Design
- Architecture
- Data base schema
- XML Schema
- OOA
- OOD
- Systems specification
- Requirements specification
- Formal model
- Handbook