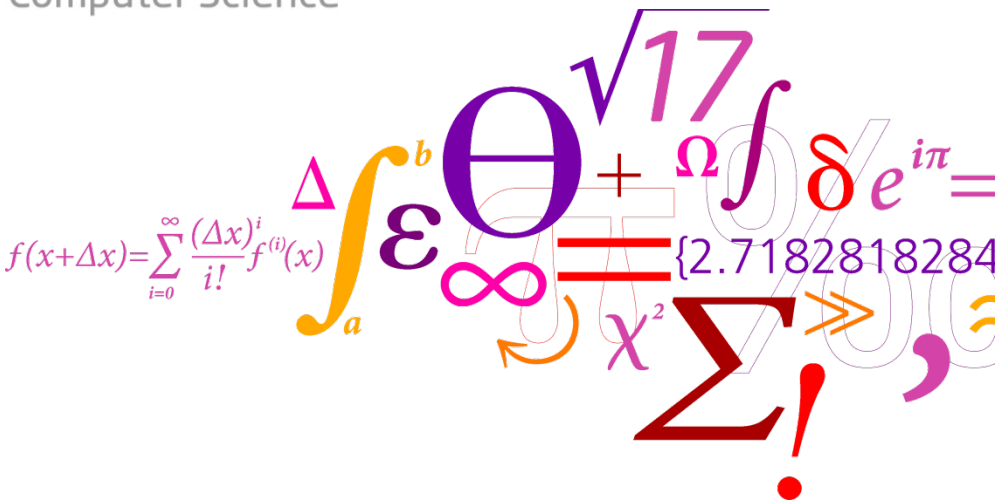
Software Engineering 2

A practical course in software engineering

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science

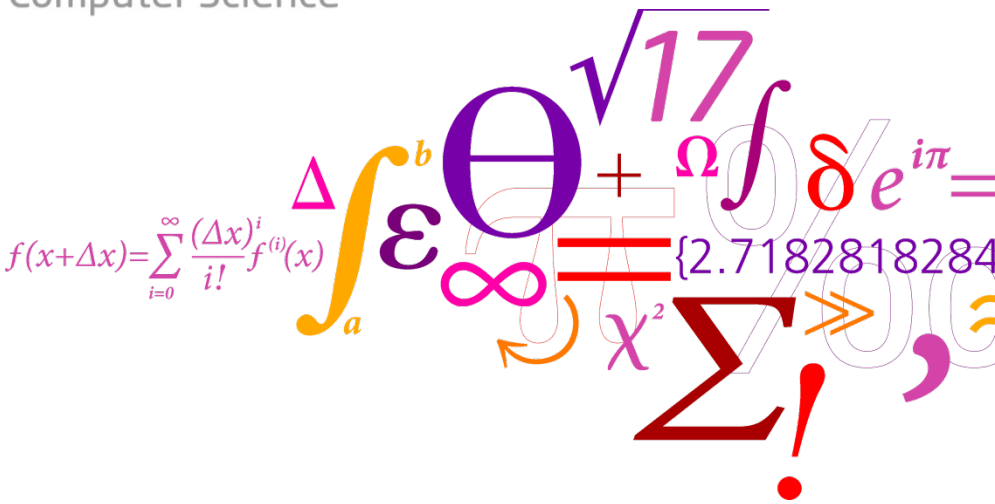


IV. Working together

IV.5. Management Principles

DTU Compute
Department of Applied Mathematics and Computer Science

Some slides repeated from lecture 4!



Definition

(Software) Management

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

Management Goals

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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!

Short- vs. Long-term Goals

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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.

Management

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Planning**
  - set goal
  - set dates
  - define course of action
  - assign resources
  - …
- **Organization**
  - define organisation structures
  - assign responsibilities
  - define tasks
  - assign tasks
  - …

Define and assign tasks:
→ see slide 9!

Management

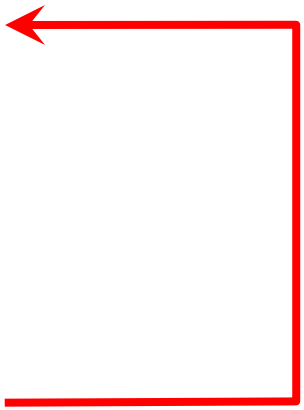
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Leading**
  - lead and motivate team members
  - improve communication
  - solve conflicts
  - …
- **Monitoring and controlling**
  - check progress
  - identify problems (early)
  - produce relief
  - …

Management cycle

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Planning
- Organization
- Leading
- Monitoring & controlling



Plans need to be followed up!

More important than following a plan:
monitoring and adjusting

Units of work: Tasks

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

In order to work effectively and efficiently, the work needs to be split up into “reasonably sized junks of work” (tasks)

In agile development, these principles are built in naturally. But, you need to be aware of them!

Driving principles

- value for owner
- manageable (timewise and cognitively)
- minimizing length of critical path
- increase parallel work / avoid idle times

For task assignment: make sure to share and spread knowledge (on domain, product and technologies); for pair programming this entails: no statically fixed pairs.

Management issues

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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.

Management:

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

- Minimise risk
  (or minimise negative impact of risk)

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

- Balance workload

Complex vs Complicated

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Complexity is inherent to the problem solved

- ”Complicatedness is difficulty that serves no purpose ...”
  http://picture-poems.com/week4/complexity.html

See lecture 2!

Make sure to

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

„WHY"

„WHAT"

C

D

„HOW"

I

O

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Conceiving
- Designing
- Implementing
- Operating

Process models

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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.

- prescriptive (e.g. waterfall, V-model, RUP, …)

Agile manifesto

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

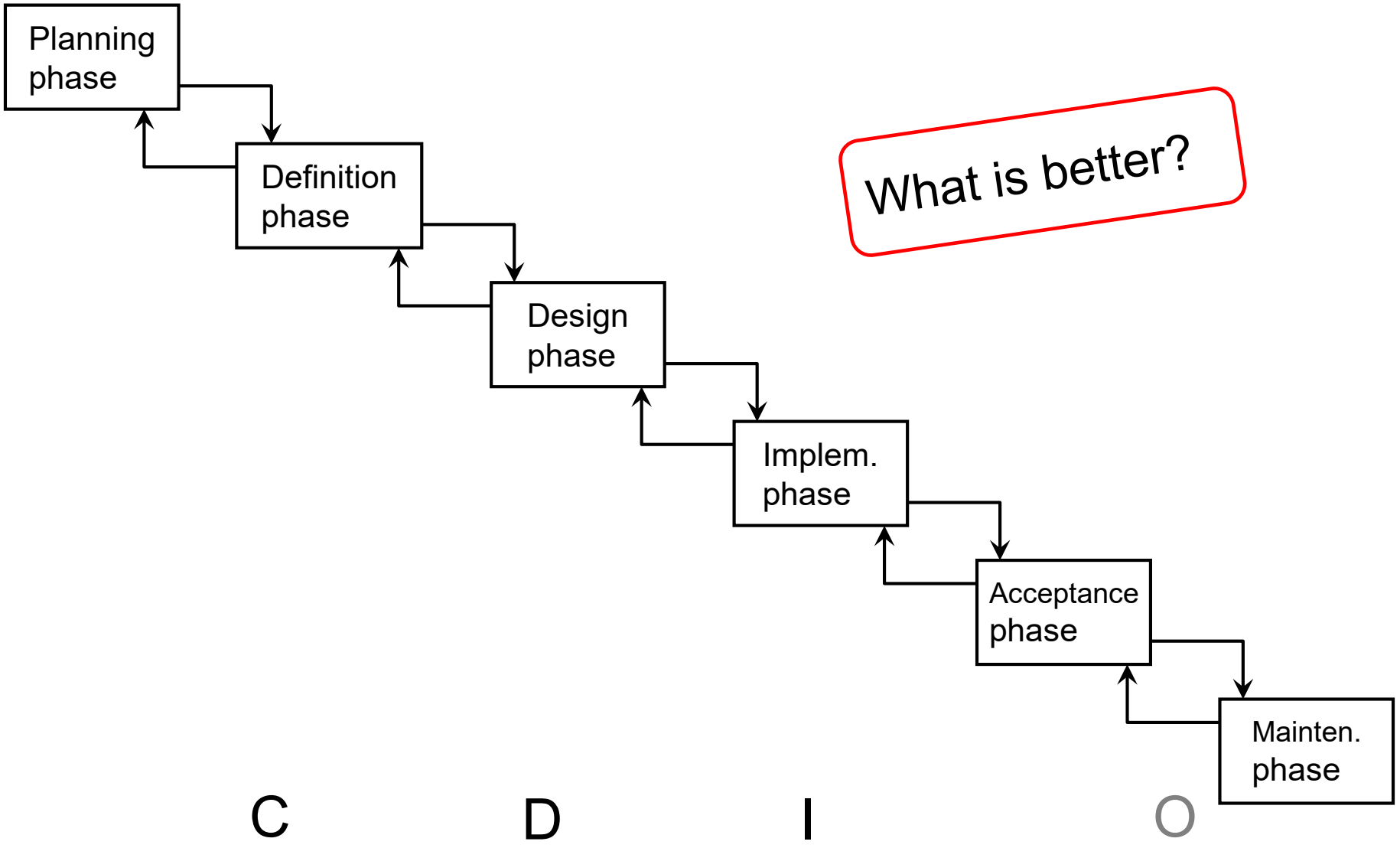
- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

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

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Mistakes

- **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 a first implementation reveals what we really wanted)
  → ”co-evolution” of understanding of problem and solution

We cannot afford to be dogmatic about methods!

In all process models

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

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

Why / what

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Why?

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

What?

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

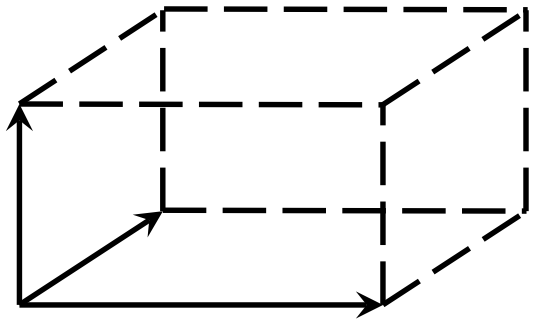
On the dimensions of software documents

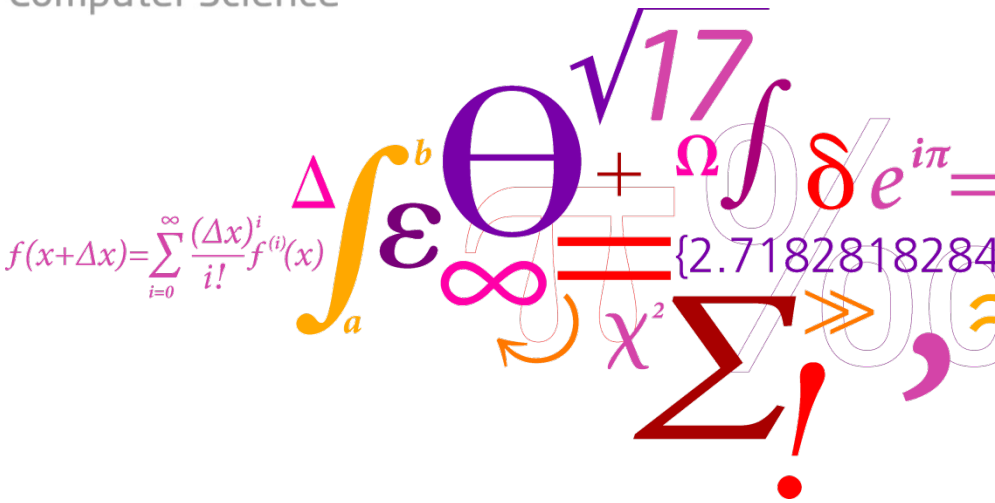
An idea for framing the software engineering process

Ekkart Kindler, Joseph Kiniry,
Anne Haxthausen, Hubert Baumeister

Talk at GTSE 2012
(Stockholm, Nov. 2012)

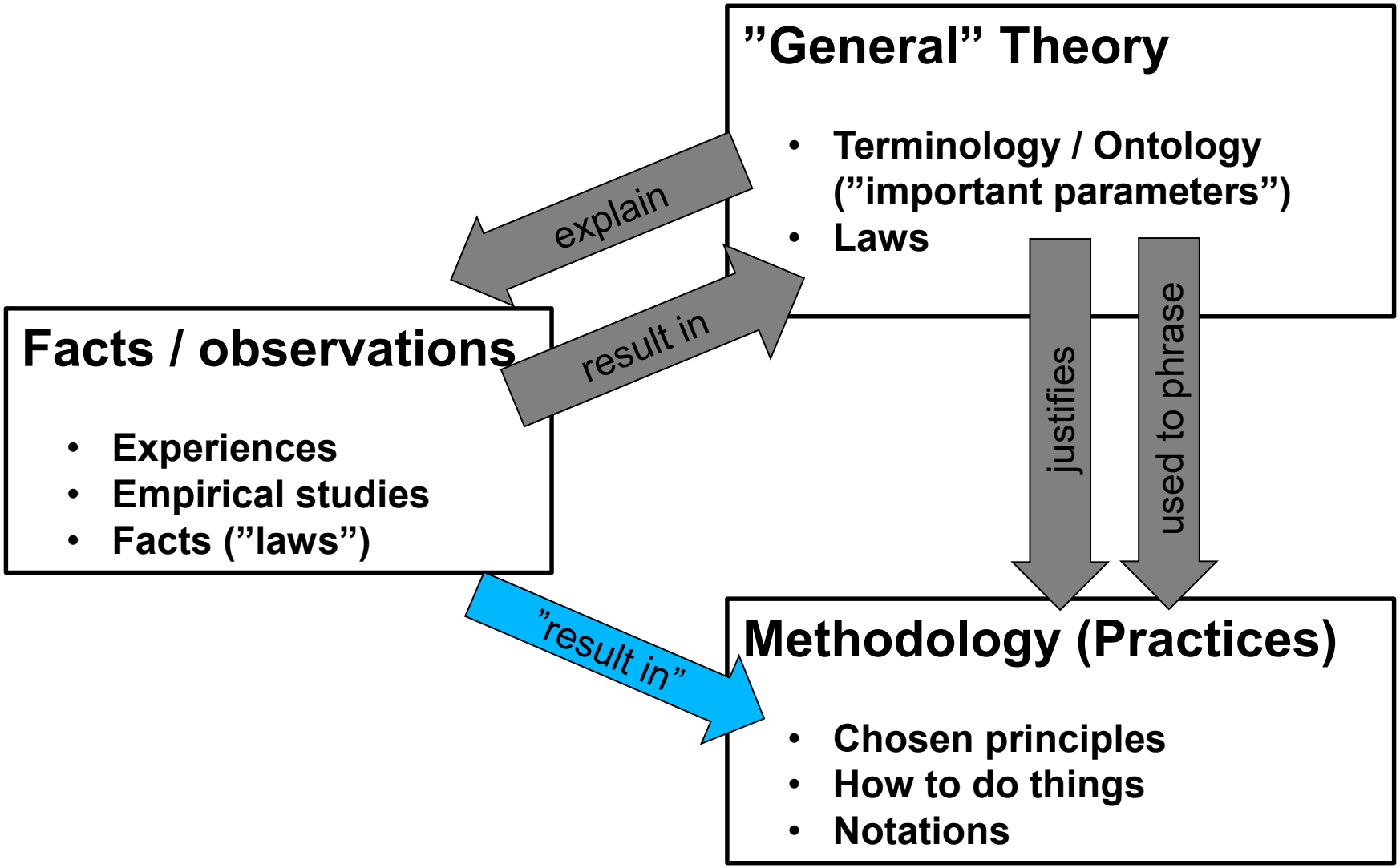
DTU Compute
Department of Applied Mathematics and Computer Science





SE ”theories”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

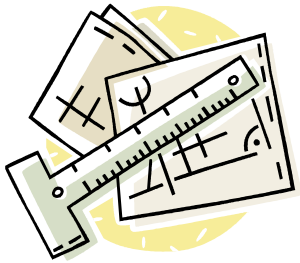


Teaching SE process

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

what

rough



how

detailed

Some of the repeating "slogans" throughout the course SE 2

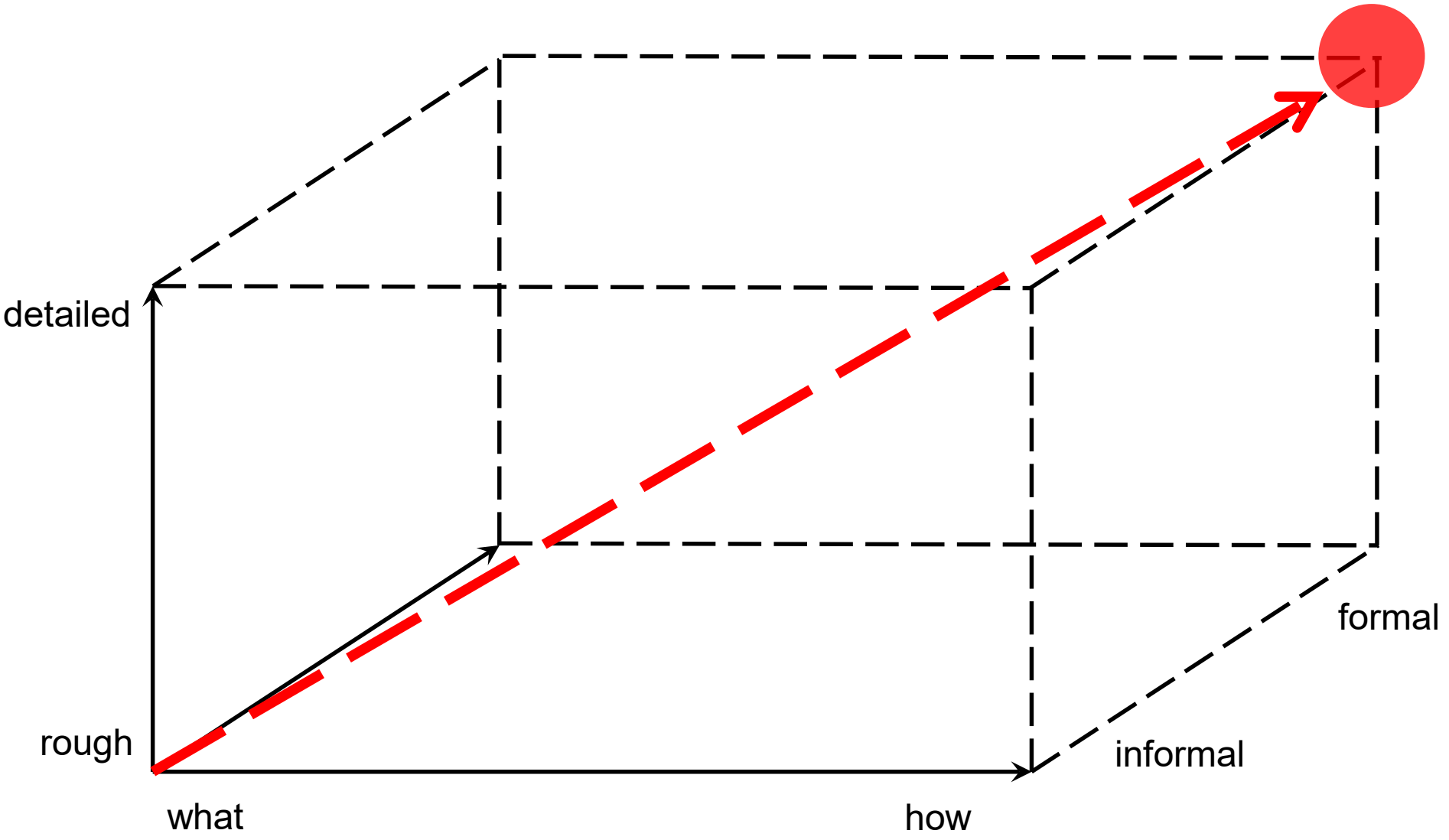
Three main dimensions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Why, what, how
- Level of detail (rough ←→ detailed)
- Level of formality (informal ←→ formal)

Describing Software

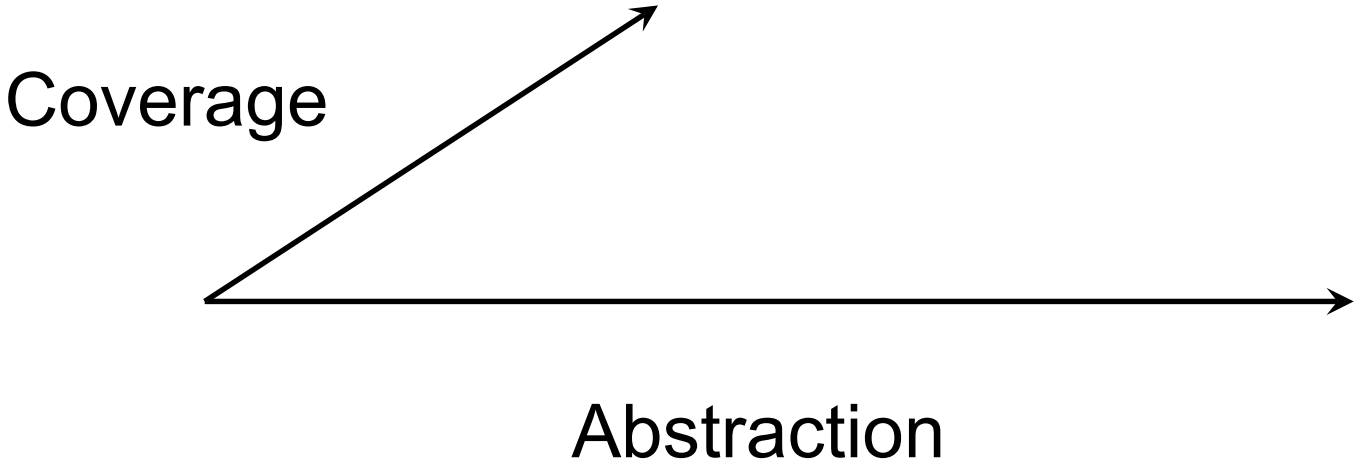
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Entanglement: ex. ”Rough/detailed”

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

Parts of these dimensions are product specific.



Entaglement

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

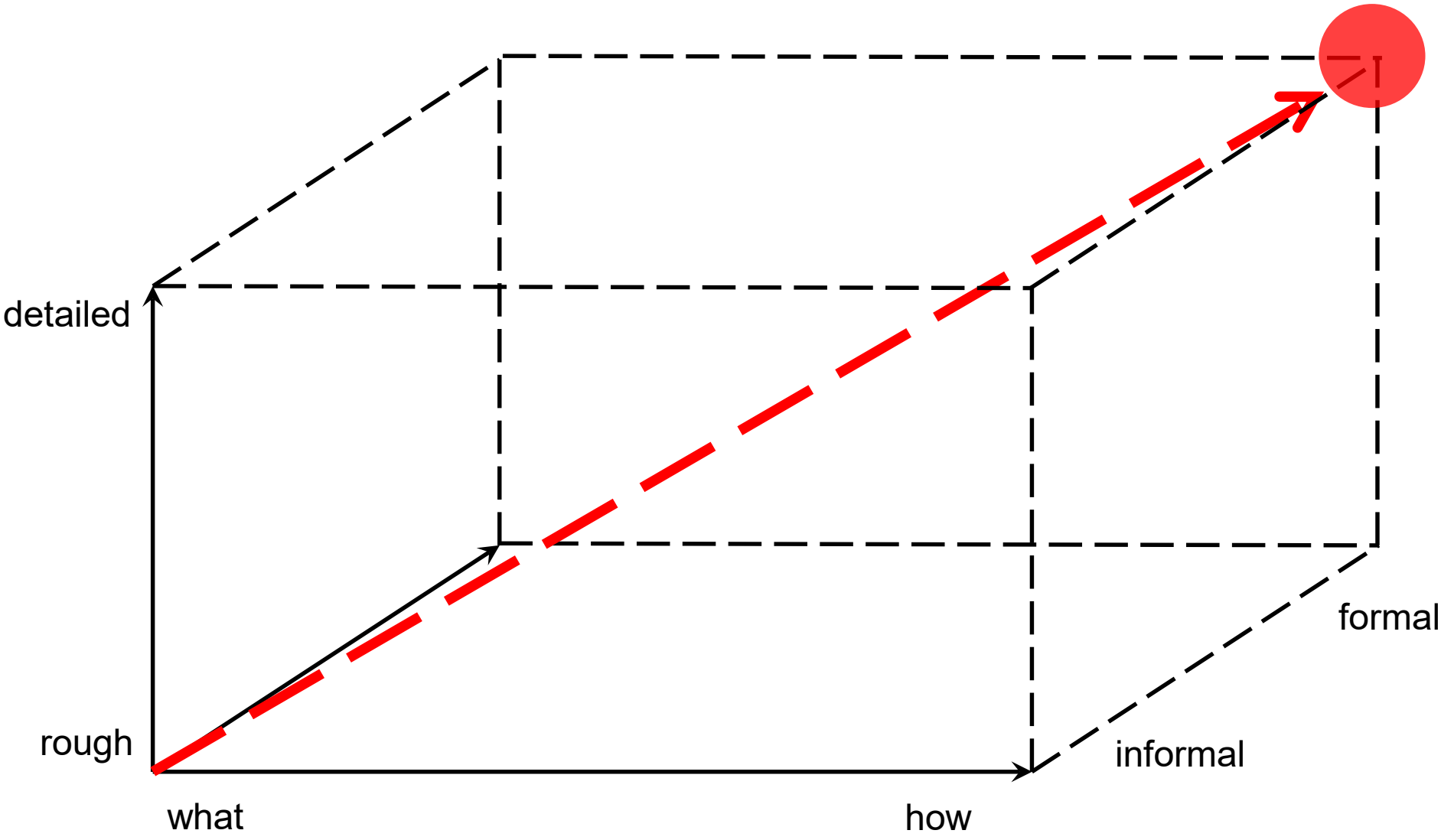
- why is typically informal

- level of detail
- ”imprecise” (loose) / precise
- declarative / executable

- informal / formal
- textual / graphical

The goal

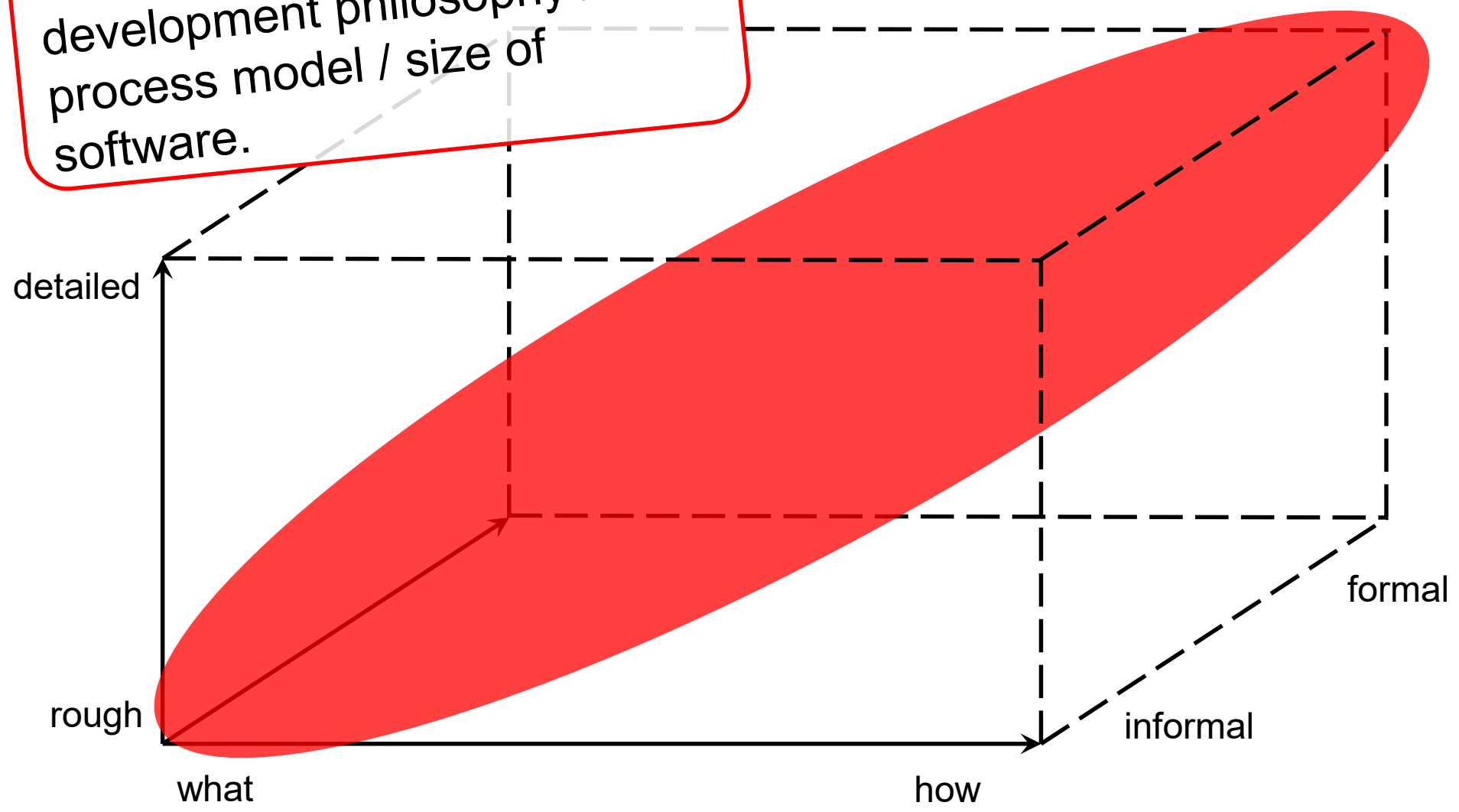
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



”Waterfall model”

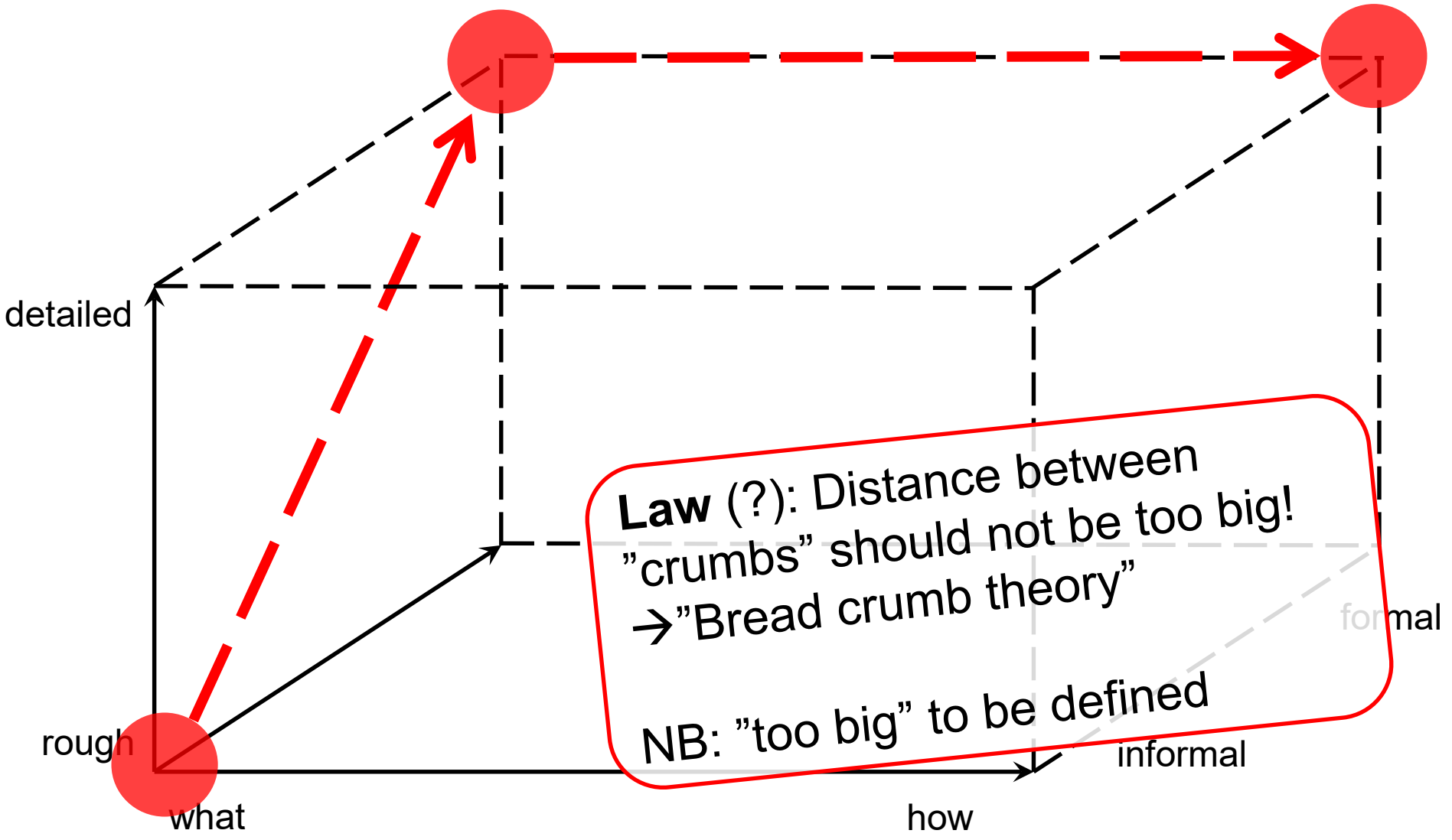
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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



”Agile”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Example documents

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Product objective
- Product use
- Use cases
- User story
- Domain model
- Code (implementation)
- Test
- Prototype
- GUI definition
- GUI mockup
- ...

- Design
- Architecture
- Data base schema
- XML Schema
- OOA
- OOD
- Systems specification
- Requirements specification
- Formal model
- Handbook

Dimensions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler