Modelling and simulation of physical systems

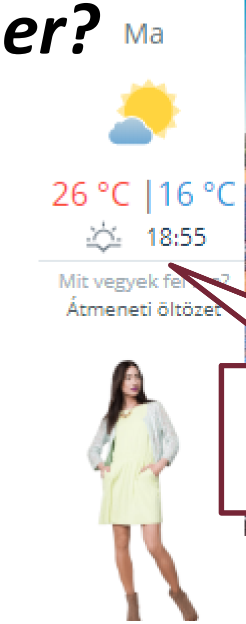
Rebeka Farkas

Budapest University of Technology and Economics
Department of Measurement and Information Systems

Motivation

How to predict the weather?

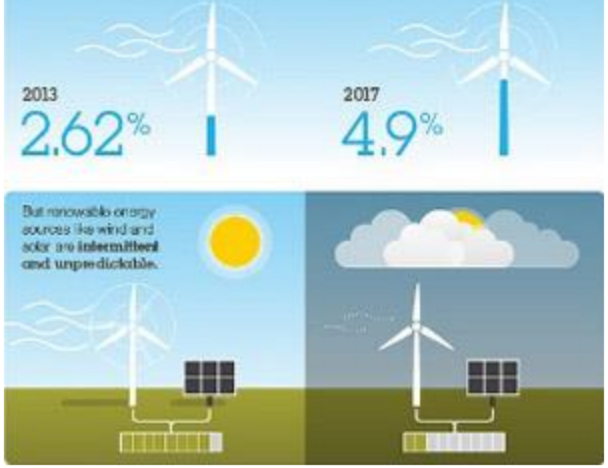
Shall I go surfing this weekend?



Should I take my umbrella today?

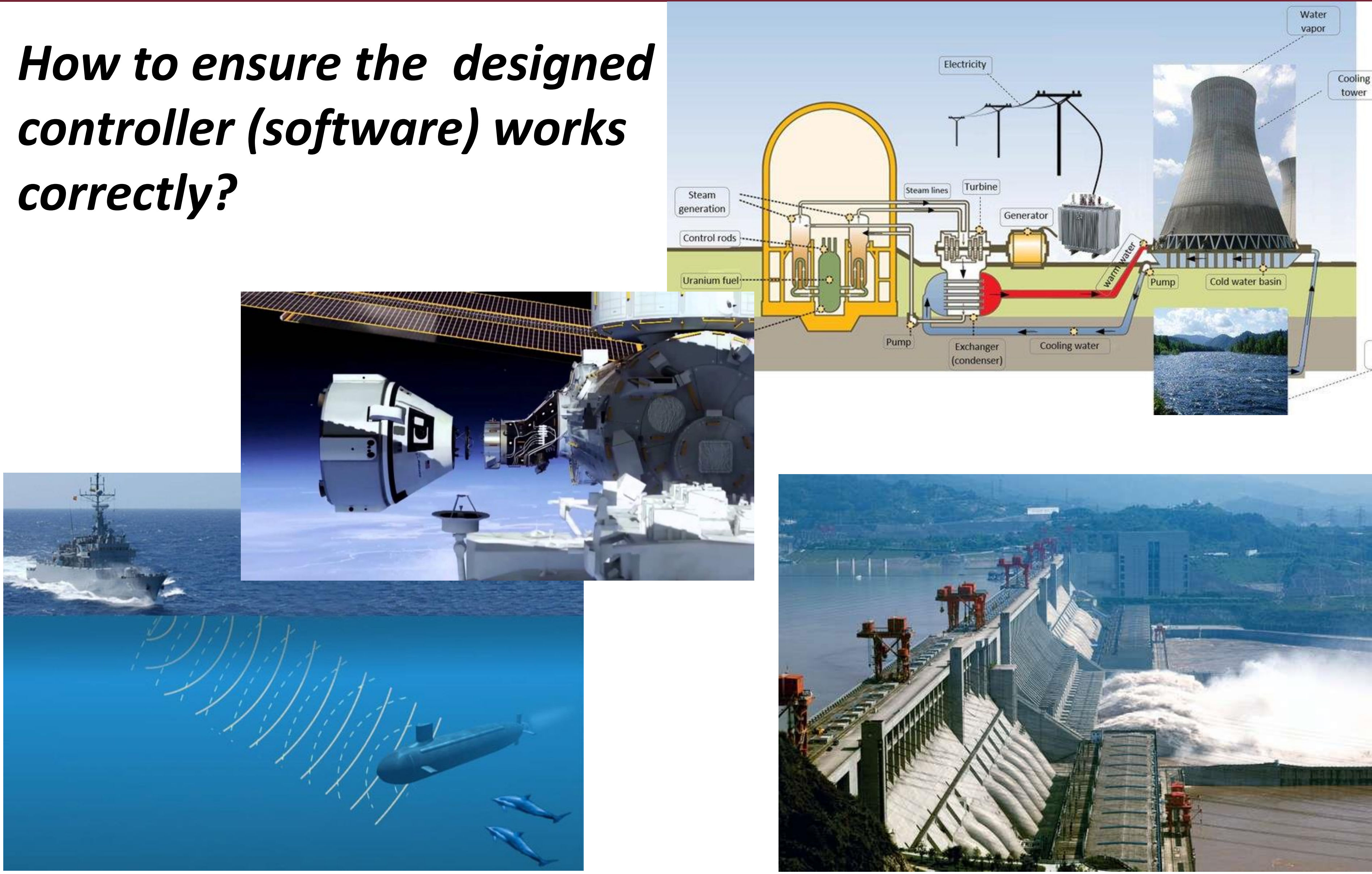
How much enegy will be generated?

Where will the stratospheric balloon land?



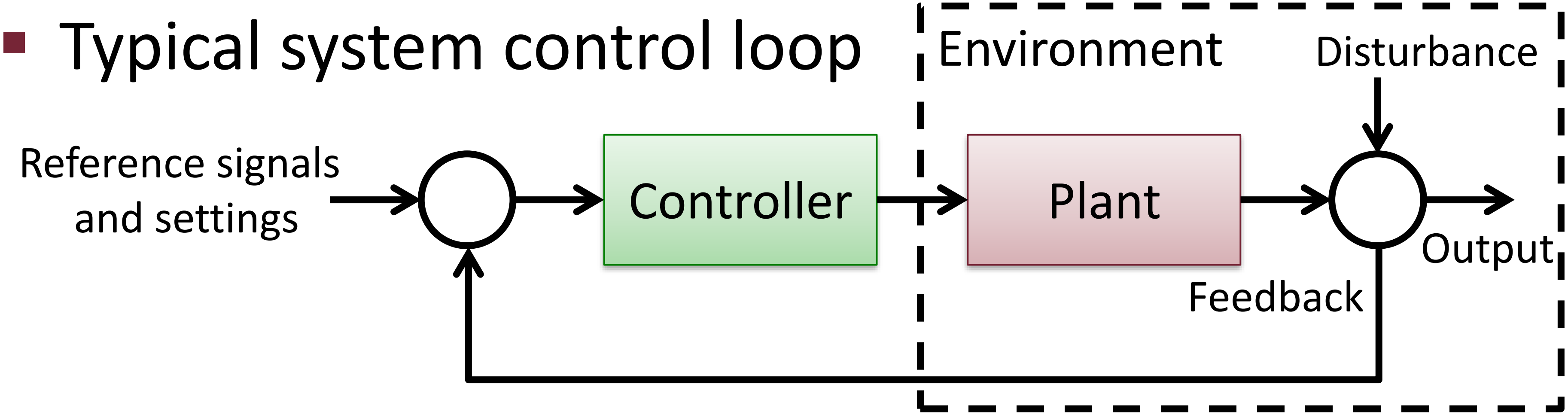
Motivation

How to ensure the designed controller (software) works correctly?



Controller, Plant, and Environment

- Typical system control loop



- Challenge: validate the design of the controller
  - On-site testing and calibration can be
    - Expensive (time + cost)
    - Dangerous
  - Instead: run simulation
    - Must model *physical properties* of plant & environment

Physical systems

- Models up to this point:
  - „Controllable” system, model
  - System under design
  - Modeling goal: facilitate design process, (formal) analysis and verification of internal behaviour

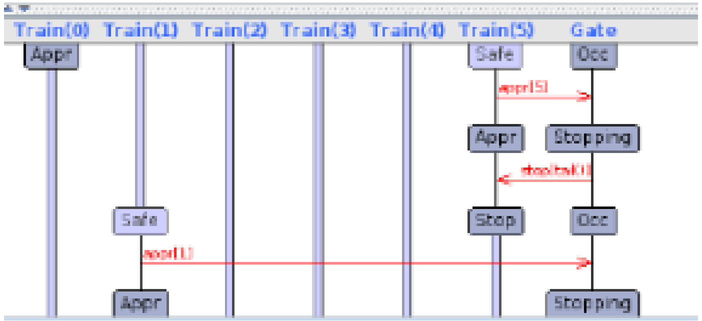
PRESCRIPTIVE

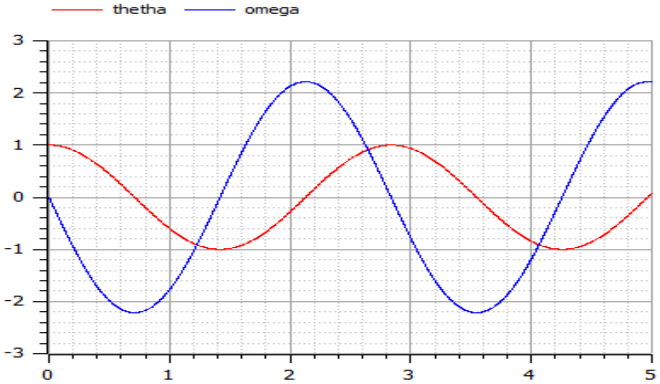
- Physical systems:
  - Laws of physics can not be changed
  - Existing system
  - Modeling goal: assure designed system interacts well with environment

DESCRIPTIVE

Discrete vs continuous models

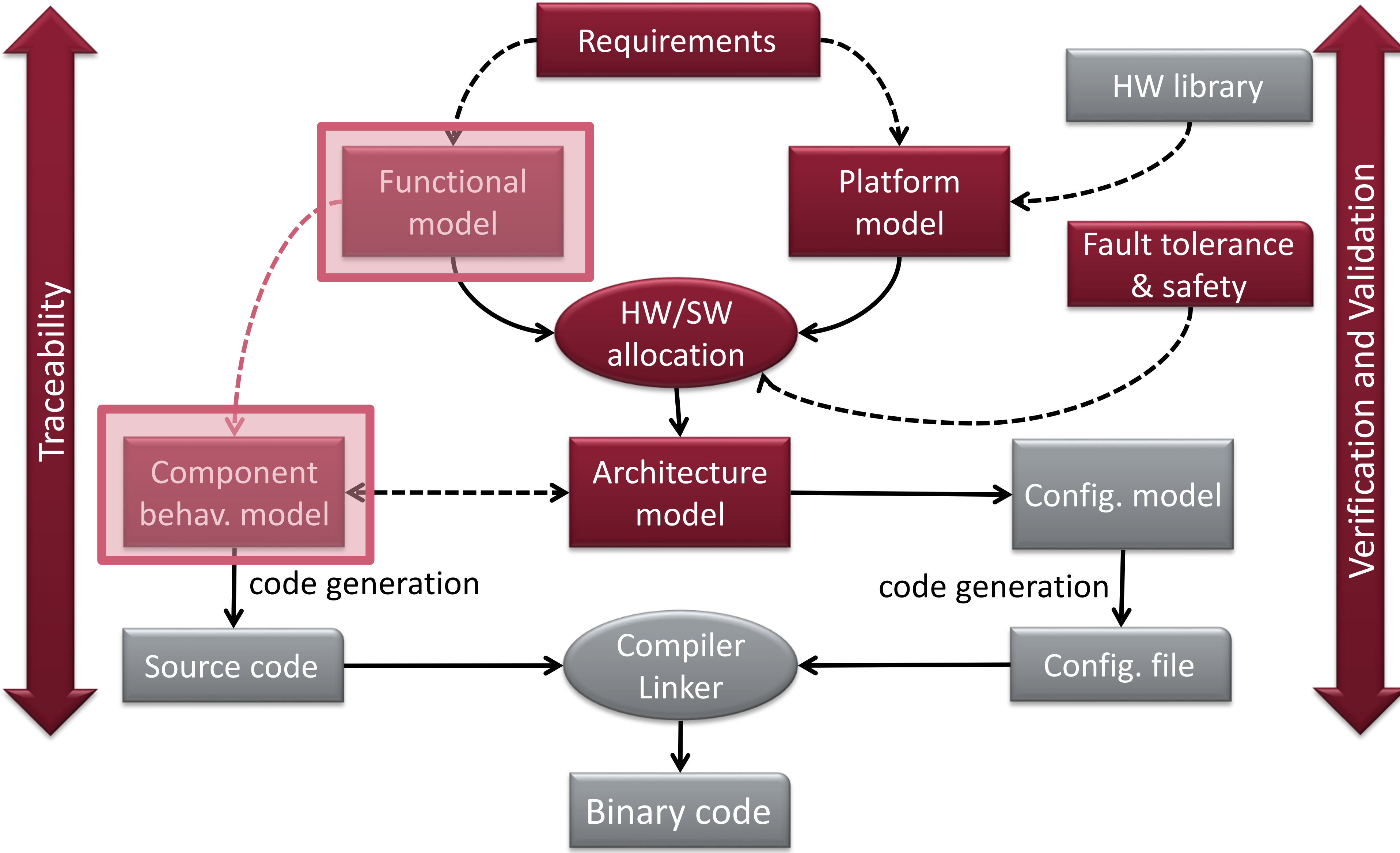
Discrete	**Continuous**
Reactive, state-based models	Differential equation systems
Changes described by assignments	Behaviour described by mathematical equations
Simulation - logical clocks	Simulation – numerically solving equation systems



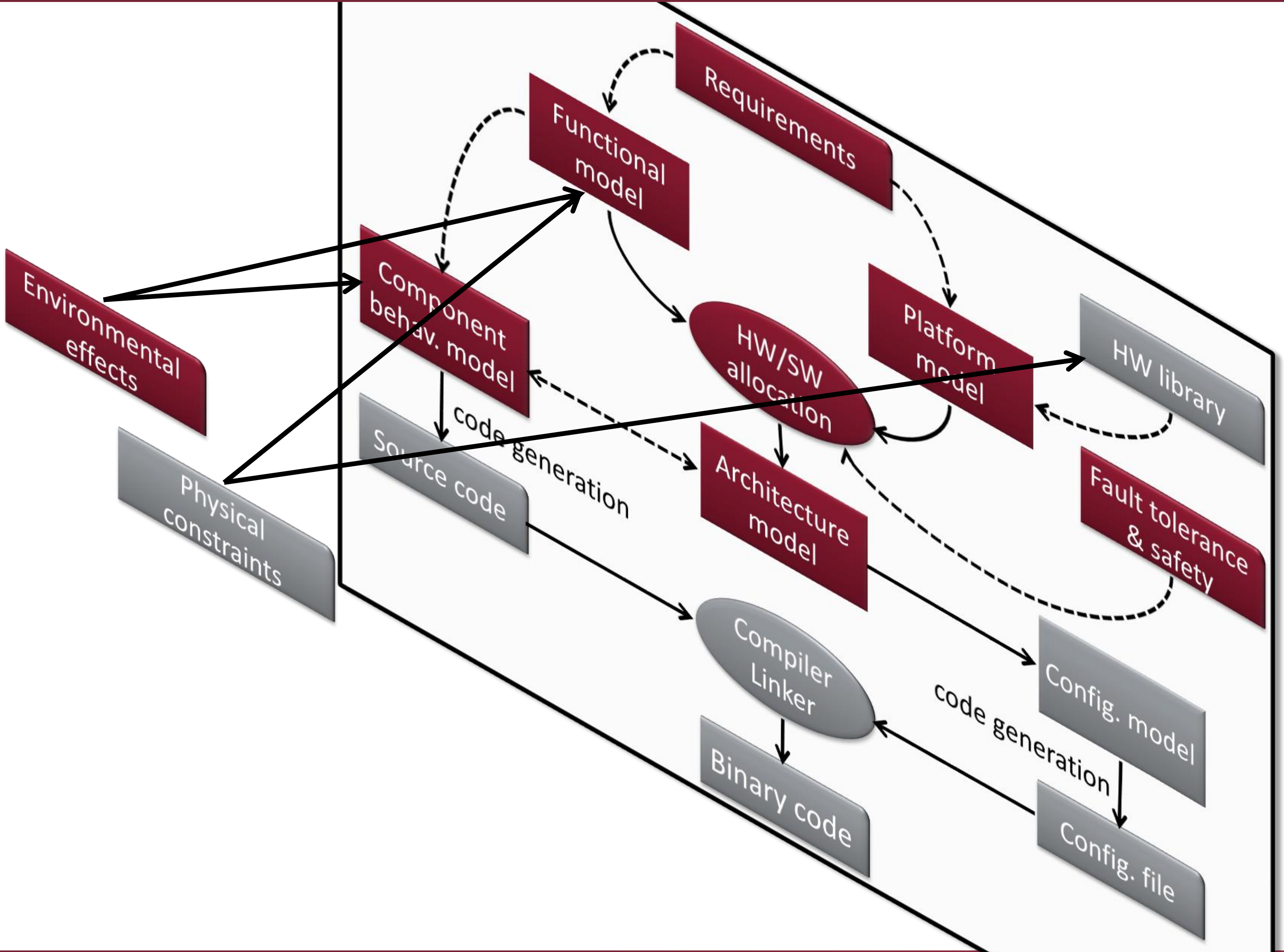


- Reality: Practically all real systems are hybrid
  - Reactive components mixed with continuous changes

Platform-based systems design



Platform-based systems design



Learning Objectives

Modeling physical parameters and constraints

- Describe continuous behaviour of physical systems
- Include rules constraining physical properties
- Capture properties and constraints using the SysML language
- Use the Modelica language to describe physical systems

Simulation of discrete and continuous models

- Work with systems of discrete and continuous states
- Capture both continuous-time and discrete time properties
- Perform discrete event and continuous time simulation
- Understand challenges of simulation in industrial settings

Outline

- Modeling physical systems
- Simulation basics
- Simulation of discrete and continuous systems
- Motivating examples and case studies

Modeling physical systems

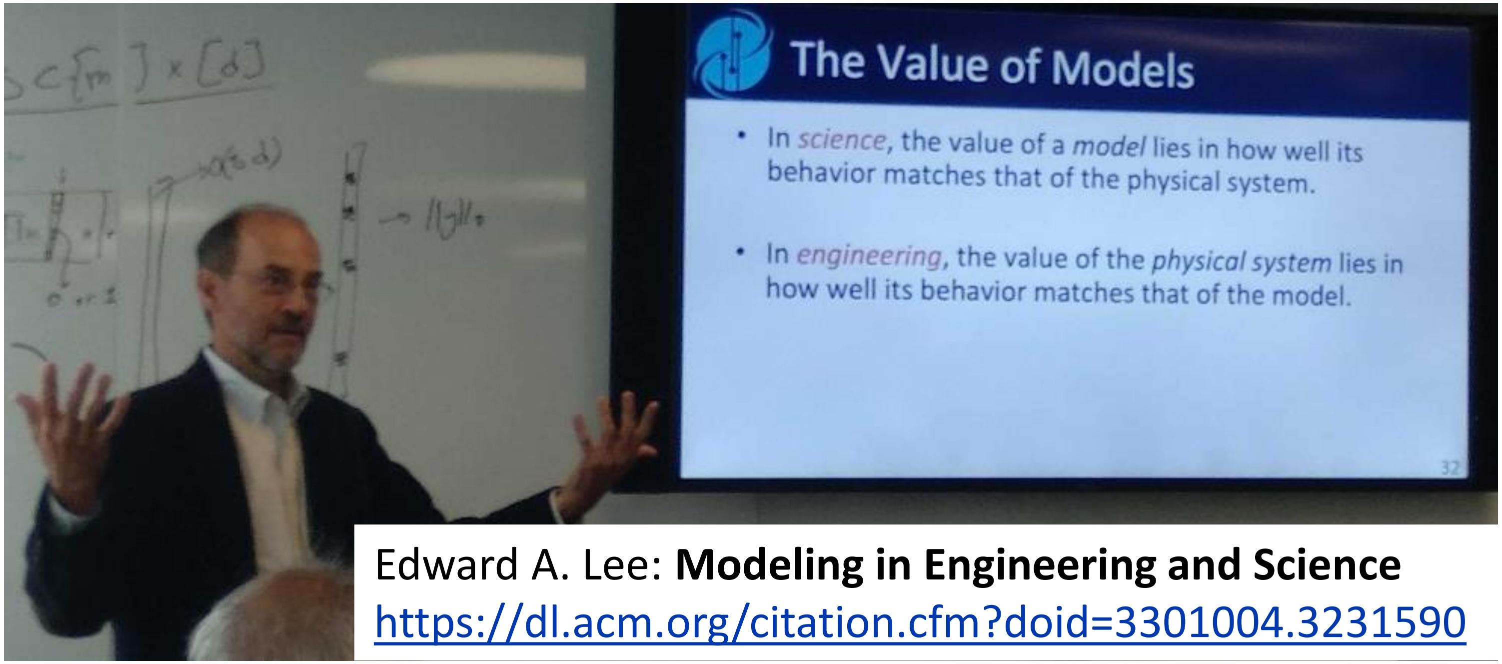
Modeling basics

SysML elements

Modelica

Physical models

Software models	**Physical models**
Usually discrete	Usually continuous
Dissected – system is built by integration of components	In many cases everything has an impact on everything (e.g. weather – temperature)
Understandable, maintainable, usable	„God doesn’t build in straight lines”
Any engineer can create the model	Good model requires domain expertise



The Value of Models

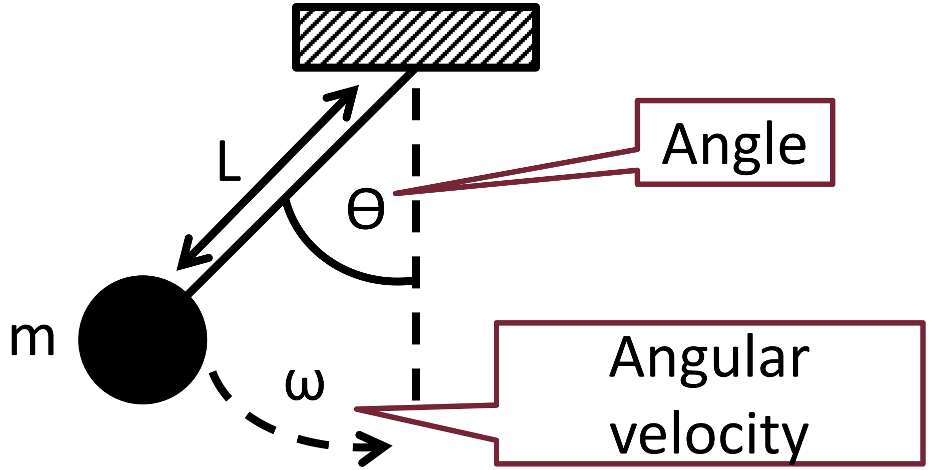
- In *science*, the value of a *model* lies in how well its behavior matches that of the physical system.
- In *engineering*, the value of the *physical system* lies in how well its behavior matches that of the model.

32

Edward A. Lee: **Modeling in Engineering and Science**
https://dl.acm.org/citation.cfm?doid=3301004.3231590

Example: modeling a simple pendulum

- Simple pendulum



- Behavior of the pendulum as a function of time:

Angular acceleration

$$\begin{pmatrix}\dot{\theta}(t)\\ \dot{\omega}(t)\end{pmatrix} = \begin{pmatrix}\omega(t)\\ -\frac{g}{L}sin\theta(t)\end{pmatrix}$$

Assignments and equations

- **Causal** connection ≈ assignment in programming language

$$y := x + 3$$

  - Right-hand-side value determines left-hands-side variable
  - Typical use: to implement controller

- **Acausal** connection ≈ mathematical equation

$$y = x + 3 \Leftrightarrow y - 3 - x = 0$$

  - Always holds; if any of the variables has a new value, it enforces that the other variables change accordingly
  - Typical use: to model behaviour of plant / environment

Requirements

- Model **can be simulated** → Modeling tools
  - Neither over- nor underdetermined equation system (theoretical requirement)
  - Modeling tools have additional constraints
- **Representative** model
  - Obeys physical laws → formulate in model
  - Accurate representation of real-world systems

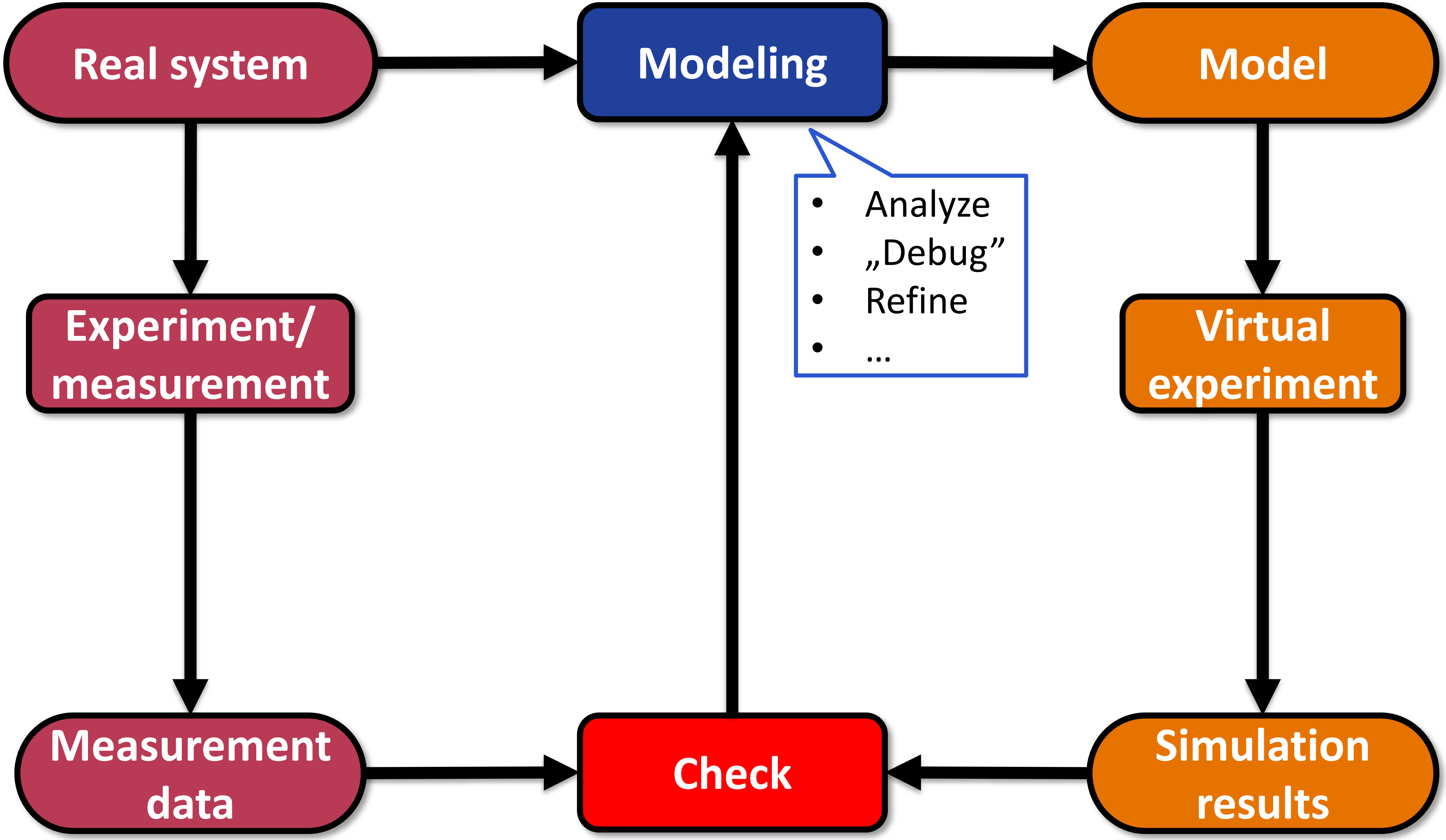
    → compare simulation with real measurements
- General **usability**
  - Maintainable, reusable, etc. → block-based modeling

How to create a model

1. Decompose the system
2. Customize existing components
   - Better to use components provided by tools → (Just like programming languages)
   - Assign parameter bindings
3. Adjust connections
4. **Check model accuracy**

- Accurate modeling is difficult
  - Models are created by domain experts
  - There are complete books on modeling
- Simulation can be used for verification
  - (non-exhaustive, just like testing)

Checking physical model accuracy



- Analyze
- „Debug”
- Refine
- …

Modeling physical systems

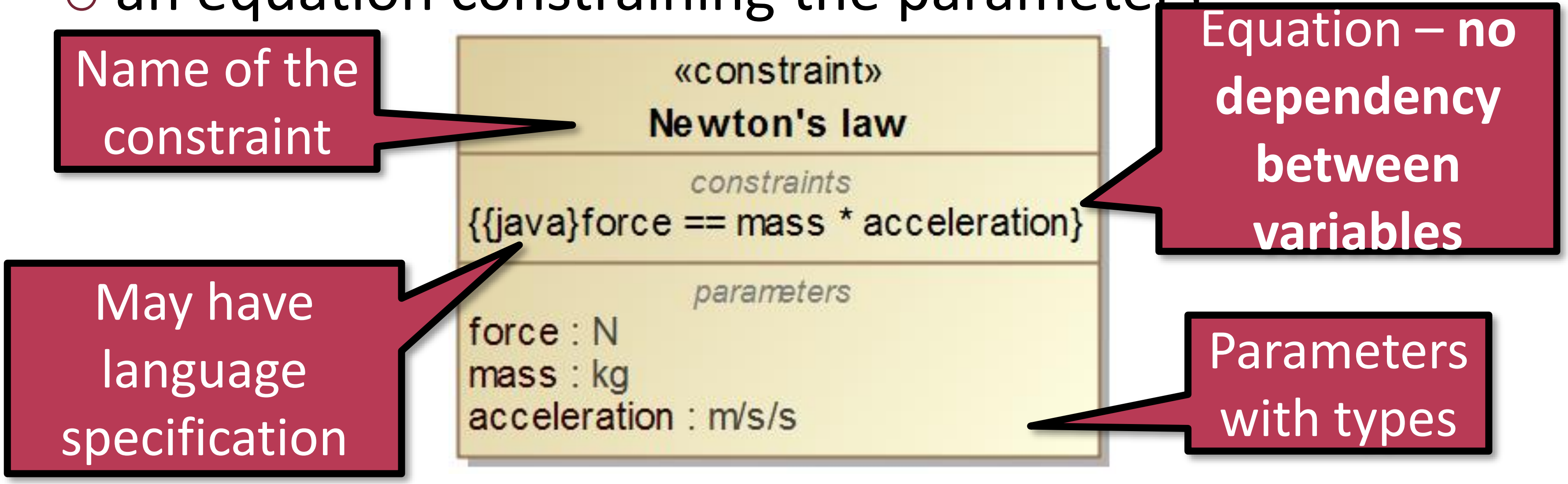
Modeling basics

SysML elements

Modelica

Constraint blocks

- **Constraint**: equations with parameters bound to the properties of the system
- **Constraint block**: supports the definition and the reuse of constraints. It holds
  - a set of parameters and
  - an equation constraining the parameters



«constraint»
Newton's law

constraints
{{java}force == mass * acceleration}

parameters
force : N
mass : kg
acceleration : m/s/s

Name of the constraint

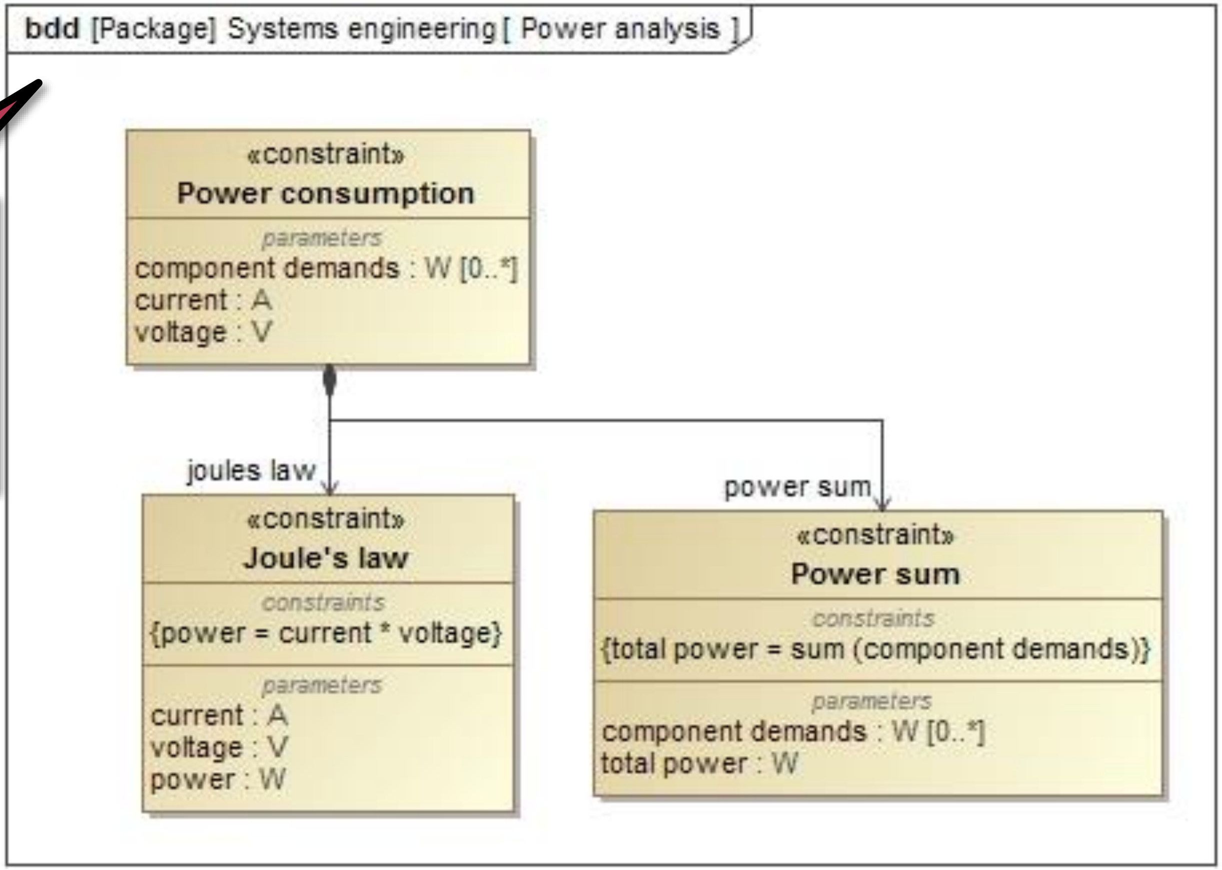
Equation – **no dependency between variables**

May have language specification

Parameters with types

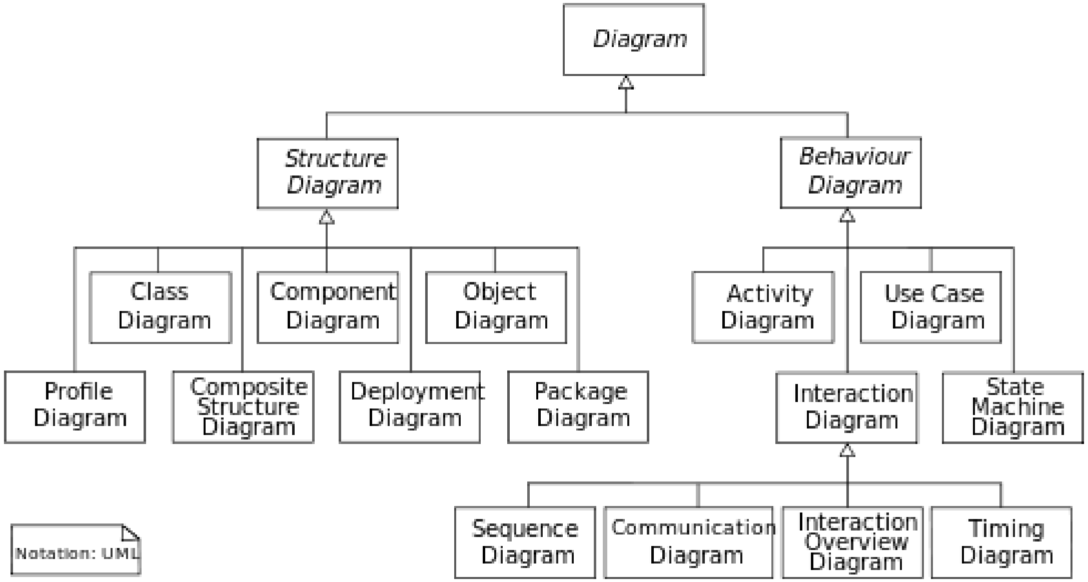
Constraint definition

- **Composition** is used to define complex constraints from simple equations

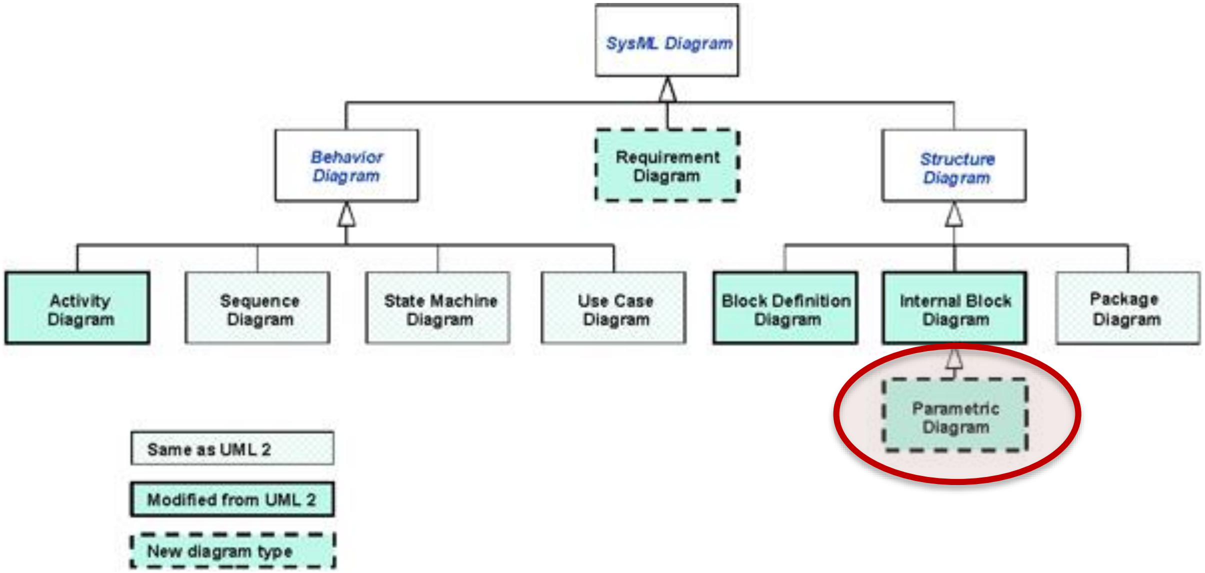


Hierarchy depicted in a BDD

Parametric Diagram (PAR)



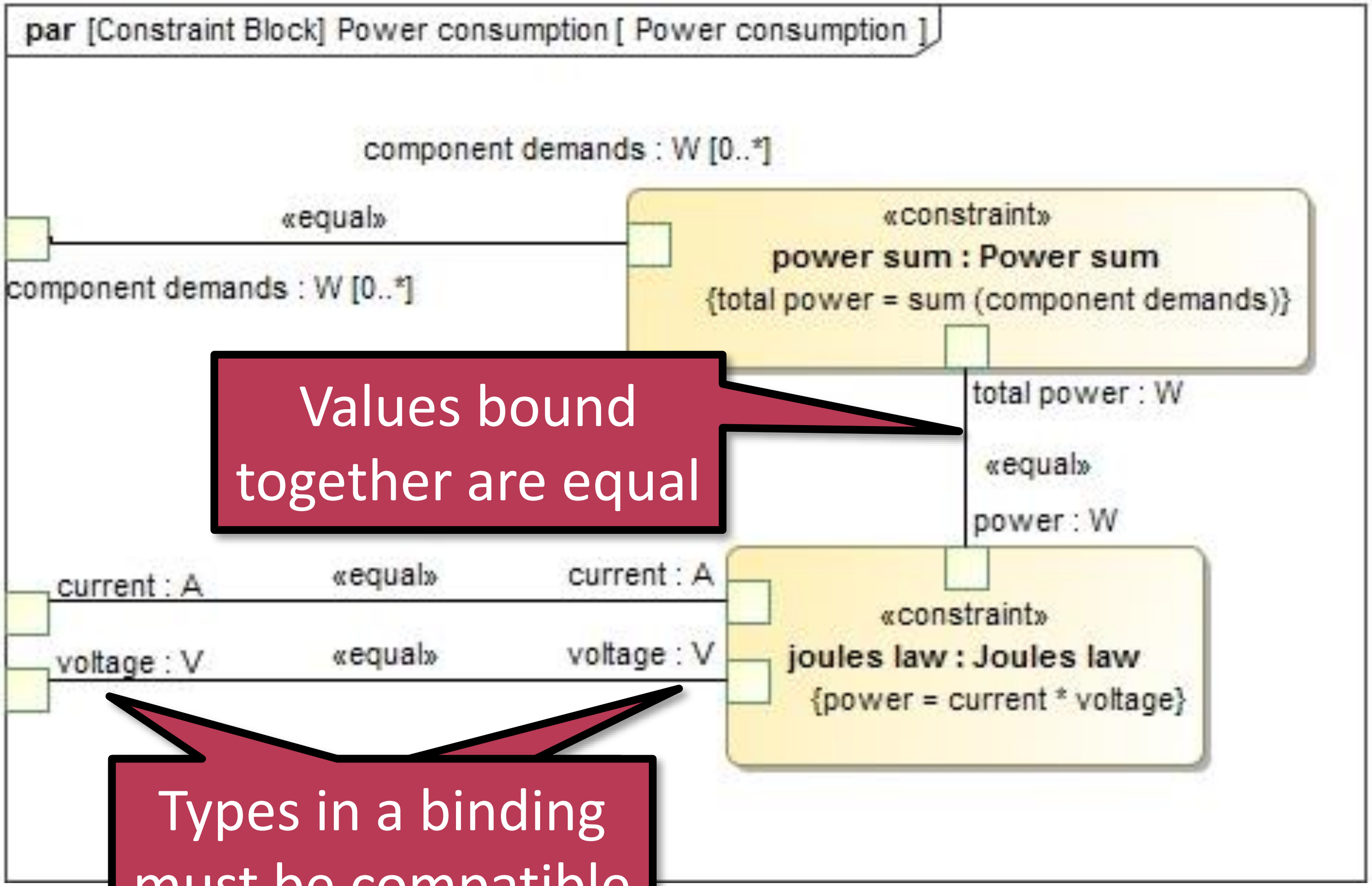
UNIFIED
MODELING
LANGUAGE



OMG
SYSTEMS
MODELING
LANGUAGE

Parameter bindings

- Goal: describe the application of constraints in a particular context



Values bound together are equal

Types in a binding must be compatible

Applications of parametrics

- Parametric specification
  - Define parametric relationships in the system structure
- Parametric analysis
  - Evaluating constraints on the system parameters to calculate values and margins for a given context
  - Checking design alternatives
  - Tool support: ParaMagic plug-in for MagicDraw
- Exact values may come from other sources
  - There are modeling standards with better support for this modeling aspect…
  - …such as Modelica

Modeling physical systems

Modeling basics

SysML elements

Modelica

Modeling Tools

- Modelica
  - OMEdit
  - Dymola
- Matlab/Simulink
- Domain specific tools
  - Ansys Simplorer (electrical systems)
  - AUTOSAR → Course: BMEVIMIAV15
  - CANoe (Engine control unit)

Overview of Modelica

- **Modelica** is an object-oriented, equation-based language designed to model complex physical systems containing process-oriented subcomponents of different nature
  - Describing both continuous-time and discrete-time behaviour
- The **Modelica Standard Library** provides more than 1000 ready-to-use components from several domains
  - Full high-school + 1st year university physics (and much more)
- Implementations
  - Commercial e.g. by Dymola, Maplesoft, Wolfram MathCore
  - Open-source: JModelica
- Modeling and simulation IDE: OpenModelica - OMEdit

Example: Modelica code for simple pendulum

Model name

Continuous time variables, constants

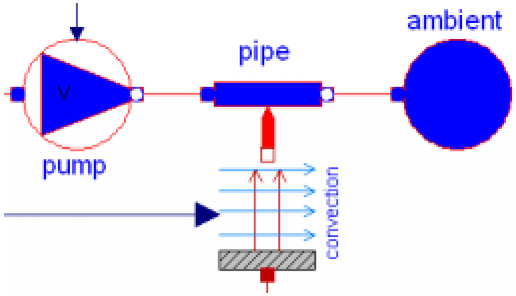
```
model SimplePendulum
    parameter Real L=2.0;
    constant Real g=9.81;
    Real theta (each start = 1.0);
    Real omega;
equation
   der(theta) = omega;
   der(omega) = -(g/L)*theta;
 end SimplePendulum;
```

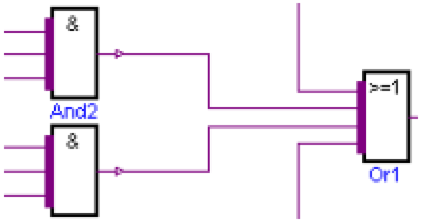
Initial value

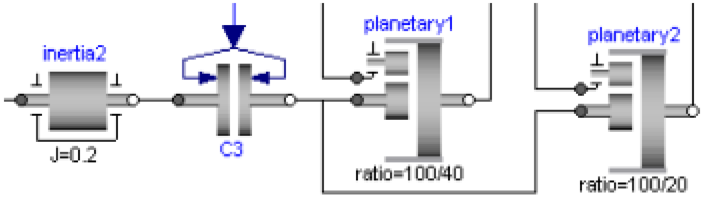
(Differential) equations

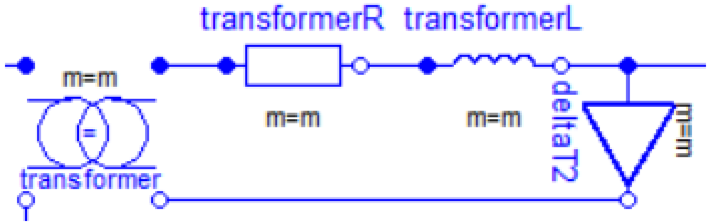
Modelica Standard Library

- Provides reusable building blocks (called classes) for Modelica models
- Version 3.2.1. has more than 1340 classes and models
- Various domains







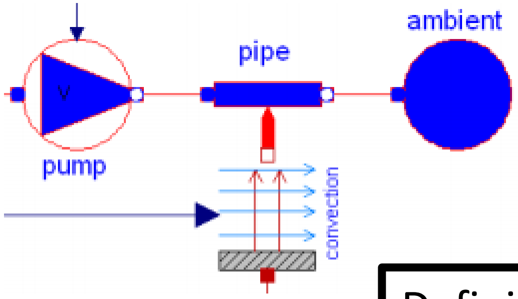


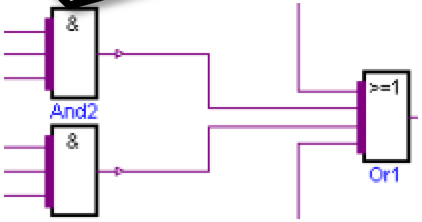
Modelica Standard Library

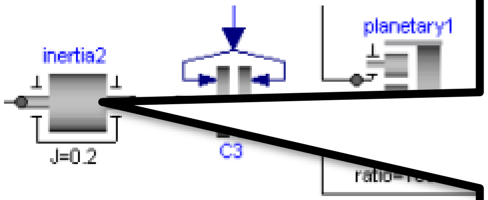
- P
  M
- V
- V

Definition in Modelica textual syntax:

```
equation
  auxiliary[1] = x[1];
  for i in 1:n - 1 loop
    auxiliary[i + 1] = D.Tables.AndTable[auxiliary[i], x[i + 1]];
        end for;
  y = pre(auxiliary[n]);
```





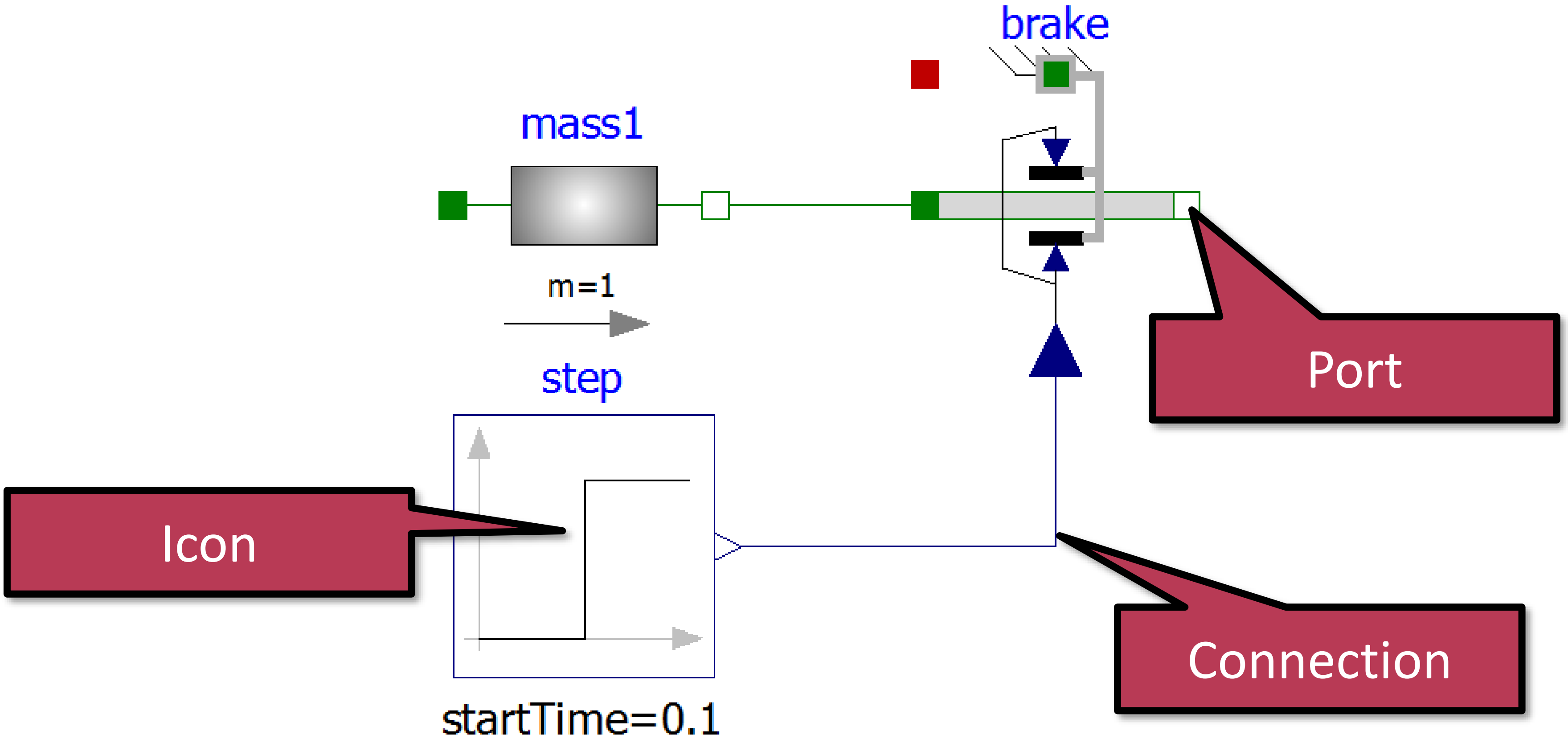


Definition in Modelica textual syntax:

```
equation
        phi = flange_a.phi;
        phi = flange_b.phi;
        w = der(phi);
        a = der(w);
        J*a = flange_a.tau + flange_b.tau;
```

Example plant model – train brakes

- Physical model for braking system carrying a mass



- Graphical notation in OpenModelicaEditor (~**ibd**)

Simulation

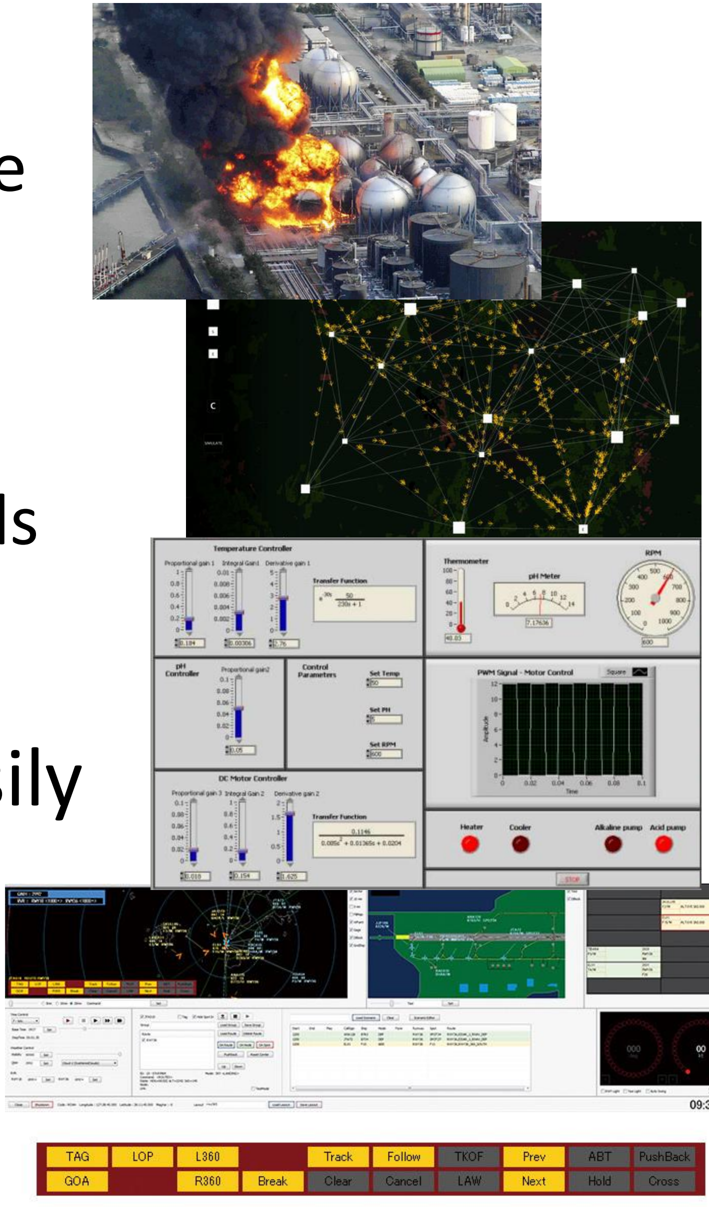
- Simulation basics
- Discrete systems
- Timed systems
- Continuous systems
- Practical aspects

Model-based verification

- Modeling
    - Builds an abstract mathematical representation
- *Simulation*
    - Executes (some parts of) the **behaviour model**
    - Virtual experiment
- Testing
    - Executes (some parts of) the real system
- Other types of verification
    - Formal verification
    - Monitoring (can be used on simulated models)

Advantages over real experiments

- Real system does not change
  - Error in simulation does not cause real problem
  - Model can simply be reset
- Simulation is much faster
  - Hours can be simulated in seconds
  - (Real-time simulation is also possible)
- Parameters can be adjusted easily
- Easier to analyze
  - Can be controlled, replayed
  - No need for complex monitoring system



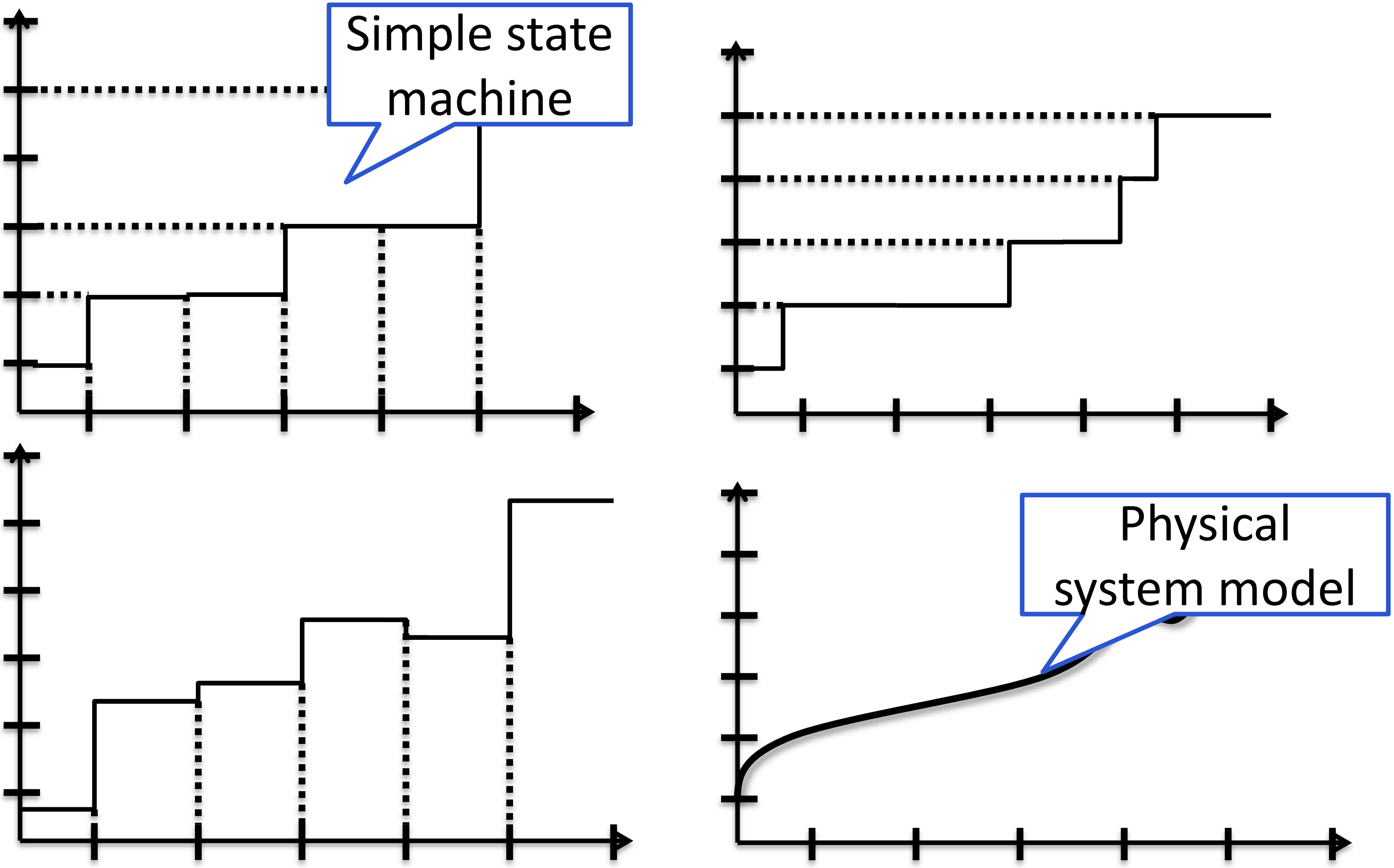
Types of behavior models

Discrete time

Continuous time

Discrete system state

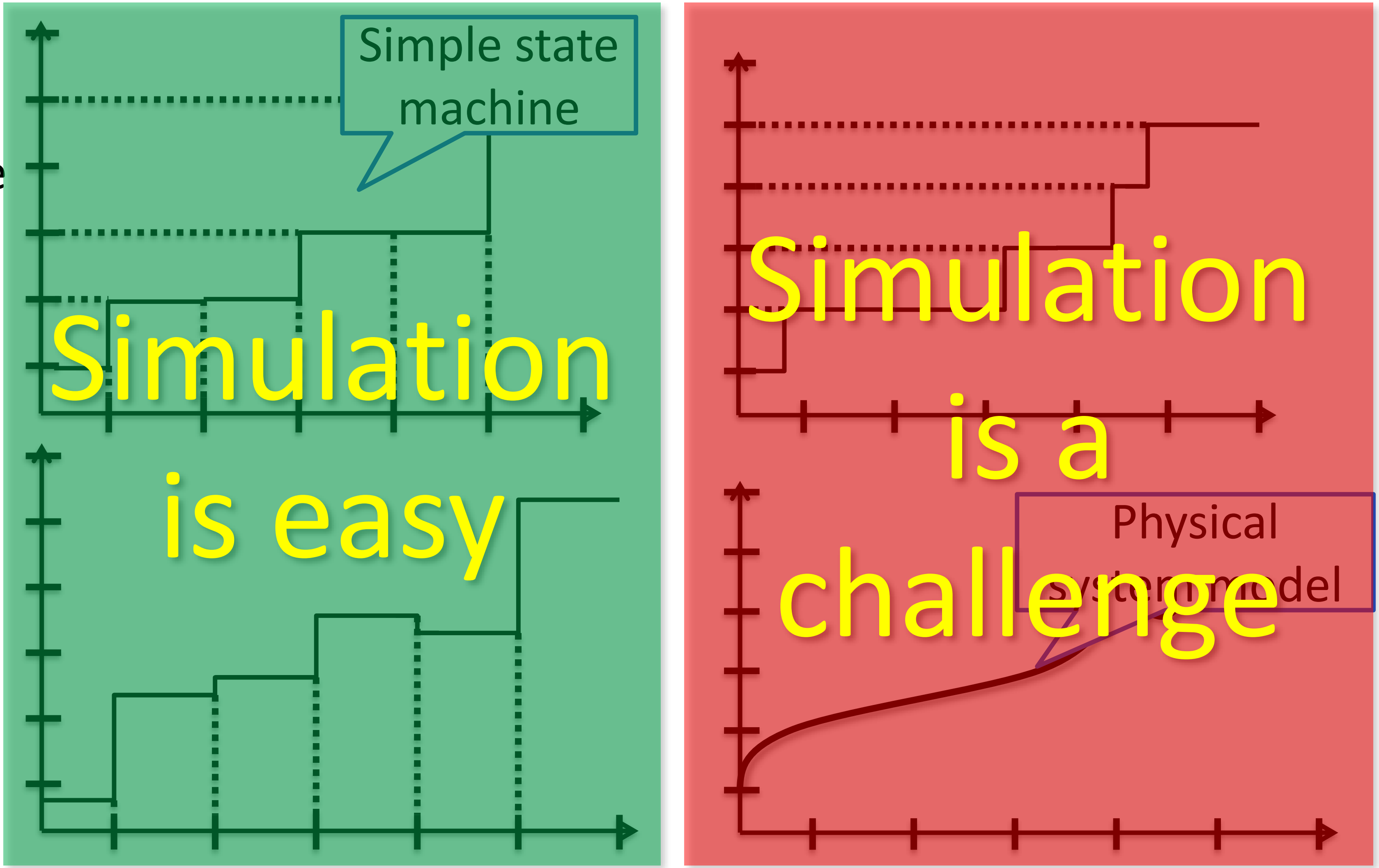
Cont. system state



Simple state machine

Physical system model

Types of behavior models



Discrete time

Continuous time

Discrete system state

Cont. system state

Simple state machine

Physical system model

Simulation is easy

Simulation is a challenge

Types of simulation

- DES: Discrete event simulation
  - Event-based model (e.g. timed state machine)
  - Simulation step by step
    - Considering events (timestamps), guards and actions
    - Event queue – order is important
  - Challenges: synchronization
- Time stepped dynamic model
  - Continuous model
    (performance model, physical model)
  - Problem: discrete time simulator

  → Discretization of time

Simulation

Simulation basics

Discrete systems

Timed systems

Continuous systems

Practical aspects

Goals of System Simulation

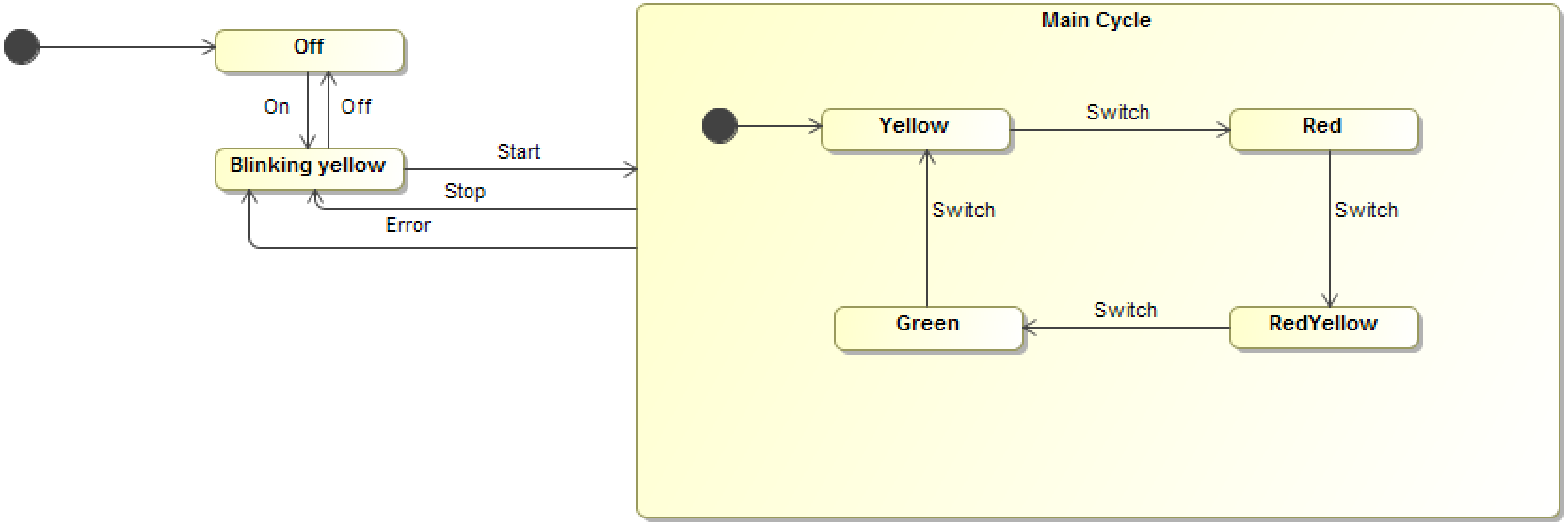
- Check the design of the system
  - Material flows – are there bottlenecks?
  - Queue locations and sizes – do they get blocked or starved?
  - Resources – are they sufficient, do they starve important operations?
  - Failure modes – what are they and what causes them?
- Check if it has the required capacity
- See what different types of downtime do to performance
- Improve the design

Components of a D. E. Simulation

- Simulations contain
  - ***Events*** causing changes in the system state
    - ***Event space*** – set of possible events: input events, timed events, etc.
  - ***Queues*** where entities wait their turn
    - Significant in case of *asynchronous* communication
    - *Synchronous* systems – **logical clock** model
- Only one event at once
  - Two events can have the same *time stamp* but they have to have an *order* (see: state charts with *exit*, *entry* and *do* actions)

Example: Traffic Light

- Events: On, Off, Start, Stop, Switch, Error



- Simulator: a program that can tell what state the system is in – given an input event sequence
  - Possible, even manually
  - MagicDraw: CAMEO toolkit

Simulation

Simulation basics

Discrete systems

Timed systems

Continuous systems

Practical aspects

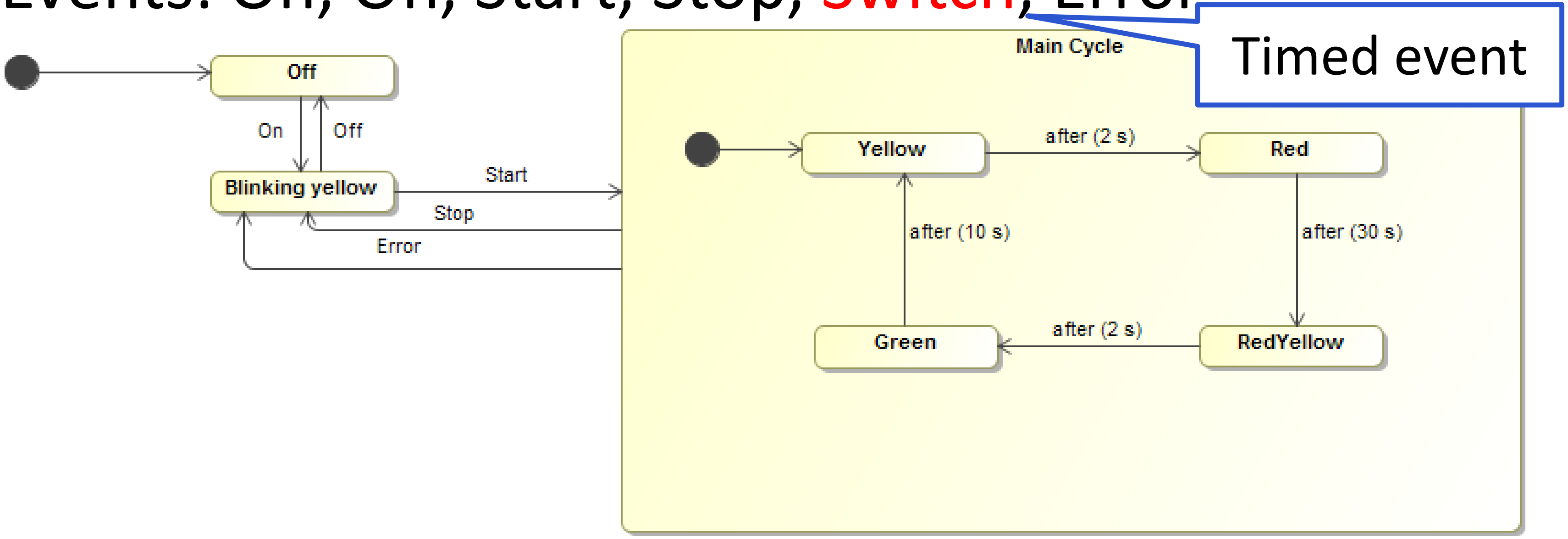
Variables and time

- Sytem state:
  - Represented by variables (explicit or implicit)
  - Continuous/discrete
- Representation of time
  - Continuous/discrete (logical clock - tick)
- Simulation uses virtual time
  - Virtual time ≠ runtime
  - (Except for real time simulation)
- Time is a variable!
  - *Although a special one*

If the only continuous component is time, discrete simulation is still possible.

Example: Timed Traffic Light

- Events: On, Off, Start, Stop, ~~Switch~~, Error



Timed event

- Simulator: a program that can tell what state the system is in – given an input event sequence incl time delays
  - Possible, even manually
  - MagicDraw: CAMEO toolkit

Simulation

Simulation basics

Discrete systems

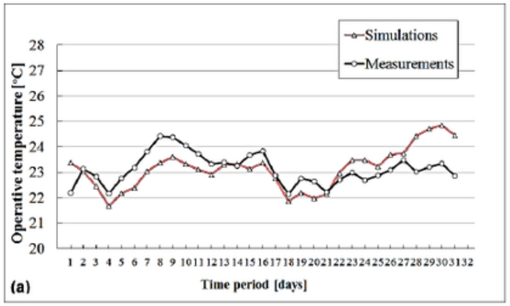
Timed systems

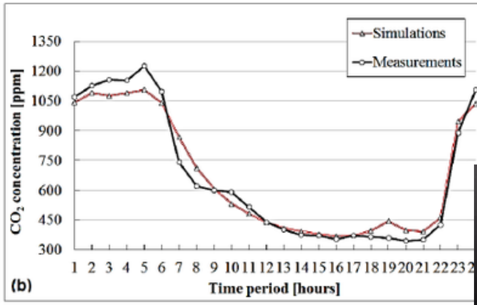
Continuous systems

Practical aspects

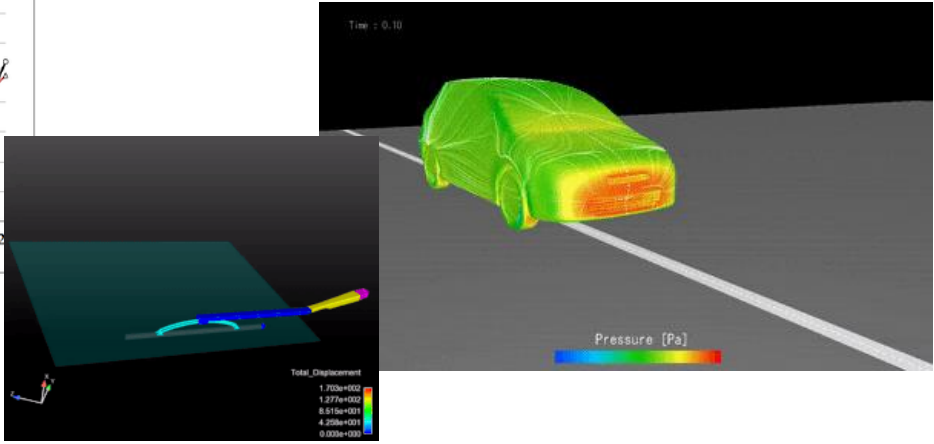
Goals of system simulation

- Ensure model correctness

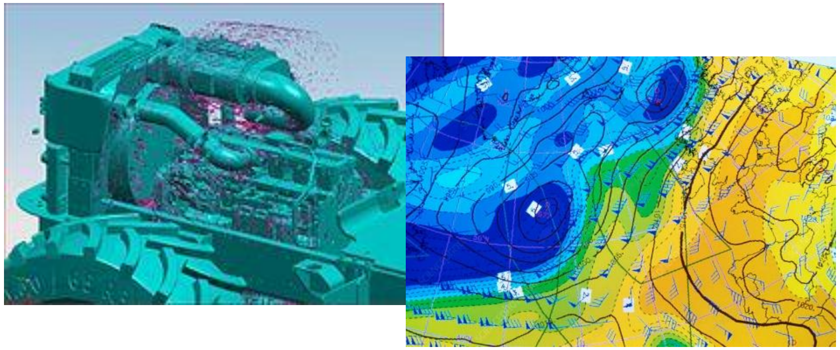




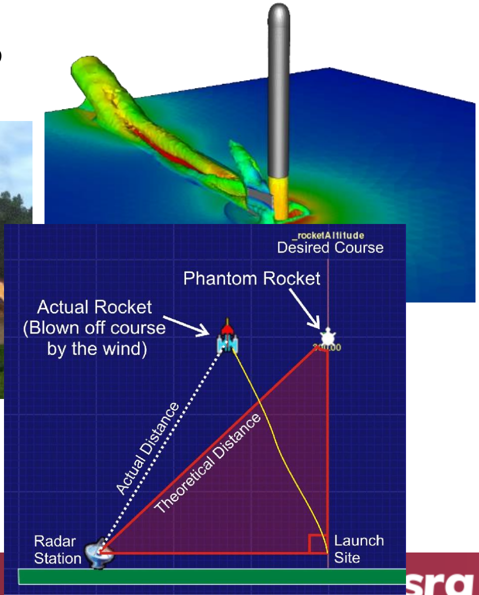
- Ensure correct interactions with designed system



- Analyse/predict system behaviour



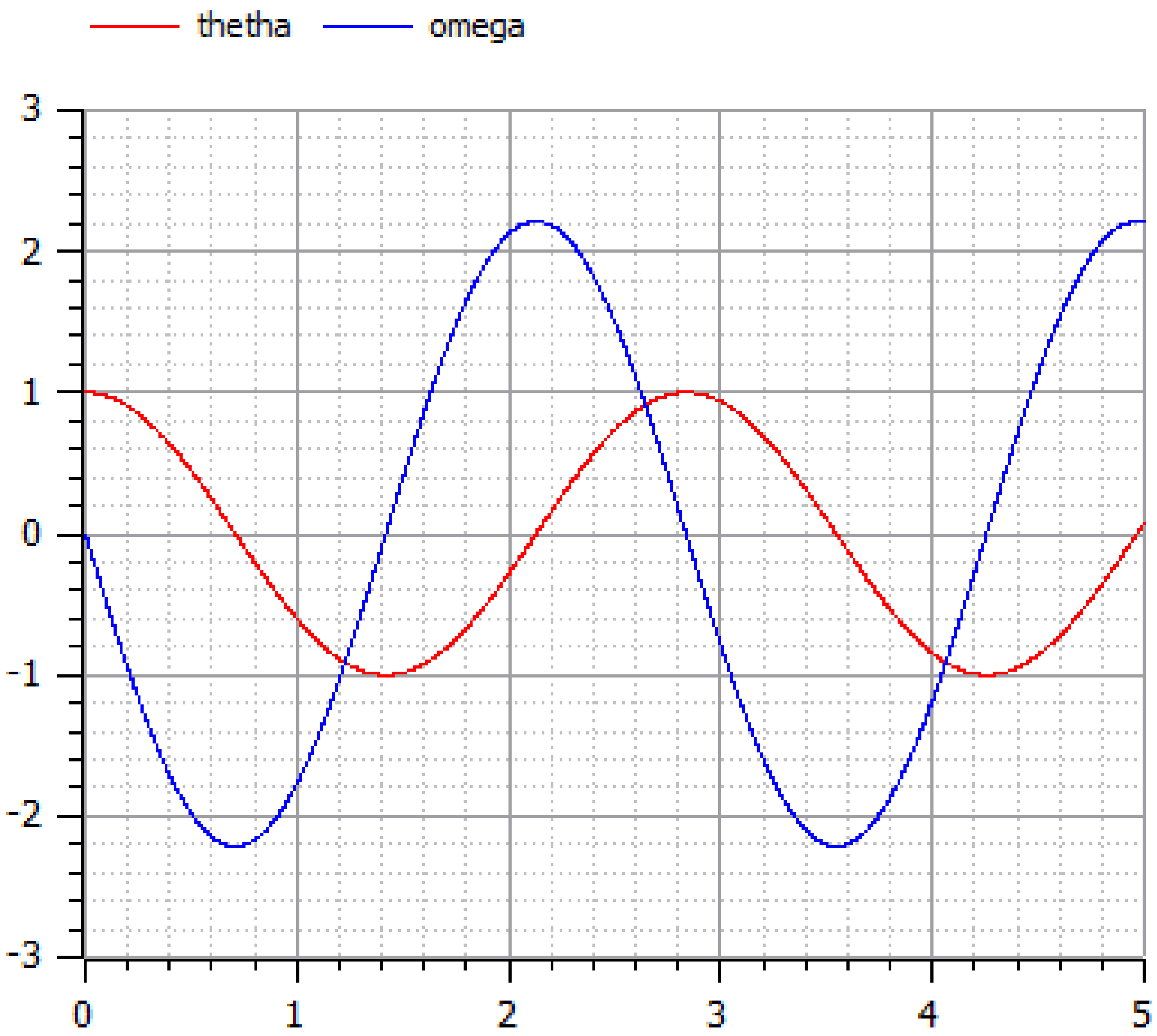
- Perform tests



Simulation of continuous systems

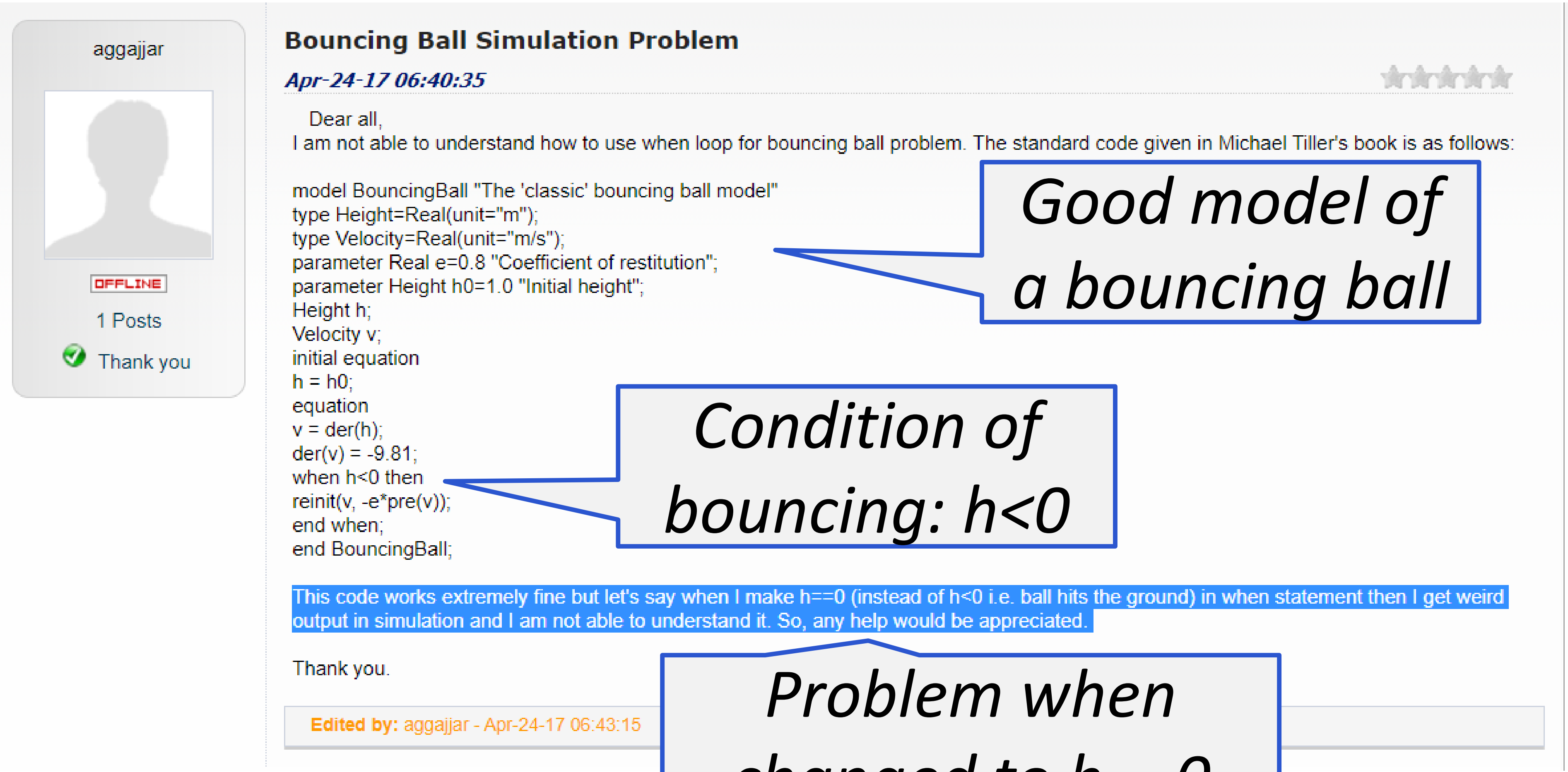
- Simulating a model means to calculate the values of its variables at certain time instants
- Different algorithms and strategies for simulation
  - The task is to solve Ordinary Differential Equations (ODEs) generated from the model
  - Numerical techniques
  - ODE specific solvers exist for this purpose

Example: Pendulum simulation results



Challenge

- Good model can be simulated incorrectly



aggajjar

Bouncing Ball Simulation Problem

Apr-24-17 06:40:35

Dear all,
I am not able to understand how to use when loop for bouncing ball problem. The standard code given in Michael Tiller's book is as follows:

```
model BouncingBall "The 'classic' bouncing ball model"
type Height=Real(unit="m");
type Velocity=Real(unit="m/s");
parameter Real e=0.8 "Coefficient of restitution";
parameter Height h0=1.0 "Initial height";
Height h;
Velocity v;
initial equation
h = h0;
equation
v = der(h);
der(v) = -9.81;
when h<0 then
reinit(v, -e*pre(v));
end when;
end BouncingBall;
```

This code works extremely fine but let's say when I make h==0 (instead of h<0 i.e. ball hits the ground) in when statement then I get weird output in simulation and I am not able to understand it. So, any help would be appreciated.

Thank you.

Edited by: aggajjar - Apr-24-17 06:43:15

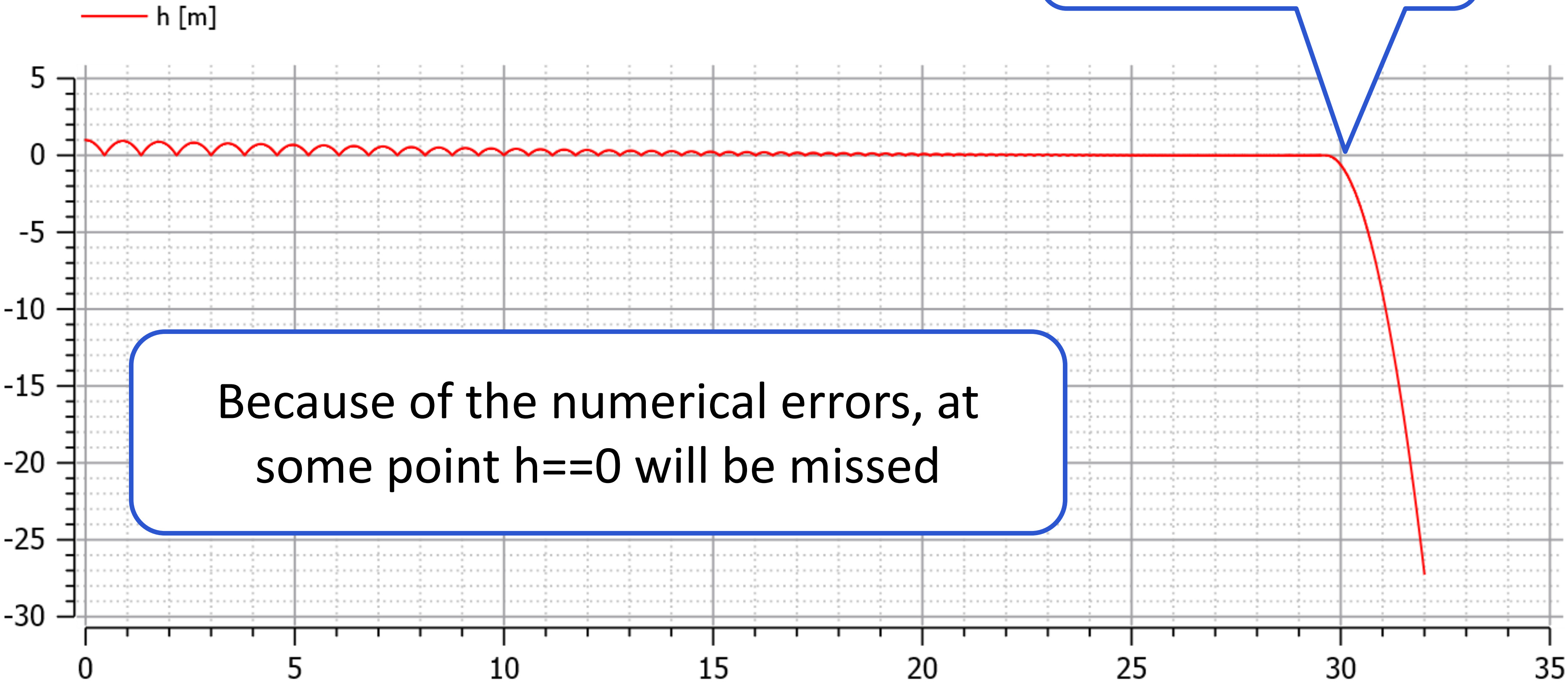
Good model of a bouncing ball

Condition of bouncing: h<0

Problem when changed to h==0

Incorrect simulation example

- Bouncing ball



Ball falls underground?

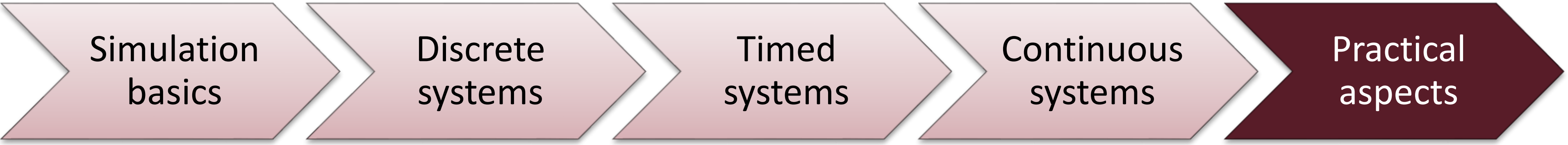
Because of the numerical errors, at some point h==0 will be missed

Continuous simulation challenges

> *„The problem with simulation is that no matter what results you need, you are probably going to be able to get them.”*
>
> [ApPLIED 2019]

- People tend to forget the limitations of simulation
  - Model limitations
    - Complex physics
    - Modeling connections is hard
    - Some environmental impacts will always be neglected
  - Limitations caused by time-stepped simulation
    - Communication delays
    - Induced reactions
  - Differential equation solver limitations
  - Propagation of numerical errors
- Correct configuration requires both domain and simulation knowledge

Simulation



Simulation basics

Discrete systems

Timed systems

Continuous systems

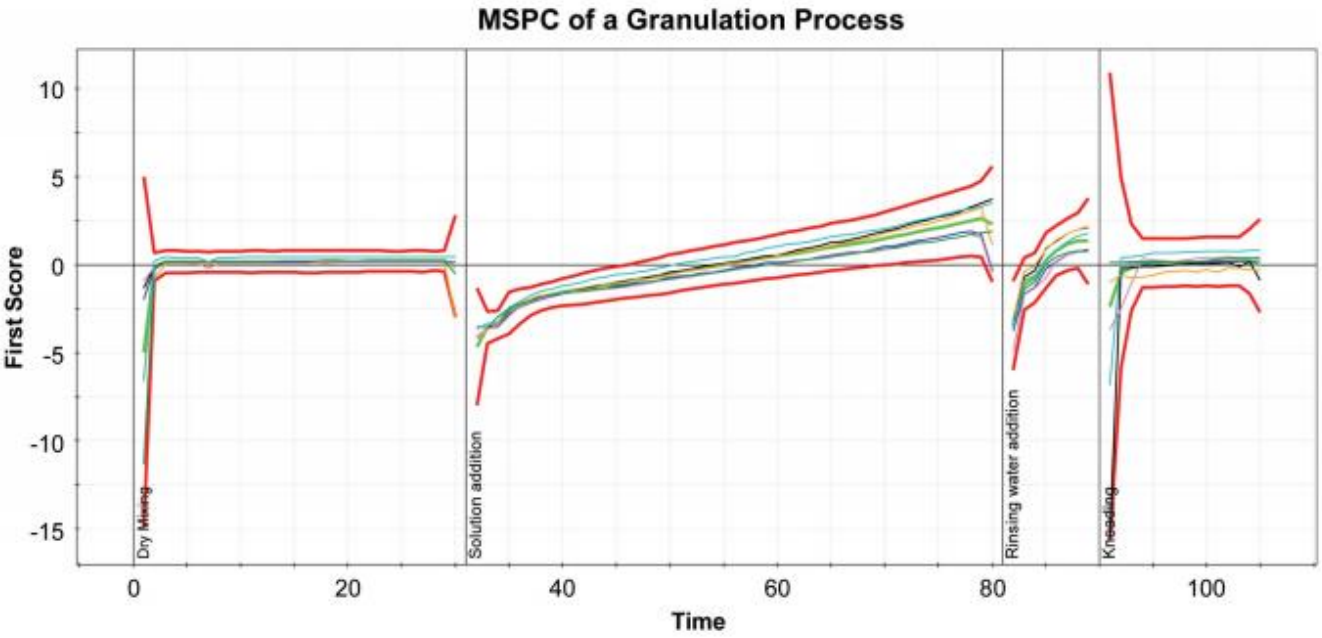
Practical aspects

Physical systems: TLM simulation

- Transmission Line Modeling
  - Every physical transmission/propagation (energy, force, etc.) in the model has a velocity
  - → Delay exists in the real system!
  - Model delay
  - Delay can be used to brake cyclic dependencies
- Most simulation limitations exist to avoid loops
  - Cyclic dependencies between variables
  - There will be a delay
    - Easy solution: always calculate with previous values
    - Better solutions: at least one delay per cycle

Hybrid simulation

- Most industrial models are hybrid
  - Contains both discrete and continuous components
  - Discrete changes effect the dynamic behaviour



Co-simulation

- Models of system components may differ
  - Model domain characteristics
    - E.g. discrete/continuous
    - Desired simulation techniques
  - Modeling environment and capabilities
  - Creator (protection of intellectual property)

Co-simulation: The parallel simulation of different models in a controlled environment, allowing them to communicate, synchronize, etc. without raising IP protection concerns

- Solution: FMI standard

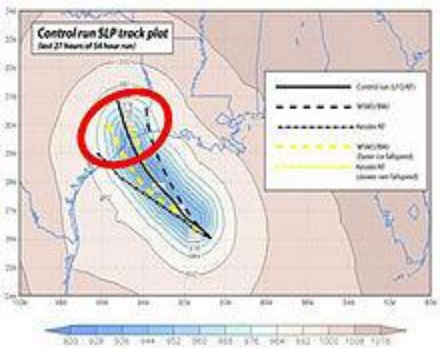
Ensemble simulation

- Problem: uncertainty in initial physical conditions

→No exact initial state

- Solution: Ensemble simulation
  - Repeat simulation multiple times from different states
  - Approximate probabilities of outcomes
- Example: weather
  - Hurricane Rita, 2005 September 07
  - ~ a month after Katarina destroyed New Orleans

  → Most probably heading for Houston



Probabilistic simulation

- Similar to ensemble simulation
  - Run multiple simulations to tackle uncertainty
  - Approximate probabilities based on results
- Difference:
  - Uncertanity not in initial state
  - Random-generator used runtime

→Often used for the sole purpose of probability approximation

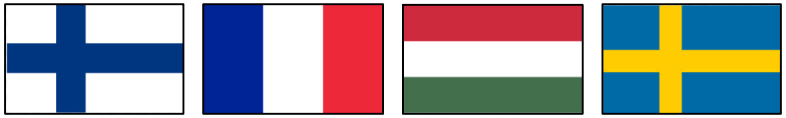
  - Unlike in case of the weather: the most probable outcome is the main interest not the exact probabilities
  - Mathematical computation is often difficult

Motivating case studies

Case studies from the OpenCPS Project

OpenCPS project

- Open Cyber-Physical System Model-Driven Certified Development
- 4 countries



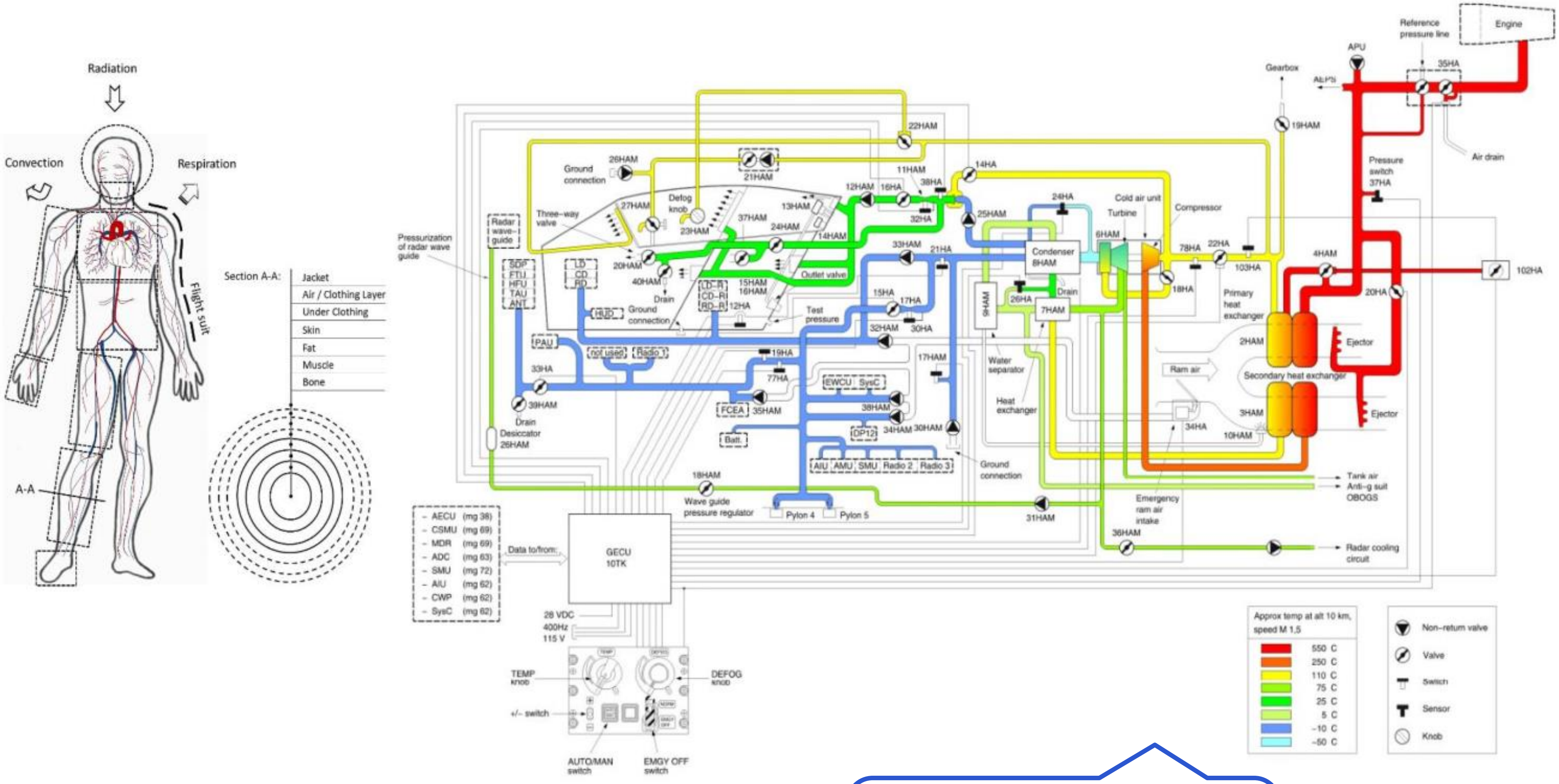
- 18 industrial partners, including

SAAB TECHNOLOGIES SAAB eDF SHERPA ENGINEERING cea ERICSSON SIEMENS

IncQueryLabs Ltd

- Industrial demonstrators in various fields
  - Building, **Aeronautics**, **Mechanics**, Naval, Power plant, Gaz turbines, **Automotive**

Thermal model of an aircraft



Huge, complex system

Copyright: SAAB

Thermal model of an aircraft

Boundary Conditions

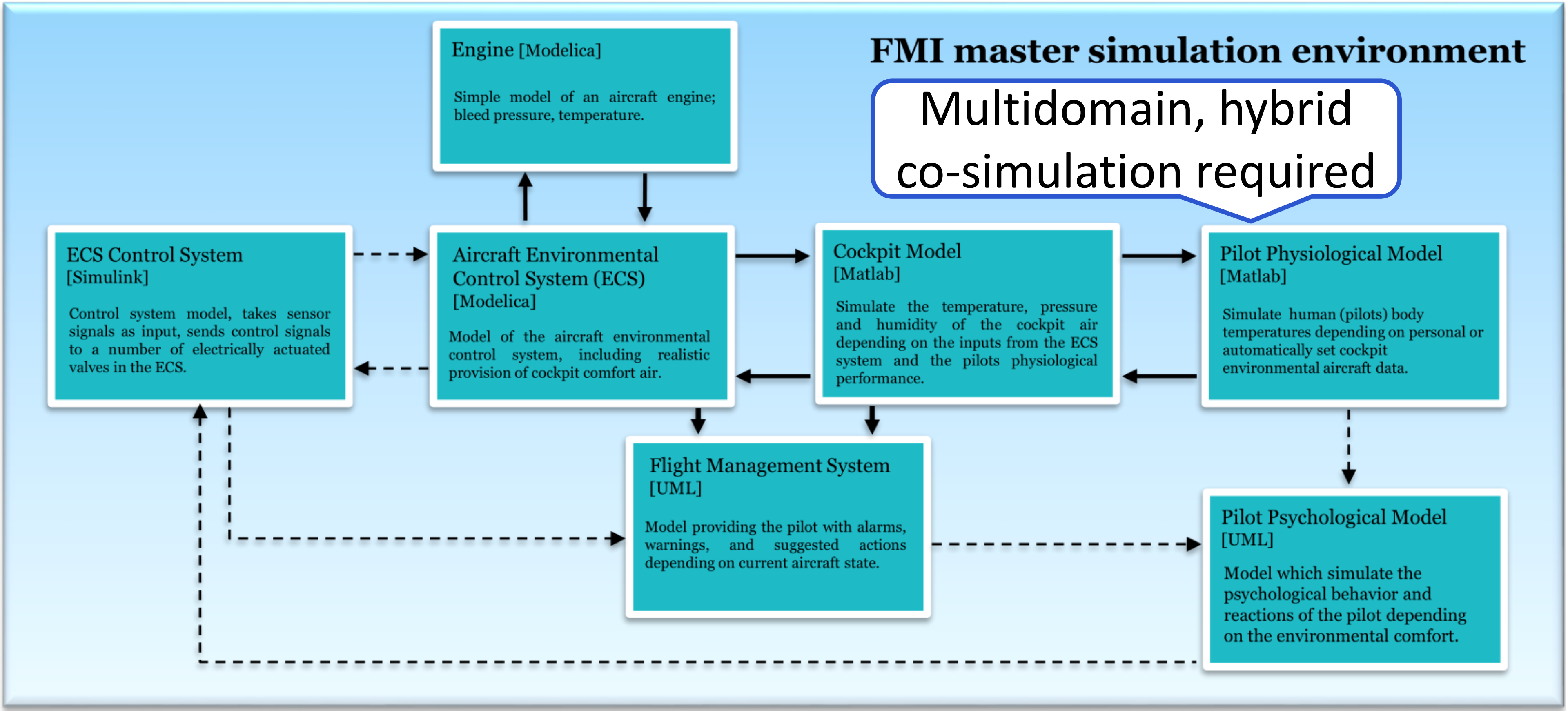
- Flight mission (Mach, altitude, ...)
- Pressure, Temp., Humidity with altitude
- Sun radiation, Sun position,
- Pressure, Temperature, Humidity change over horizontal distance
- Non standard atmospheres model?
- Time varying heat loads from e.g. sensors

Geometry Data

Model description [language/tool origin] — Functional Mock-up Unit (FMU)

→ Physical connection

⇢ Information signal



FMI master simulation environment

Multidomain, hybrid co-simulation required

Engine [Modelica]

Simple model of an aircraft engine; bleed pressure, temperature.

ECS Control System [Simulink]

Control system model, takes sensor signals as input, sends control signals to a number of electrically actuated valves in the ECS.

Aircraft Environmental Control System (ECS) [Modelica]

Model of the aircraft environmental control system, including realistic provision of cockpit comfort air.

Cockpit Model [Matlab]

Simulate the temperature, pressure and humidity of the cockpit air depending on the inputs from the ECS system and the pilots physiological performance.

Pilot Physiological Model [Matlab]

Simulate human (pilots) body temperatures depending on personal or automatically set cockpit environmental aircraft data.

Flight Management System [UML]

Model providing the pilot with alarms, warnings, and suggested actions depending on current aircraft state.

Pilot Psychological Model [UML]

Model which simulate the psychological behavior and reactions of the pilot depending on the environmental comfort.

Copyright: SAAB

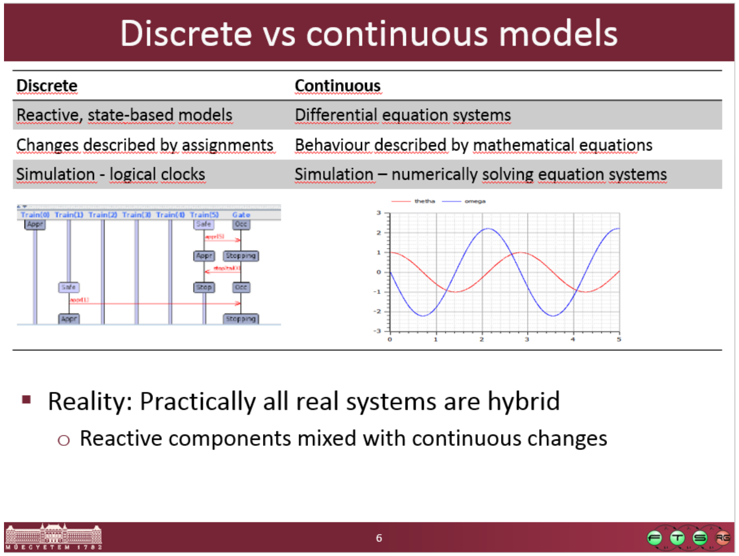
Project experiences, lessons learnt

- Saab’s aircraft model
    - Huge complexity
    - Continuous components required very small simulation step size → highly inefficient simulation
    - Solution: better tools, distributed algorithm
- SKF’s bearing model
    - Required very precise simulation
    - Solution: TLM simulation
- Sherpa’s hybrid electric vehicle model
    - Required fast and accurate simulation
    - (My) solution: New simulation algorithm

DEMO Real world simulation case study

- Nagy Simon és Vajda Máté munkája
- Forgalom-elemző rendszer szimulációja
  - Rendszer: Autók rendszámtáblájának leolvasása + adatok rögzítése/feldolgozása felhőben
  - Szimulációs cél: Késleltetések elemzése az informatikai infrastruktúrában

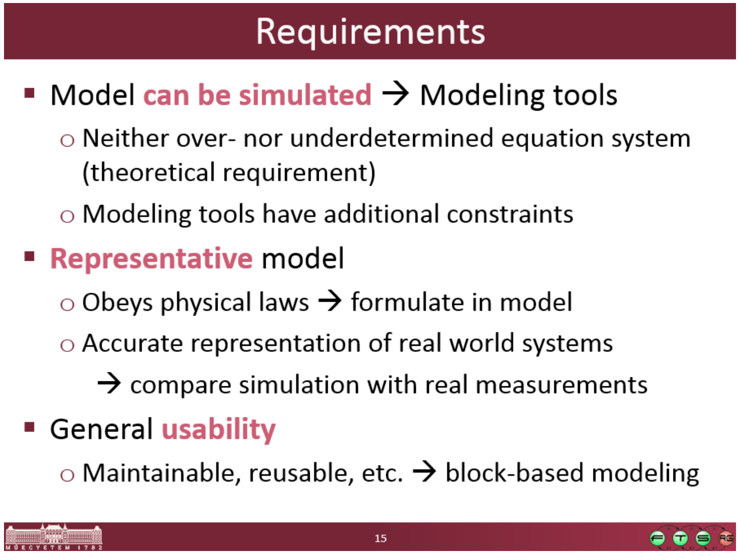
Summary



Discrete vs continuous models

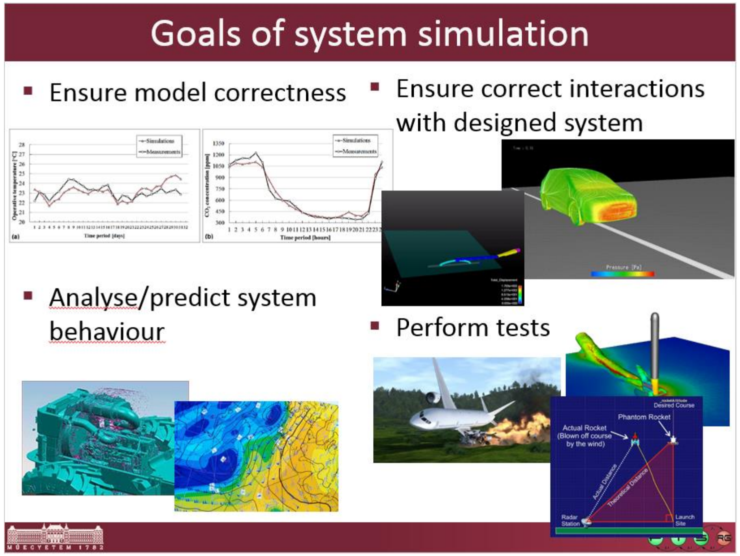
Discrete	Continuous
Reactive, state-based models	Differential equation systems
Changes described by assignments	Behaviour described by mathematical equations
Simulation - logical clocks	Simulation – numerically solving equation systems

- Reality: Practically all real systems are hybrid
    - Reactive components mixed with continuous changes



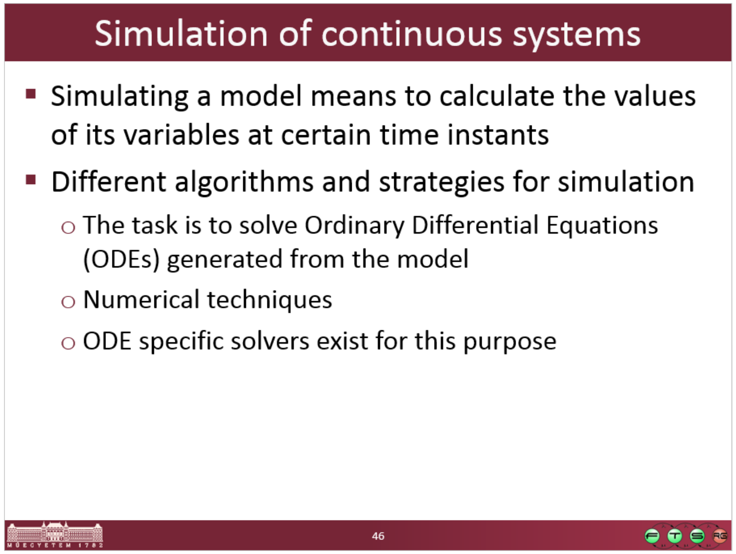
Requirements

- Model **can be simulated** → Modeling tools
    - Neither over- nor underdetermined equation system (theoretical requirement)
    - Modeling tools have additional constraints
- **Representative** model
    - Obeys physical laws → formulate in model
    - Accurate representation of real world systems
        → compare simulation with real measurements
- General **usability**
    - Maintainable, reusable, etc. → block-based modeling



Goals of system simulation

- Ensure model correctness
- Ensure correct interactions with designed system
- Analyse/predict system behaviour
- Perform tests



Simulation of continuous systems

- Simulating a model means to calculate the values of its variables at certain time instants
- Different algorithms and strategies for simulation
    - The task is to solve Ordinary Differential Equations (ODEs) generated from the model
    - Numerical techniques
    - ODE specific solvers exist for this purpose