Einführung in die Informatik

1. Computational Thinking

Institut für Informatik und angewandte Mathematik

UNIVERSITÄT BERN

Roadmap

- What is Computer Science?
- Computational Thinking
- Modeling
- Software Engineering

Roadmap

- **What is Computer Science?**
- Computational Thinking
- Modeling
- Software Engineering

Schedule

Datum	Thema	Dozent
9-17-2009	Computational Thinking	Nierstrasz
9-24-2009	Programmiersprachen, -paradigmen und -technologie	Nierstrasz
10-1-2009	Endliche Automaten, reguläre Ausdrücke	Studer
10-8-2009	Syntax und Semantik	Jäger
10-15-2009	Diskretisierung/Signalverarbeitung	Zwicker
10-22-2009	Simulation	Zwicker
10-29-2009	Computernetze	Braun
11-5-2009	Programme und Beweise	Jäger
11-12-2009	Berechenbarkeit	Strahm
11-19-2009	Komplexität	Strahm
11-26-2009	Betriebssysteme	Braun
12-3-2009	TBA	TBA
12-10-2009	Datenbanken und Privacy	Studer
12-17-2009	Prüfung	

© Oscar Nierstrasz

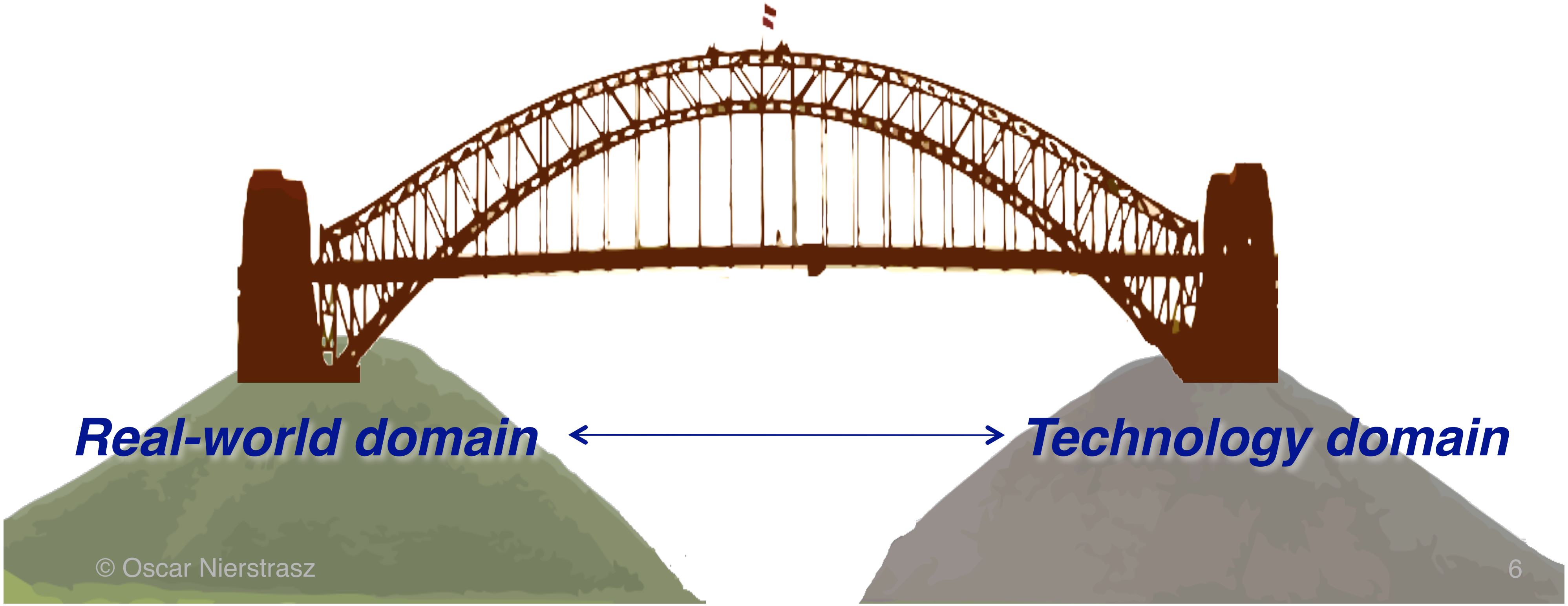
What is Computer Science … not

Computer Science is not

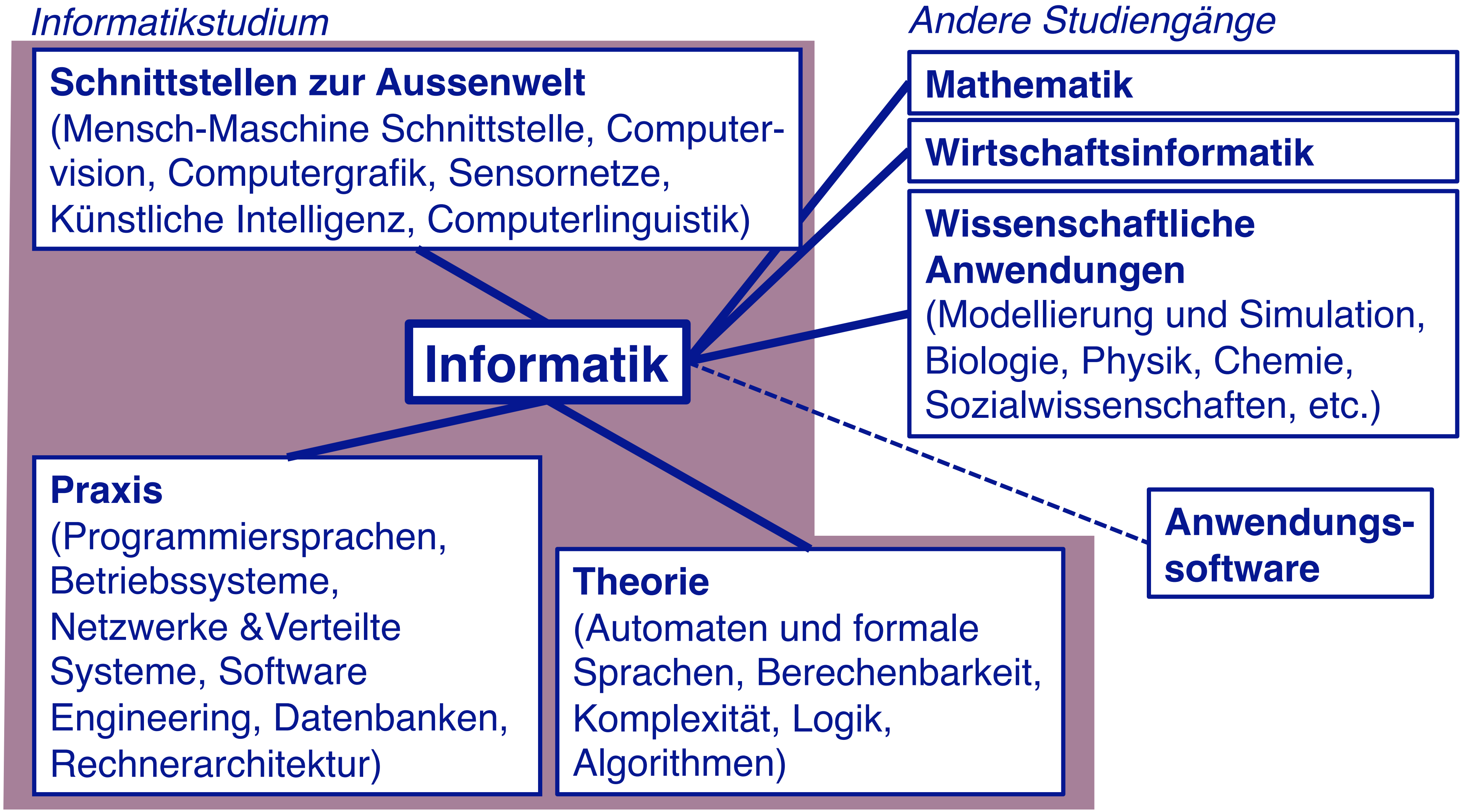
- hacking
- programming
- technology

What is Computer Science?

Computer Science is about *building bridges* between user problems and technological solutions



Übersicht


Informatikstudium
Andere Studiengänge
Schnittstellen zur Aussenwelt
(Mensch-Maschine Schnittstelle, Computervision, Computergrafik, Sensornetze, Künstliche Intelligenz, Computerlinguistik)
Mathematik
Wirtschaftsinformatik
Wissenschaftliche Anwendungen
(Modellierung und Simulation, Biologie, Physik, Chemie, Sozialwissenschaften, etc.)
Informatik
Anwendungs-software
Praxis
(Programmiersprachen, Betriebssysteme, Netzwerke &Verteilte Systeme, Software Engineering, Datenbanken, Rechnerarchitektur)
Theorie
(Automaten und formale Sprachen, Berechenbarkeit, Komplexität, Logik, Algorithmen)

Roadmap

- What is Computer Science?
- **Computational Thinking**
- Modeling
- Software Engineering

Computational Thinking

> *"Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science"*
>
> — *Jeannette M. Wing, CACM 2006*

Abstraction

Abstractions strip away details to help us cope with complexity

Decomposition & Separation of concerns

We decompose complex tasks by separating concerns

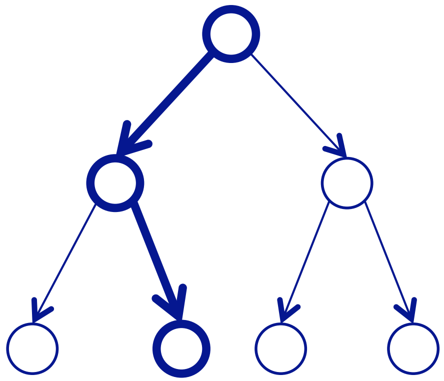
© Oscar Nierstrasz

Parallel algorithms

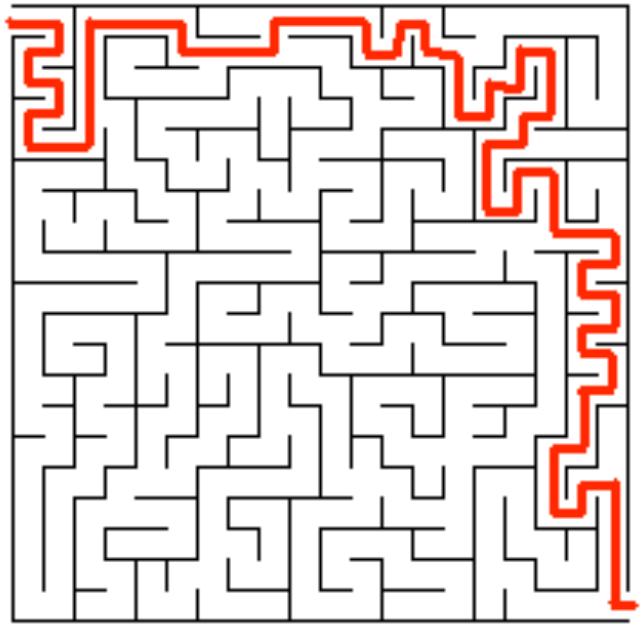
By distributing tasks to many workers, we get more done in the same amount of time

Recursion

Many navigation problems can be effectively solved using recursion.



Backtracking



Many search problems can be solved with the help of backtracking

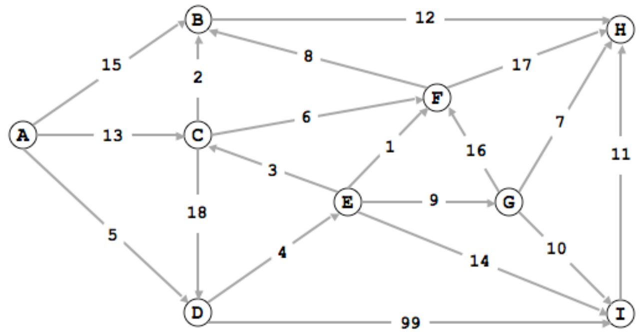
Prefetching and cacheing

Prefetching = get what you need before you need it

Cacheing = keep handy whatever you will need again

Planning and optimization

Planning may require sophisticated analysis of multiple scenarios

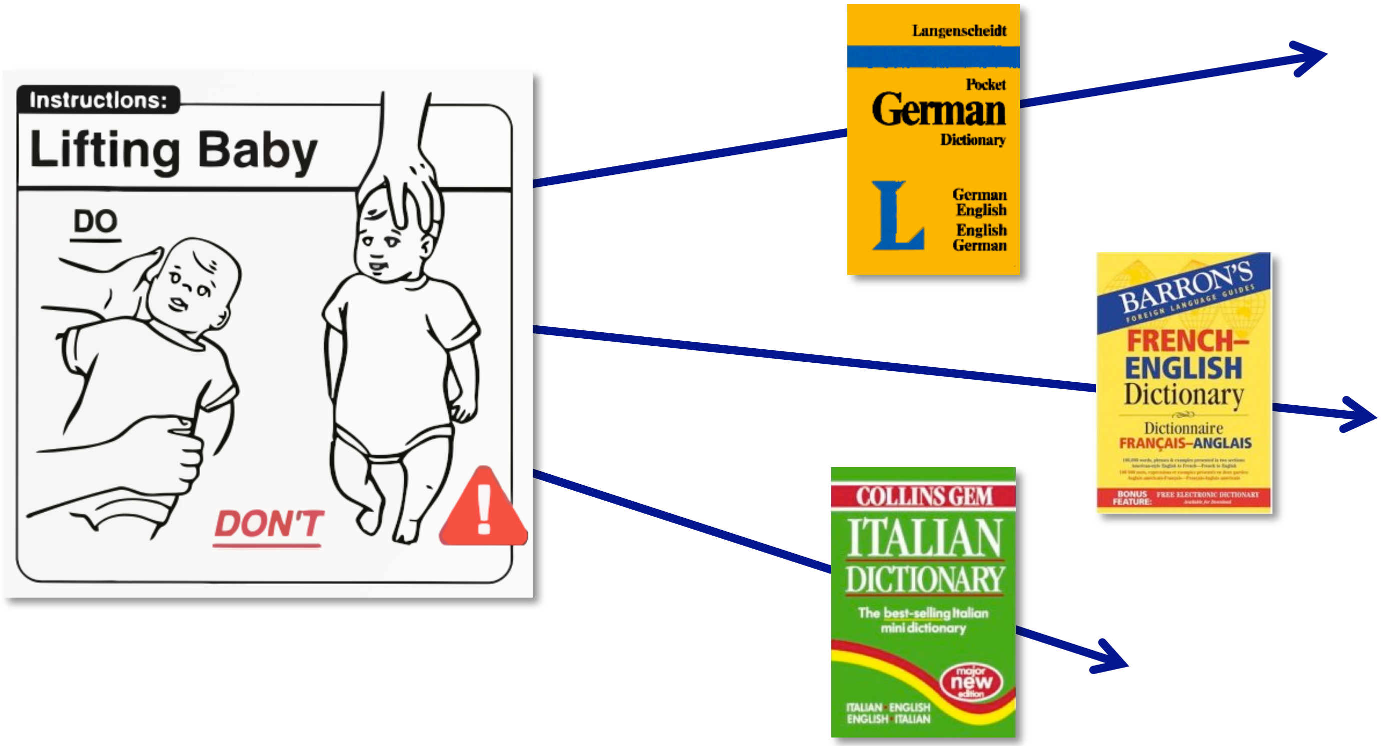


Pipelining

Pipelining makes efficient use of expensive resources

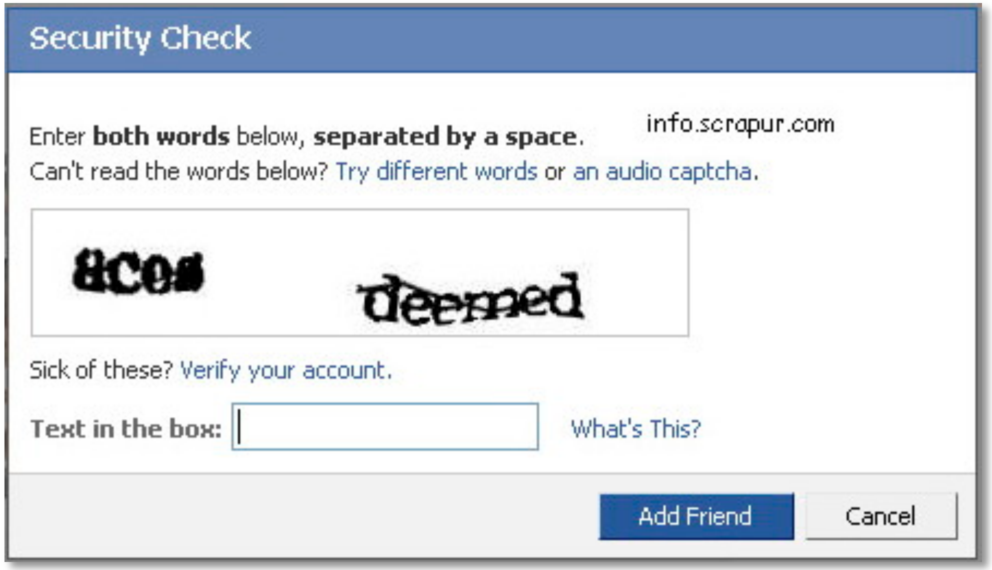
Interpreting data as code and code as data

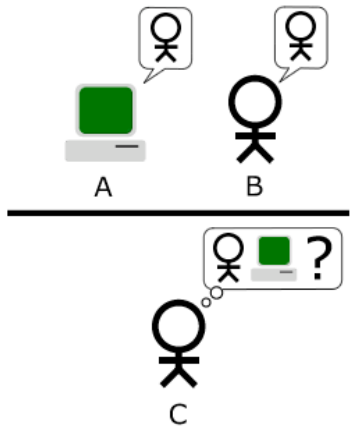
We process instructions before acting on them



Distinguishing humans from computers

Certain tasks are known to be difficult for computers to solve



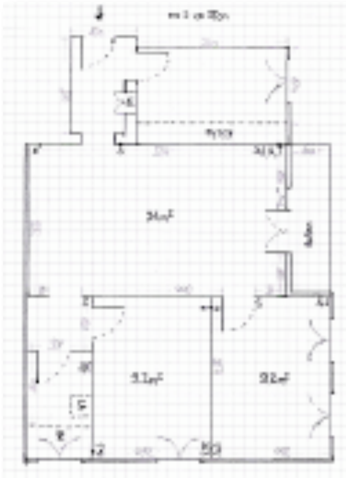


Concurrency control

Contention for shared resources can lead to both *safety* problems and *liveness* problems.

Simulation

Expensive experiments can be replaced by inexpensive simulations

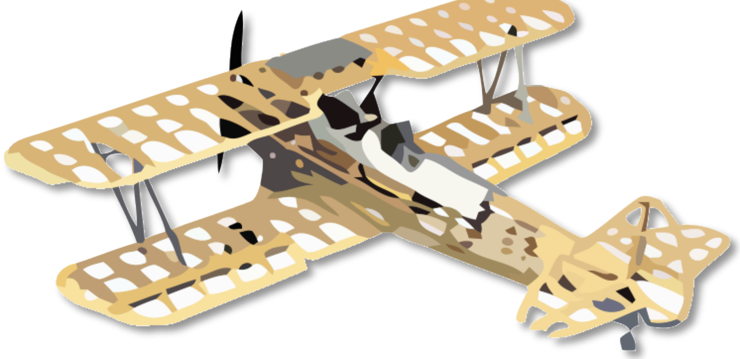


Roadmap

> What is Computer Science?
> Computational Thinking
> **Modeling**
> Software Engineering

Modeling

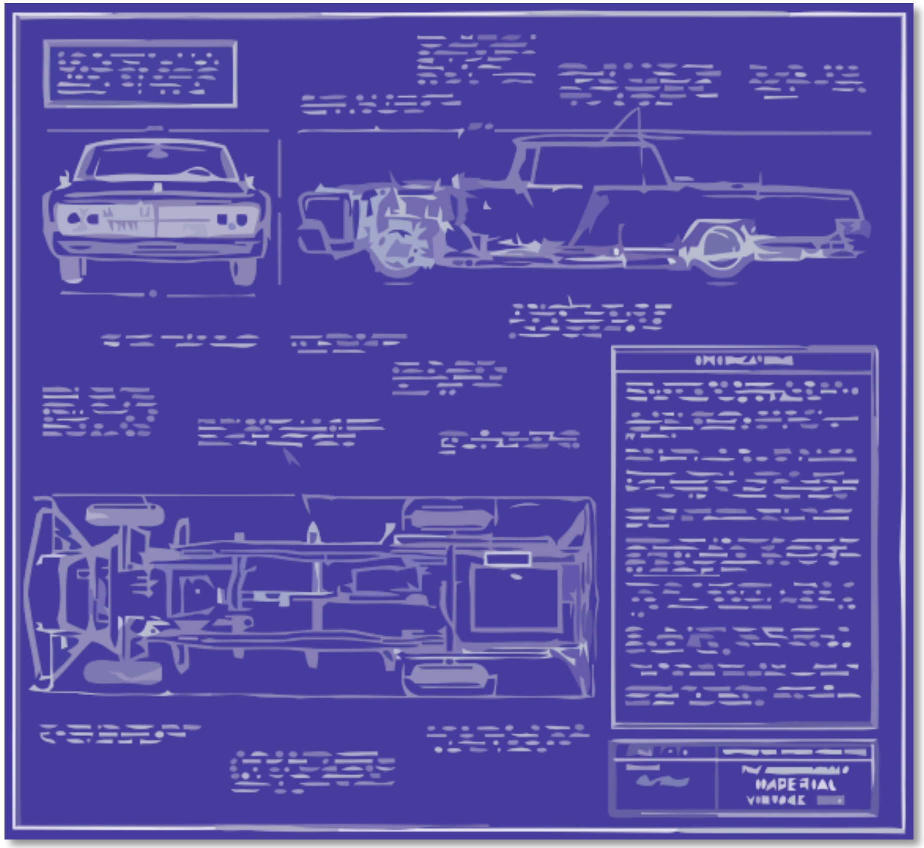
A model captures certain properties of interest of a subject



Human models capture how clothes might look on us (or not)

Models only stand in for the real thing

Blueprints specify a design

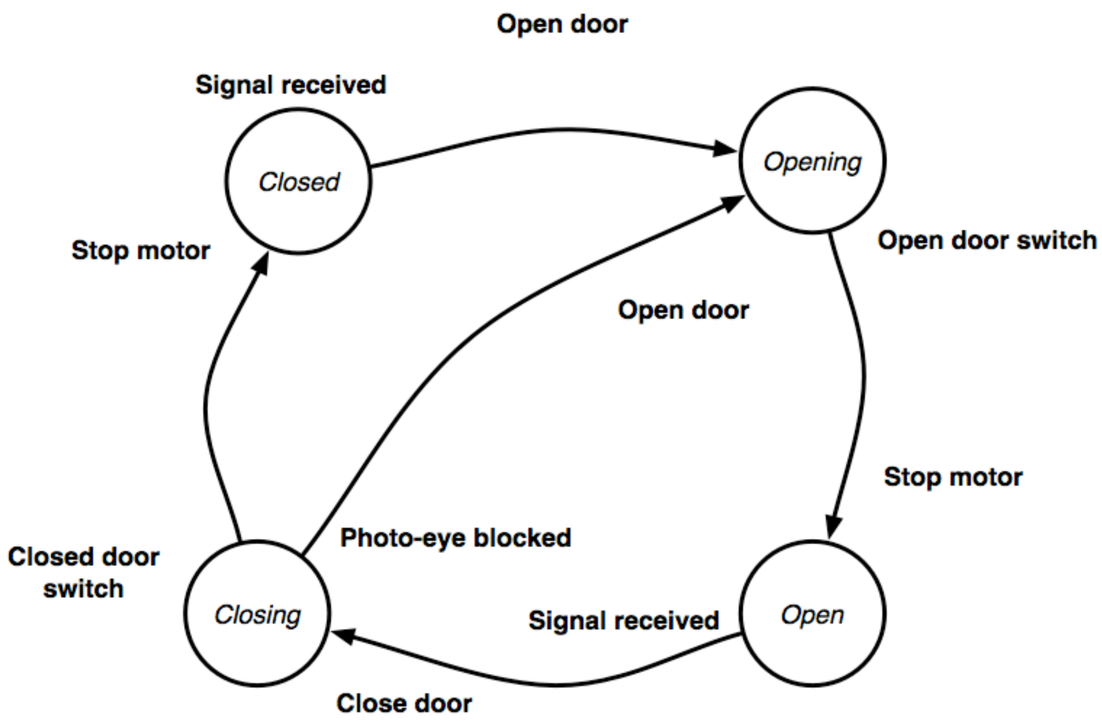


Models as specifications allow us to reason about a thing before we have built it.

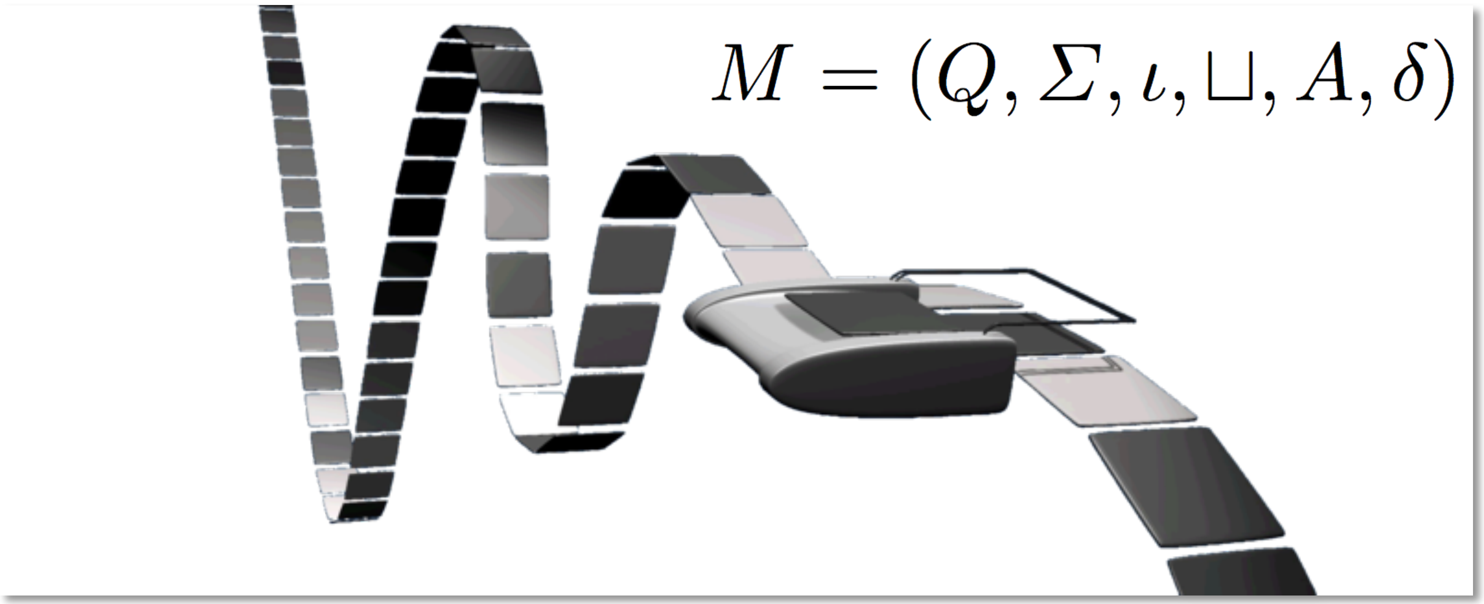
Physical models let us test feasibility of designs

Scale models only capture some aspects of reality.

Finite state models express states and transitions of a process

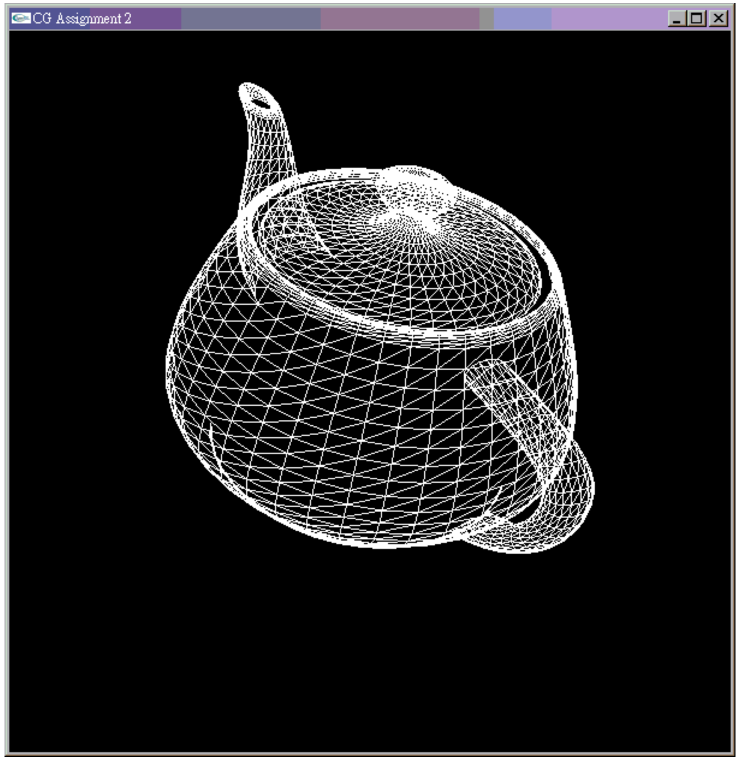


Turing machines model computability



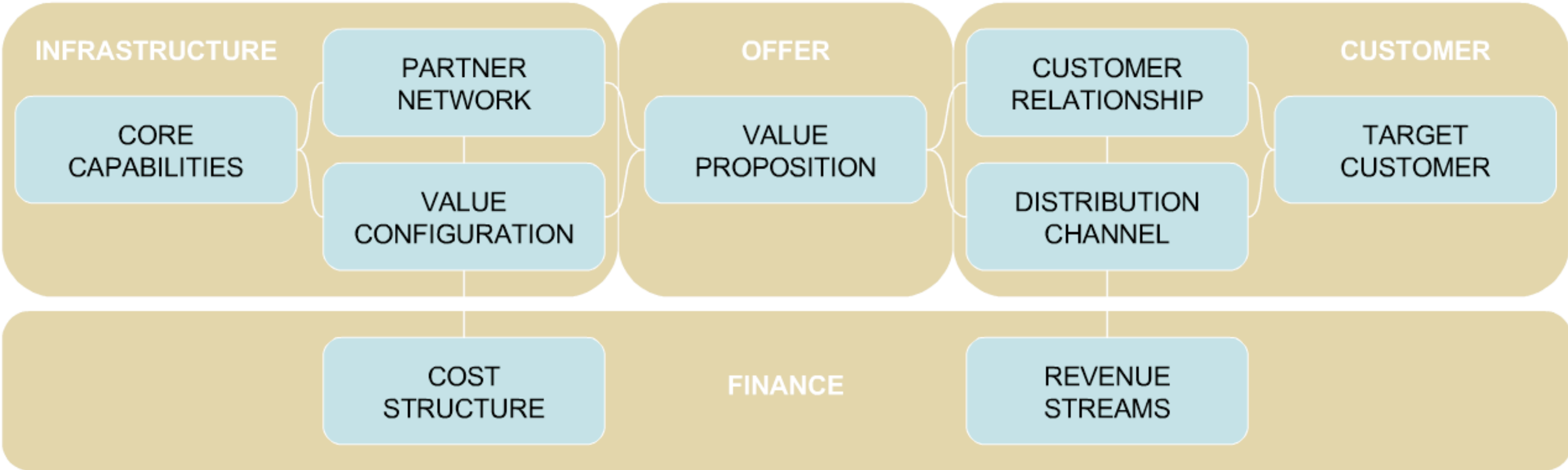
We use mathematical models to reason about computability and complexity

Computer graphics models capture visual aspects of real (or imagined) objects



Different models express different properties

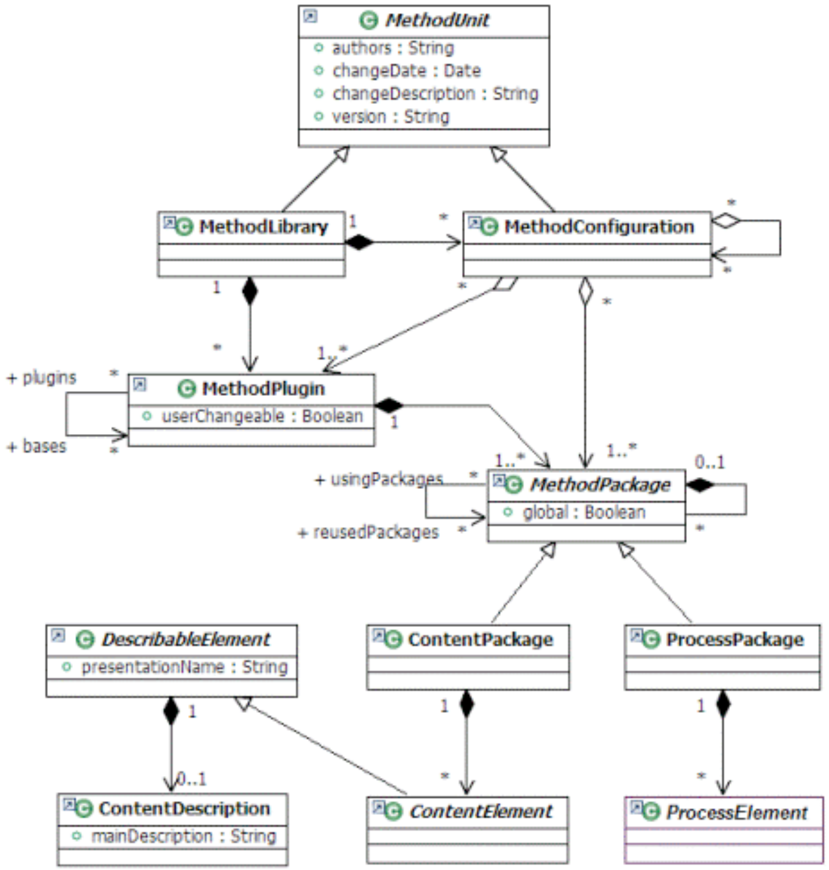
Business models support decision making



A business model expresses the core values of a business

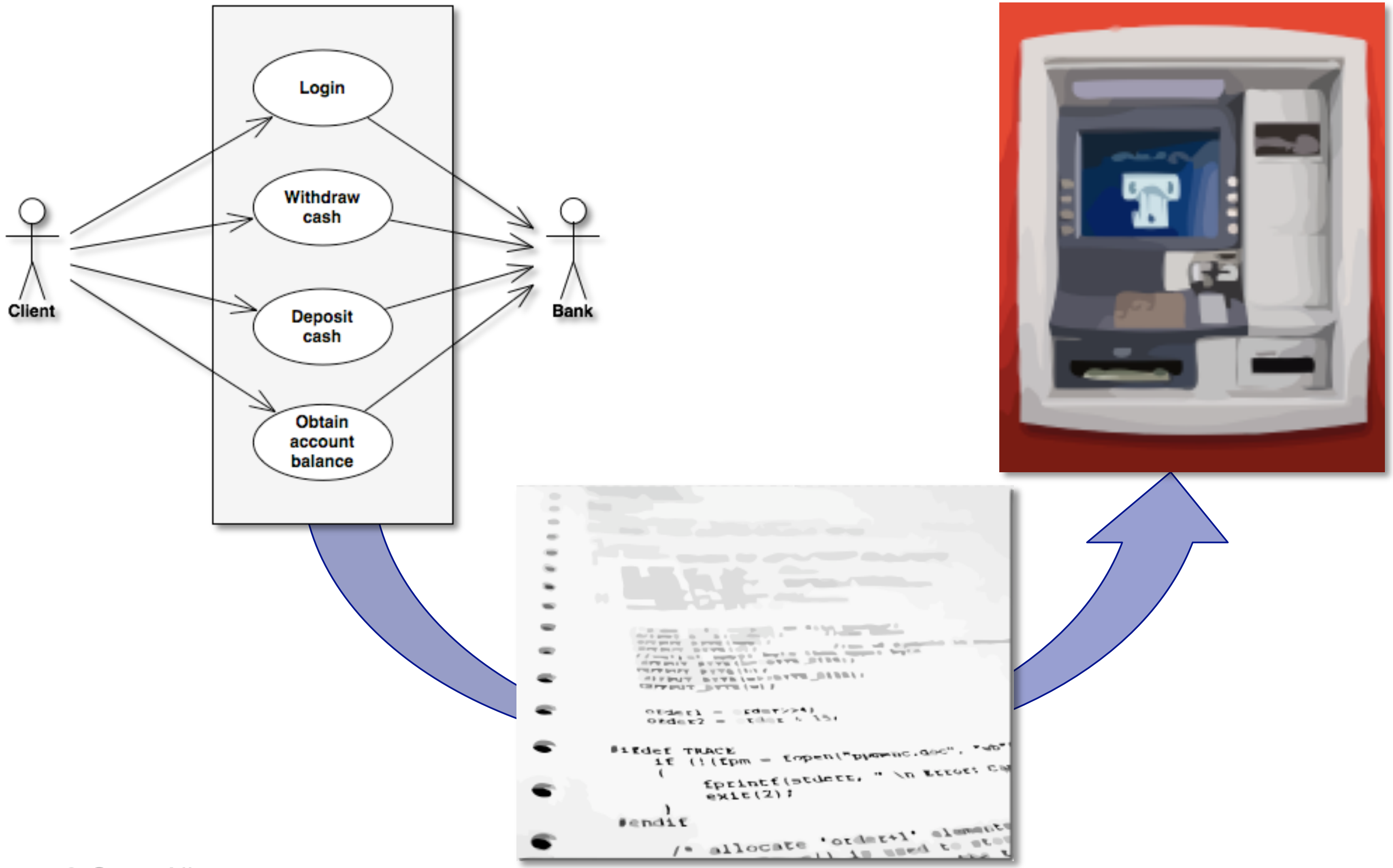
© Oscar Nierstrasz

Software models express architecture and design



The Unified Modeling Language (UML) consists of several different types diagrams to describe software designs, architectures, and requirements models.

Programming is modeling



Roadmap

- > What is Computer Science?
- > Computational Thinking
- > Modeling
- > **Software Engineering**

Software Engineering

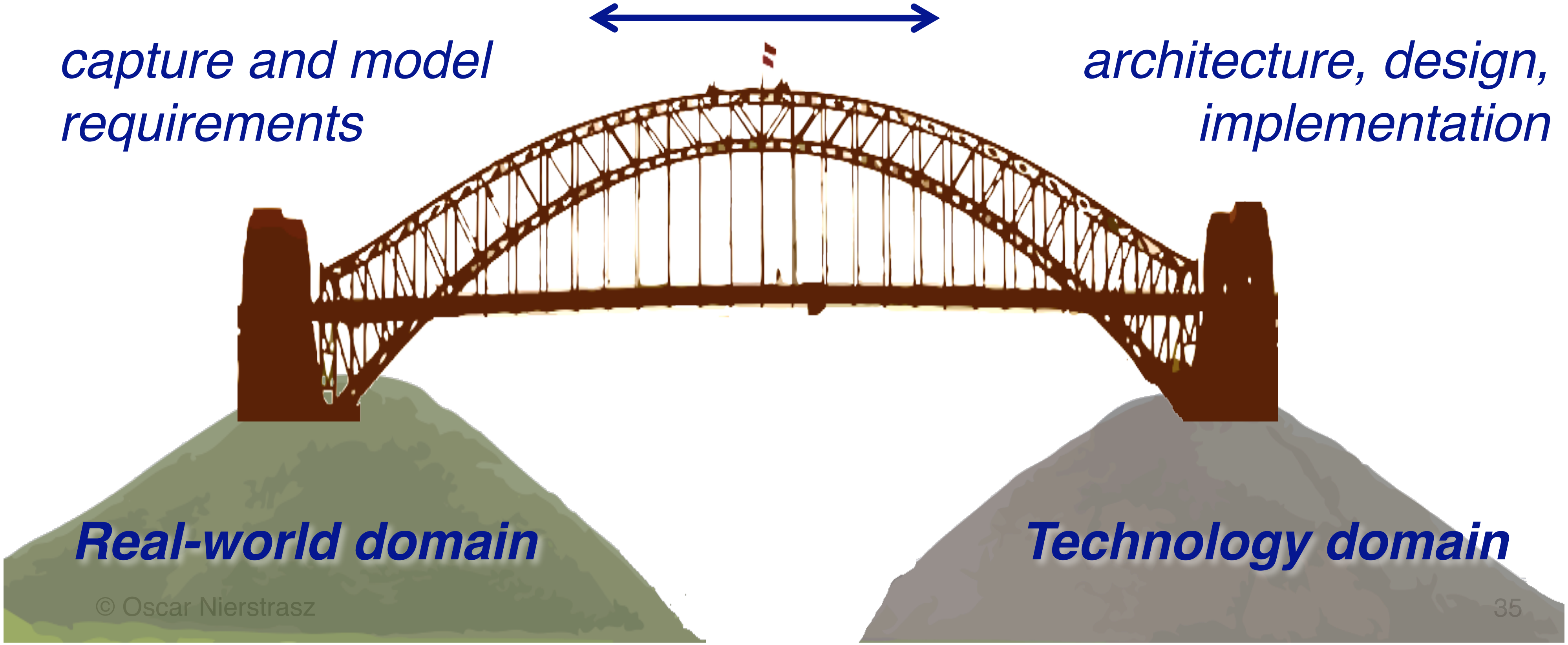
Software Engineering consists of
- processes and *techniques*
- to develop software *products*
- within a given *budget* and *deadline* and
- satisfying *functional* and *quality requirements*

How Software Engineering bridges domains

quality assurance and testing

capture and model requirements

architecture, design, implementation



Conclusion

Computer science requires computational thinking

Programming is modeling

Software Engineering builds bridges between users and technology