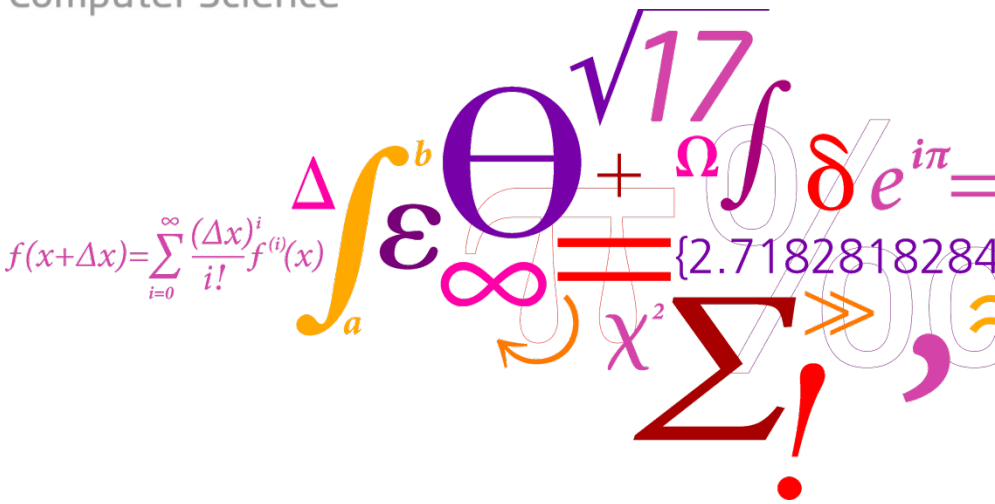
Software Engineering 2

A practical course in software engineering

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



Agenda today

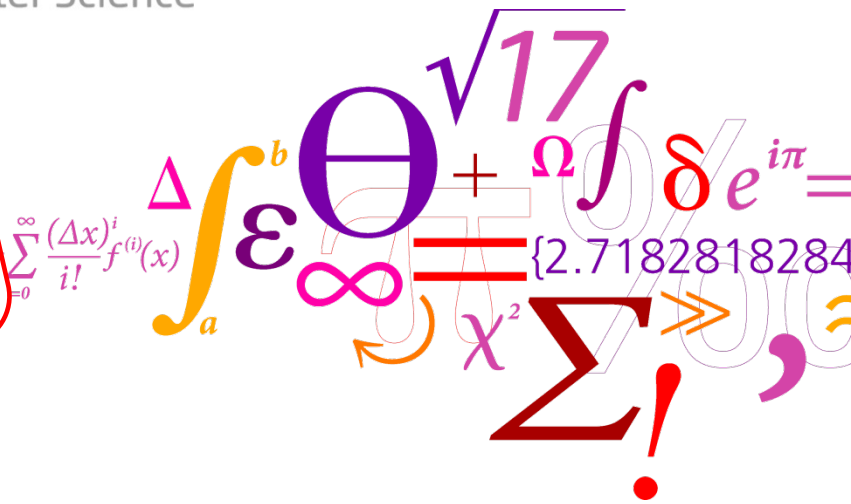
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Agile Development
- More details on project
- Discussion on project (Q&A)
- Getting existing software running
  (by Shahrzad M. Pour)

II. Agile Development

DTU Compute
Department of Applied Mathematics and Computer Science

Following (to some extent):
Kent Beck: Extreme Programming Explained, Addison Wesley, 1999.
(available online via FindIt).



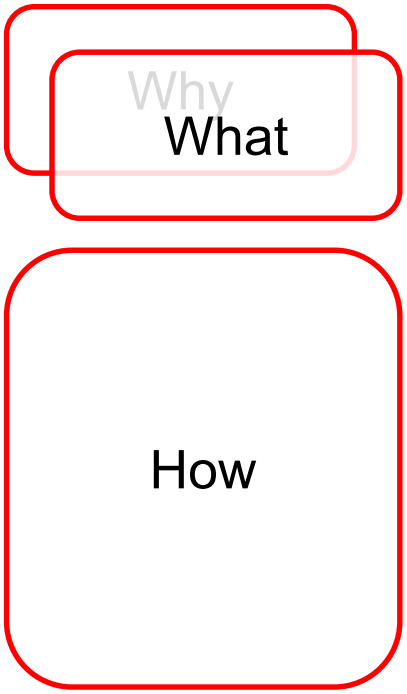
- In Lecture 1 already:
  A quick overview of some agile *practices*.

- Today:
  - Conceptual background
  - More details

1. Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

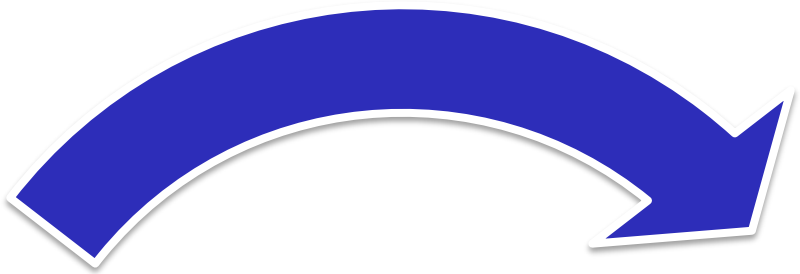
Conceive
Design
Implement
Operate



Co-evolution

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

"What" should the software do?

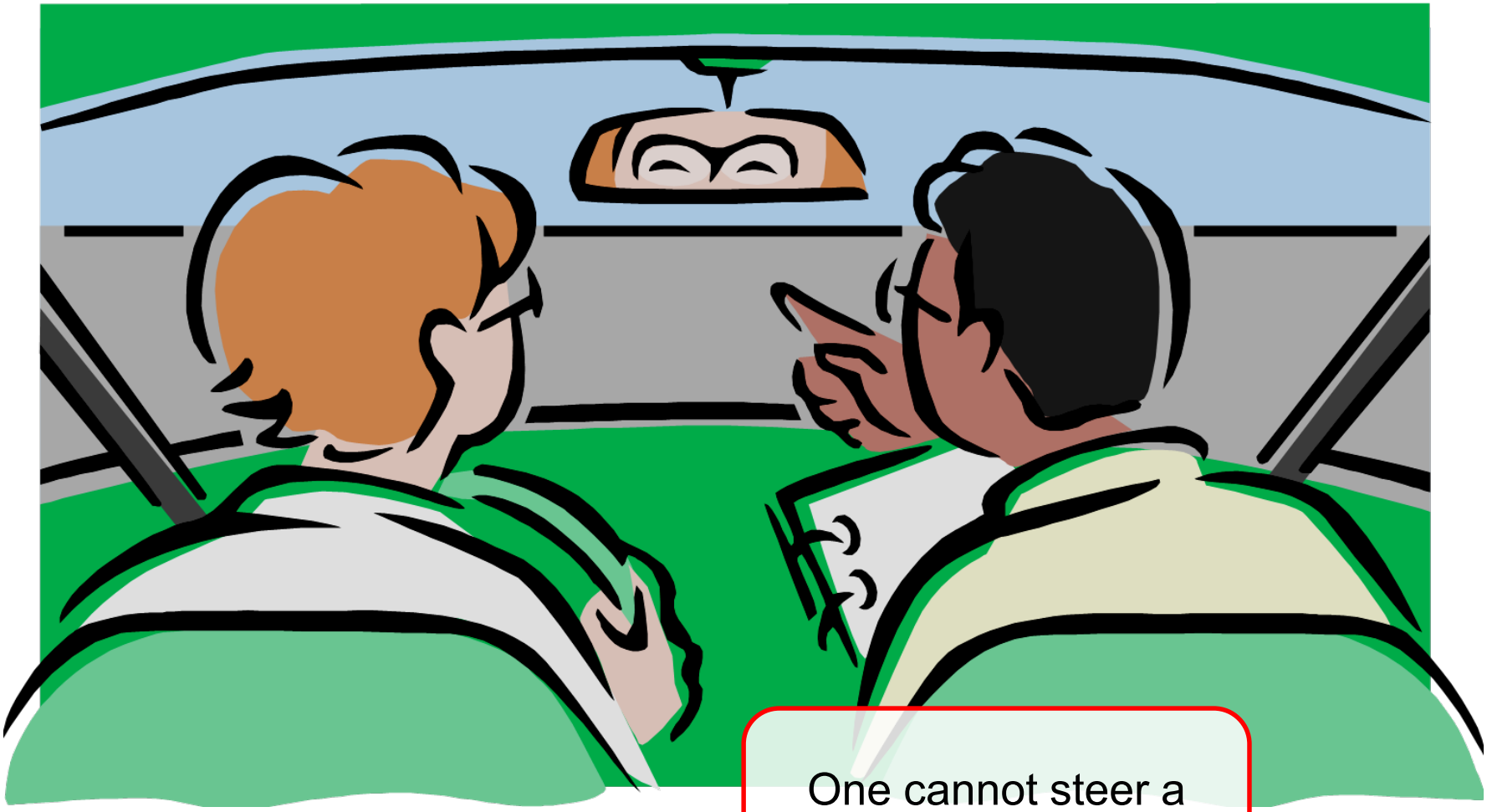


WHAT HOW

"How" is it realized?

Driving a car

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Gall's law

"A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a working simple system."

John Gall: Systemantics: An essay on how systems work, and especially how they fail. General Systemantics Press, Ann Arbor, Michigan, 1975.

Main message

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Experience shows (for complex systems):

- **CDIO** does not work purely sequentially
- Once we have implemented a system (**how**), we get a (much) better understanding of **what** the system should do!
- Software development and management is very much about managing risk
- Development process needs adjustments while we are going

→ **Agile Software Development**

Agile manifesto

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

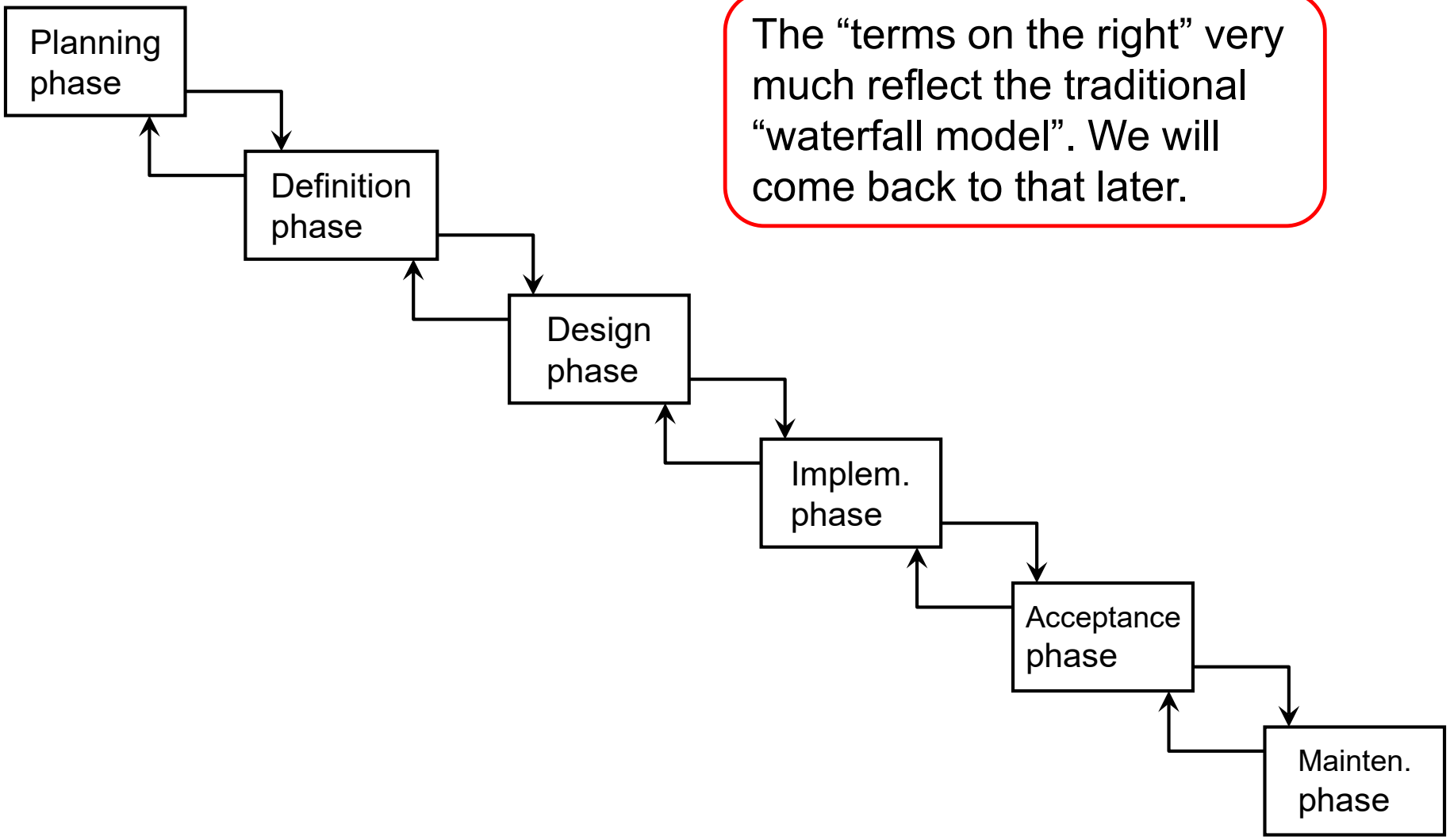
- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

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



The “terms on the right” very much reflect the traditional “waterfall model”. We will come back to that later.

2. Background

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Four variables of software development
- Cost of change

Agile development

- Values
- Principles
- Basic activities

The following presentation follows "The Book": Kent Beck: Extreme Programming Explained, Addison-Wesley 1999 (online via DTU see course's web page)

Variables

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Four variables of software development process

- Cost
- Time
- Quality
- Scope

Basically, three variables can be freely chosen (e.g. by customer). The fourth is a result of the pick of the three (i.e. decided by develoment team).

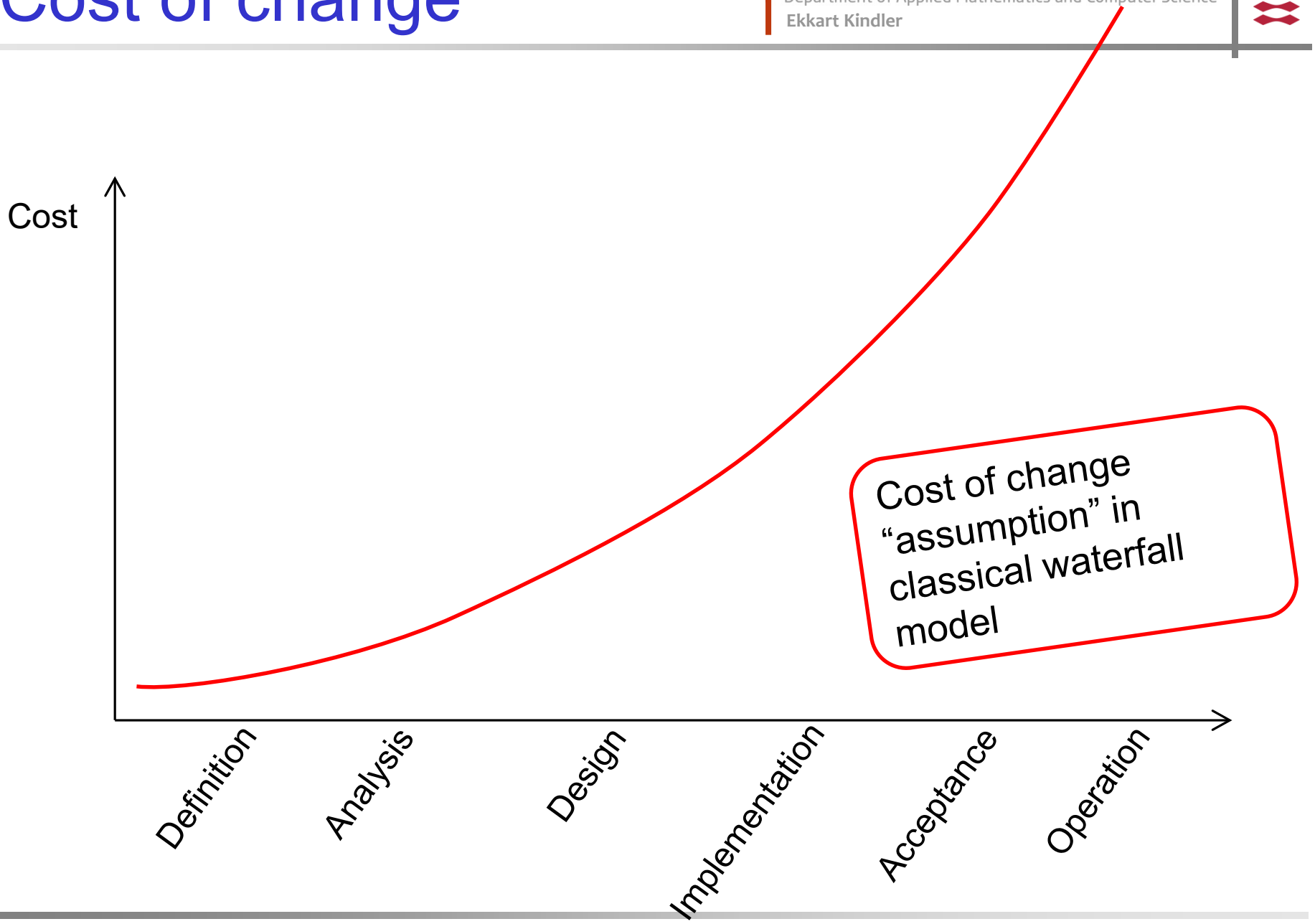
Note

- There is no way the customer could pick all four variables
- Even three variable are not fully independent (e.g. more developers do not necessarily speed up the process, definitely and not in a linear way)

- Agile helps adjusting these variables dynamically (and get the most value for the customer):
  - Short iterations
  - More practise in estimating user stories (developers)
  - Prioritization of user user stories (customer)

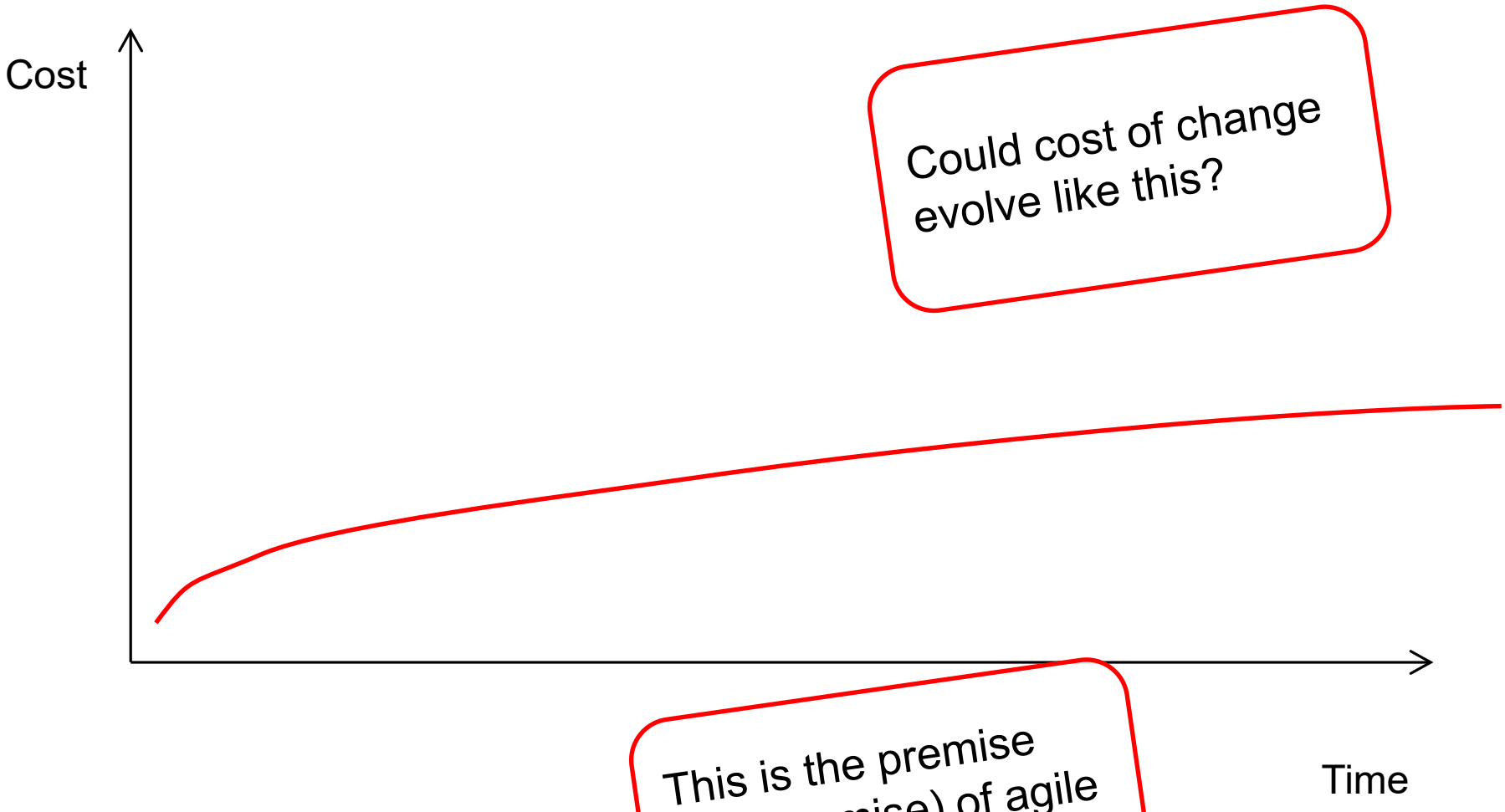
Cost of change

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Cost of change

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Agile Values

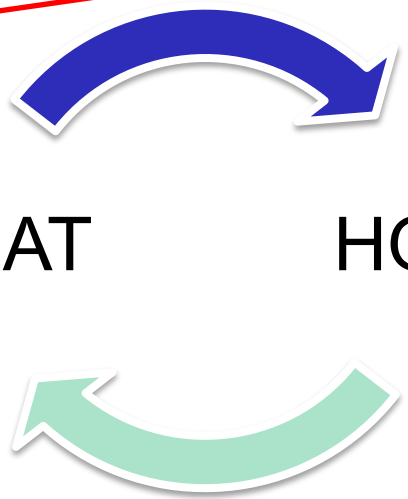
- Communication

Talking & interacting

- Simplicity

Better a simple thing that serves the purpose today, than a complicated thing that might never serve a purpose

- Feedback



But a lot more that that

- Courage

Embrace challenges & changes

BTW: Complex vs Complicated

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Complexity

- Complicatedness

Is there a difference!

BTW: Complex vs Complicated

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Complexity is inherent to the problem solved

→ over-engineering

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

In agile: Complicatedness is difficulty that serves no purpose **today** (i.e. the current release)!

Agile Principles

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Values give some orientation and criteria for a successful agile development
- But, values are too vague for defining concrete practices

→ Therefore, agile builds on principles

Fundamental Principels

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Rapid feedback
- Assume Simplicity
- Incremental change
- Embracing change
- Quality work

Assume that the problem can be solved in a simple way.

Other Principles

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Teach learning
- Small initial investment
- Play to win
- Concrete experiments
- Open, honest communication
- Work with peoples' instincts, not against them
- Accepted responsibilities
- Local adaptation
- Travel light
- Honest measurements

As opposed to "play not to loose"

Activities

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Coding
- Testing
- Listening
- Designing

Agile development and practices help structuring these activities, so that you can keep doing them indefinitely.

3. Agile Practices (overview)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **On-site customer** (Ekkart and, sometimes, Shahrzad)
- **Small/short releases** 2-3 week
- **Planning game**

Agile Practices

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Coding standards**
- **Testing**
- **Continuous integration**

Don't forget to define your coding standards (most IDEs have predefined some; and most programming languages have their own standards)

Test driven development: start with programming the test; automated test (unit tests)

Agile Practices

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Pair programming**
- **Simple design**
- **Refactoring**

Missing last time

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Metaphors:
  Simple story (and terminology) how the system should work; speak simple pictures

- Collective ownership:
  Anyone can change, anyone has obligation to change (if value increases)

- 40-hour week (currently in Denmark 37.5 hour week)

Synergy of Practices

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- The different practices support each other
- One practice's weakness is the others strength
- You cannot pick/chose practices arbitrarily

If you pick arbitrarily, the result might be hacking instead of agile development!

Example

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Short Releases would be impractical, unless

- **Planning Game** helps identifying the user stories with most value
- **Continuous Integration** allows deploying it with minimal effort
- **Testing** would guarantee low (or desired) defect rate
- **Simple Design** allows you what is needed now

4. Details

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Planning Game
- Pair Programming
- Testing

- Organizing facilities

Planning Game

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Project owner:
  Formulates user stories
- Developers:
  Estimate stories (how long do they take to implement)
- Developer: If a story is too big, story is split up in sub-stories and tasks
- Project owner prioritizes and chooses stories for next release (choose scope), but not more than developers estimated they can do!
- During iteration: Developers, assign and implement stories and adjust (if needed together with project owner)

→ Last week

User Stories: Examples

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Example from last years' projects

- As a home owner, I want to be able to use NorthQ devices with openHAB, so that I can use NorthQ devices together with other devices available in openHAB
- As an inhabitant of some home, I want the openHAB system to be aware of my location (even outside the home) so that the system can take actions based on this information (which, for example, could be turning on the heating in my room when I am on my way home from work).
- As a home owner, I want openHAB to collect data from sensors and record events, which later can be visualized and analyzed for "some" purpose so that I can optimize my home (e.g. w.r.t energy consumption, well-being, ...).
- As a home owner, I want that the openHAB system and the involved NorthQ system cannot be "hacked" (in particular not through exploiting the extensions), so that I can trust the system.
- ...

User Story: Structure

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

As a …,

- **Who** is involved in the user story: user in a role

I want to do …

- **What** does the user do in the story

so that …

- **Why** is that user story relevant

Pair programming

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

is **NOT**

- one person programming, another only watching
- one person typing
- a tutoring session for the other person
- always pairing up with the same person
- programming with the buddy you best get along with

Pair programming

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

is

- communication and interaction
- getting a second opinion
- critically reviewing the other's work
  - correctness
  - simplicity
  - avoid tunnel view
- learning from each other
- dynamic

Use one machine (one keyboard and one mouse) only

BTW: Reviewing is one of the most powerful techniques to identify problems and issues

If you are not an expert, ask one to join you for the task at hand

Testing

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Two sources of tests:

- Programmers:
  Unit tests for everything which potentially or likely could be wrong (or which went wrong at a some time)
- Customers (maybe implemented by programmer):
  Functional tests for user stories

Tip: For testing complex user stories with hardware, it might be worthwhile to realize an emulator for hardware devices!

Organizing Facilities

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Group work and group / meeting:
  - Arrangement of tables
  - Use projectors and blackboards (focus discussion)
  - Make status of current user stories and tasks visible
  - Show metrics (tests, completion status)

  - Be clear about: Are you having a group discussion or are you working individually (in pairs)
  - Make sure everybody knows what their tasks are until the next meeting