Overview

1. History of Modelling and Simulation
2. Modelling and Simulation Concepts
3. Levels of Abstraction
4. Experimental Frame
5. Validation
6. Studying a mass-spring system
7. The Modelling and Simulation Process

Modelling and simulation: past

(1950–): Numerical simulations: numerical analysis, statistical analysis, simulation languages (CSSL, discrete-event world views).
focus: performance, accuracy

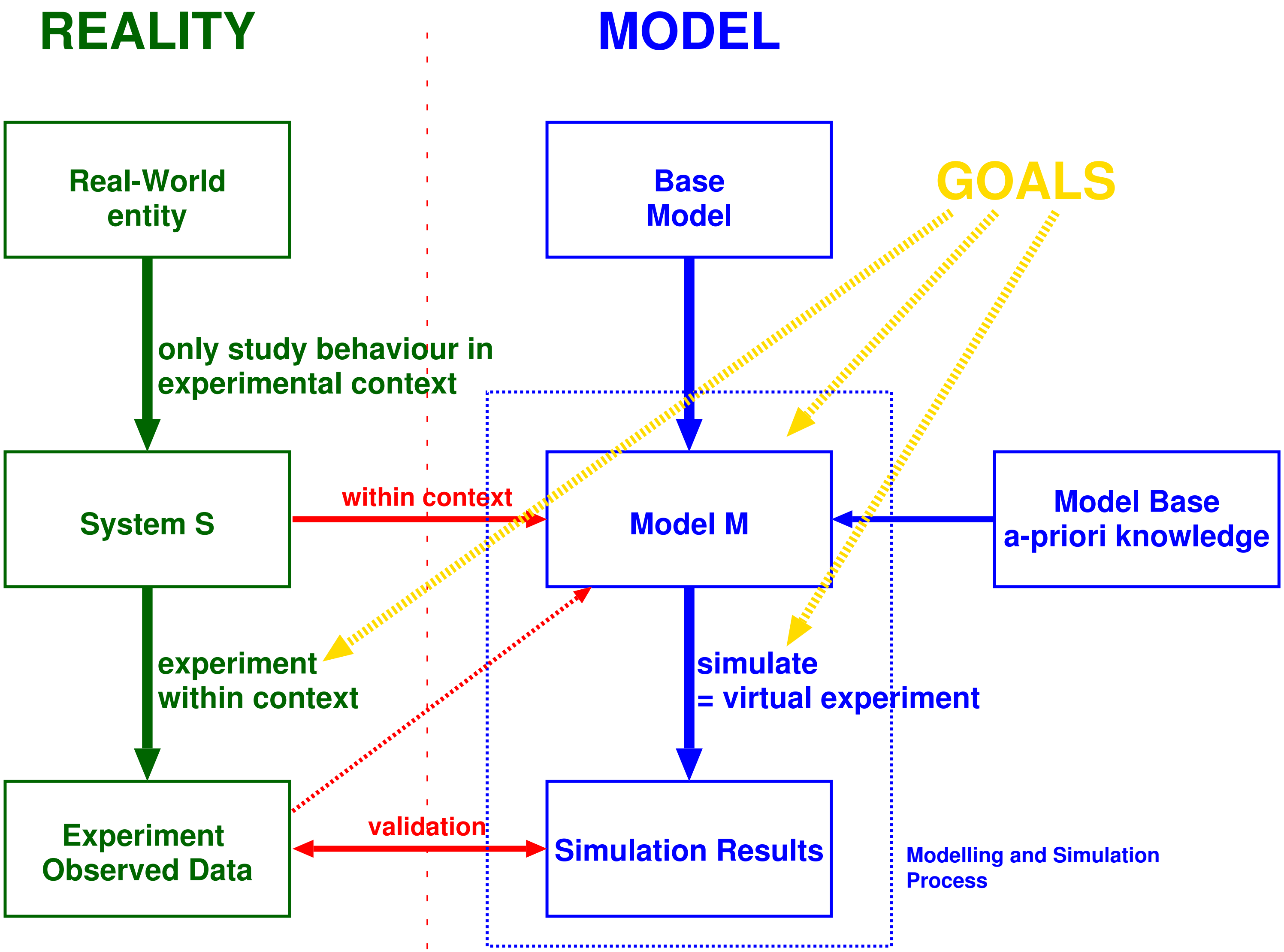
(1981–): Artificial Intelligence: model = knowledge representation
Use AI techniques in modelling, AI uses simulation (“deep” knowledge)
focus: knowledge

(1988–): Object-oriented modelling and simulation
focus: object orientation, later “agents”, non-causal modelling

Modelling and simulation: past, present, future

(1993–): Multi-formalism, Multi-paradigm (2001 –)

1. Do it right (optimally) the first time (market pressure)
2. Complex systems: **multi-formalism**
3. Hybrid: continuous-discrete, hardware/software
4. **Exchange** (between humans/tools) and **re-use** (validated model)
5. User focus: do not expect user to know details
   (software: glueing of components), need for **tools**



REALITY

MODEL

Real-World entity

only study behaviour in experimental context

System S

experiment within context

Experiment Observed Data

Base Model

GOALS

within context

Model M

Model Base a-priori knowledge

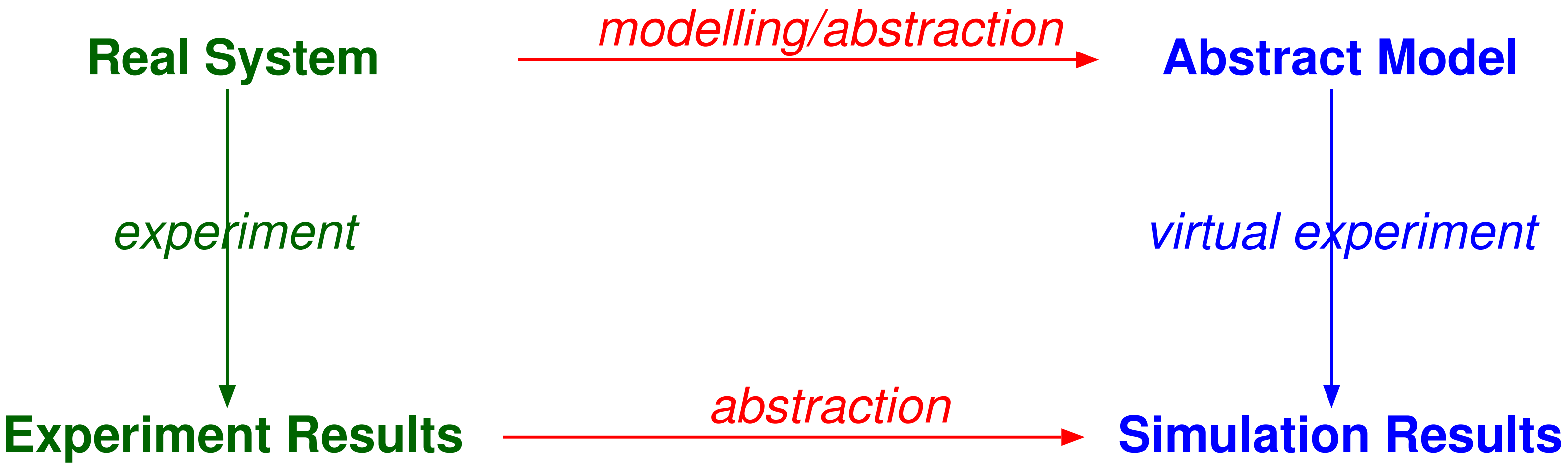
simulate = virtual experiment

validation

Simulation Results

Modelling and Simulation Process

Behaviour Morphism



Real System

modelling/abstraction

Abstract Model

experiment

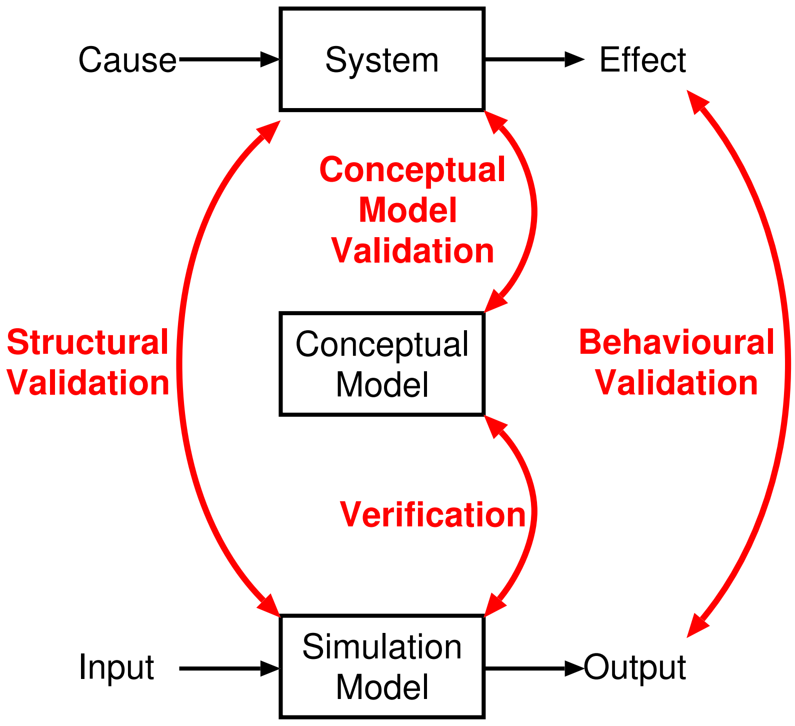
virtual experiment

Experiment Results

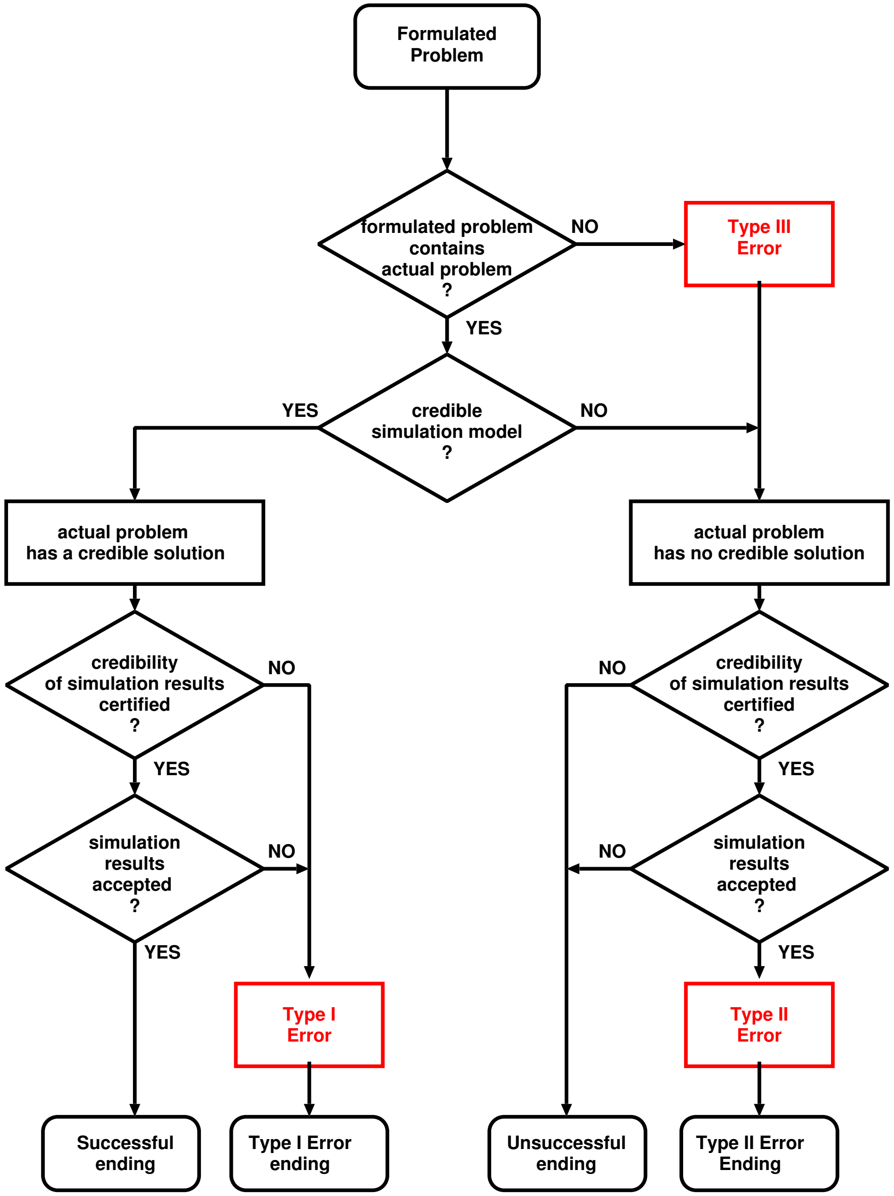
abstraction

Simulation Results

Verification and Validation



Popper: Falsification, Confidence



Formulated Problem

formulated problem contains actual problem ?

NO

Type III Error

YES

credible simulation model ?

YES

NO

actual problem has a credible solution

actual problem has no credible solution

credibility of simulation results certified ?

NO

YES

simulation results accepted ?

NO

YES

Type I Error

credibility of simulation results certified ?

NO

YES

simulation results accepted ?

NO

YES

Type II Error

Successful ending

Type I Error ending

Unsuccessful ending

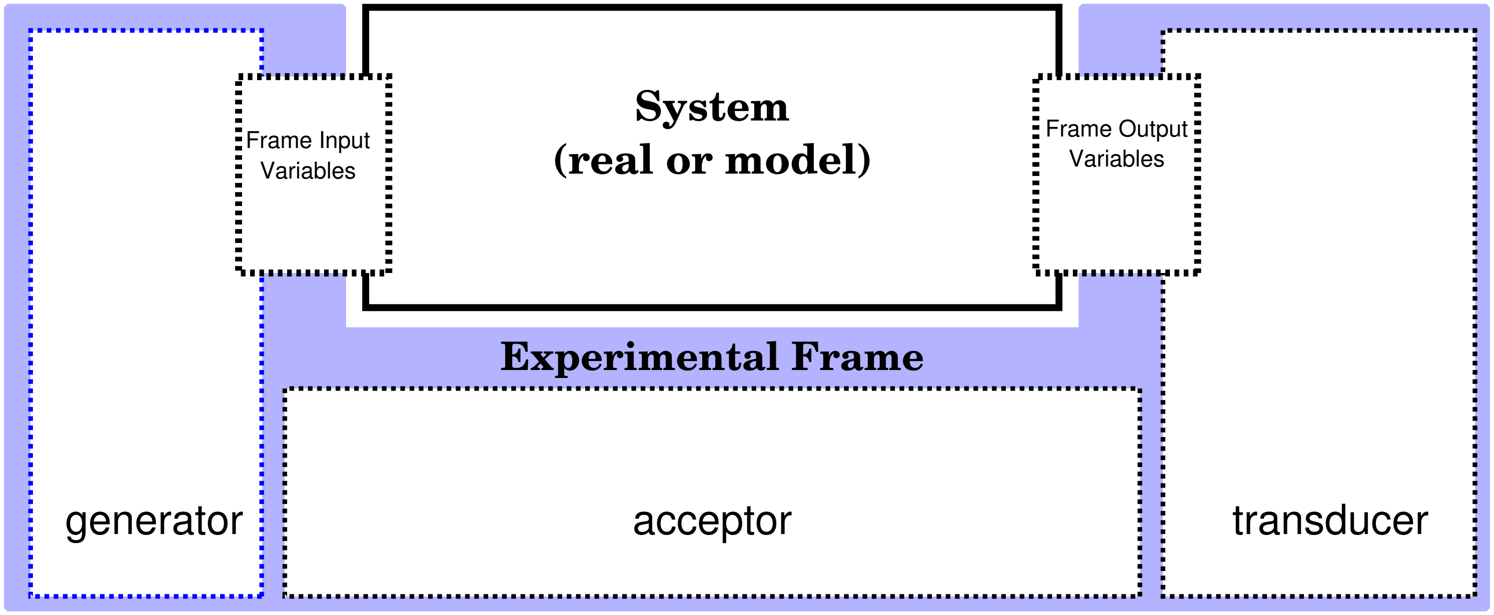
Type II Error Ending

System, Base Model, Lumped Model

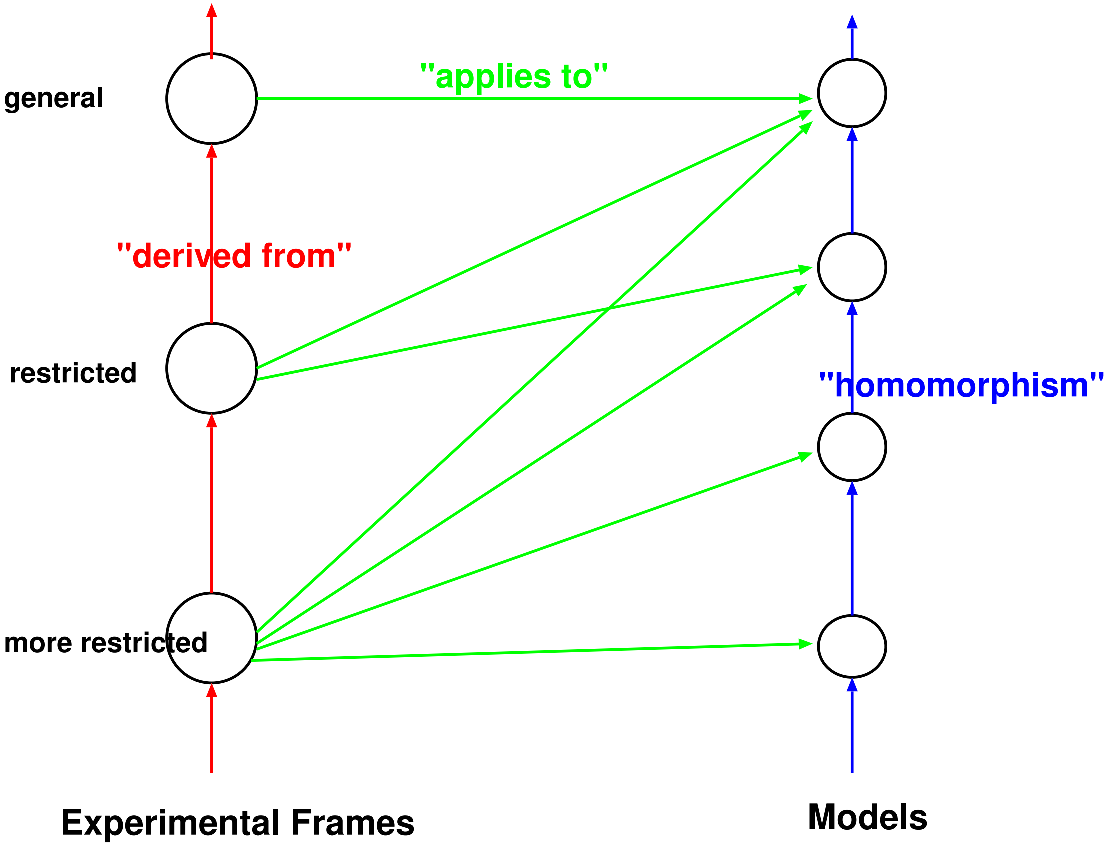
$$D_{BaseModel} \equiv D_{RealSystem}$$

$$D_{LumpedModel} || E \equiv D_{RealSystem} || E$$

Experimental Frame Structure



∼ Programming Language Types


"applies to"
general
"derived from"
restricted
"homomorphism"
more restricted
Experimental Frames
Models

Experimental Frame and Validity

Replicative Validity ($\equiv$: accuracy):

$$D_{LumpedModel} || E \equiv D_{BaseModel} || E$$

Predictive Validity:

$$F_{LumpedModel} || E \subseteq F_{BaseModel} || E$$

Structural Validity (morphism $\overset{\triangle}{=}$):

$$LumpedModel || E \overset{\triangle}{=} BaseModel || E$$

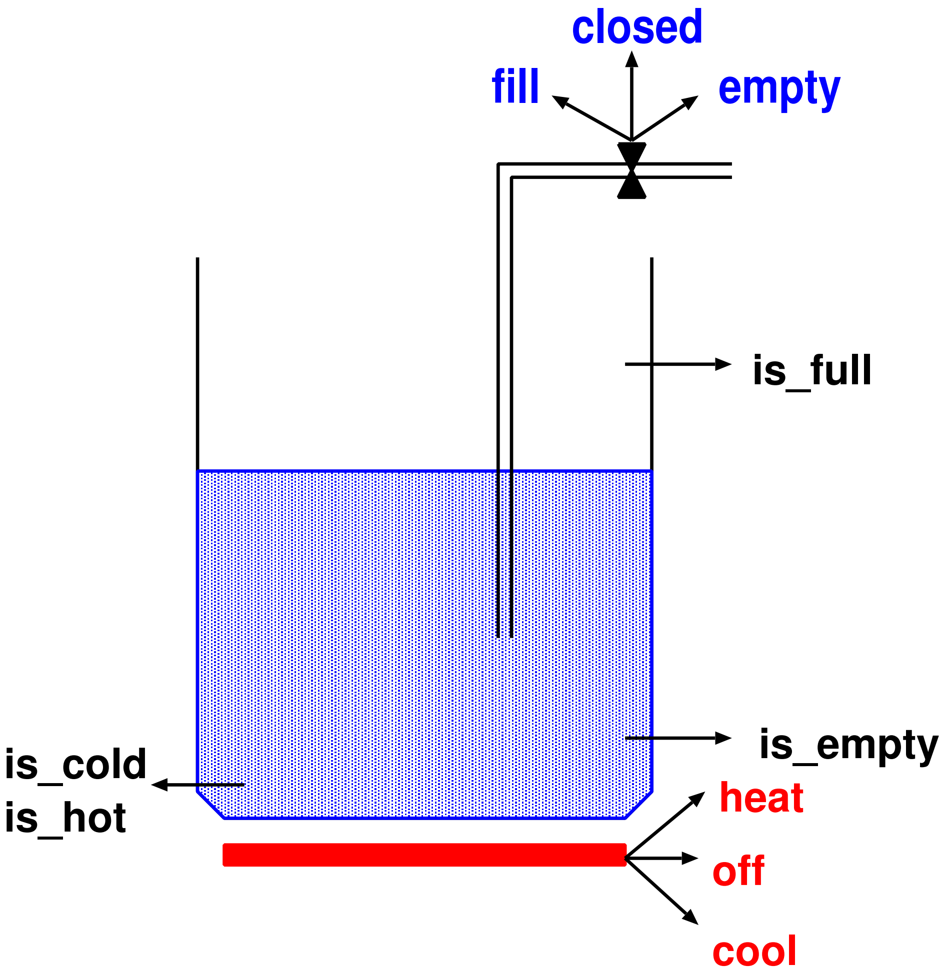
Simulator:

$$D_{Simulator} \equiv D_{LumpedModel}$$

Modelling (and Simulation) Choices

1. System Boundaries and Constraints: Experimental Frame (EF)
2. Level of Abstraction
3. Formalism(s)
4. Level of Accuracy

System under study: T, l controlled liquid



System Boundaries (Experimental Frame)

- Inputs: liquid flow rate, heating/cooling rate
- Outputs: observed level, temperature
- Contraints: no overflow/underflow, one phase only (no boiling)

Abstraction: detailed (continuous) view, ALG + ODE formalism

Inputs (discontinuous → hybrid model):

- Emptying, filling flow rate ϕ
- Rate of adding/removing heat W

Parameters:

- Cross-section surface of vessel A
- Specific heat of liquid c
- Density of liquid ρ

State variables:

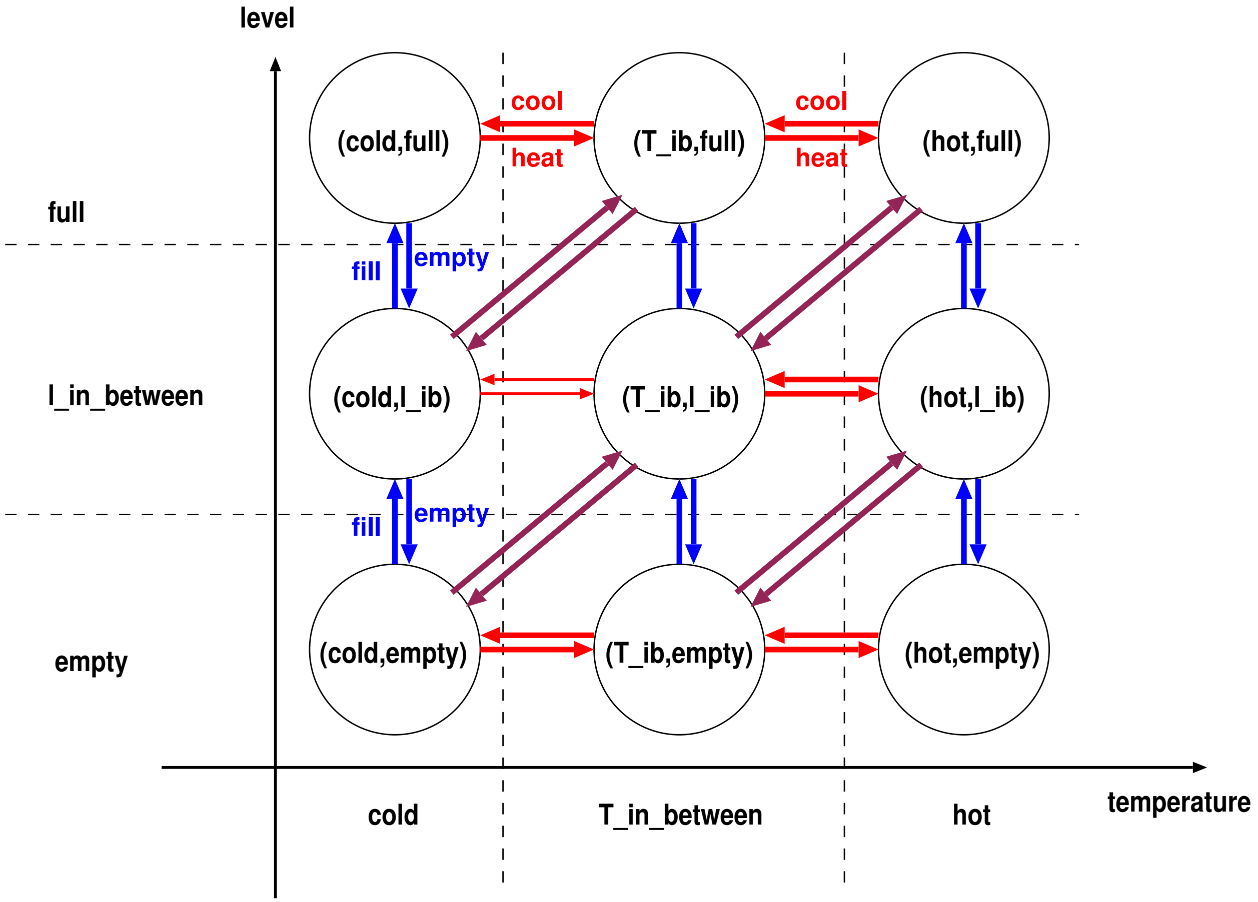
- Temperature T
- Level of liquid l

Outputs (sensors):

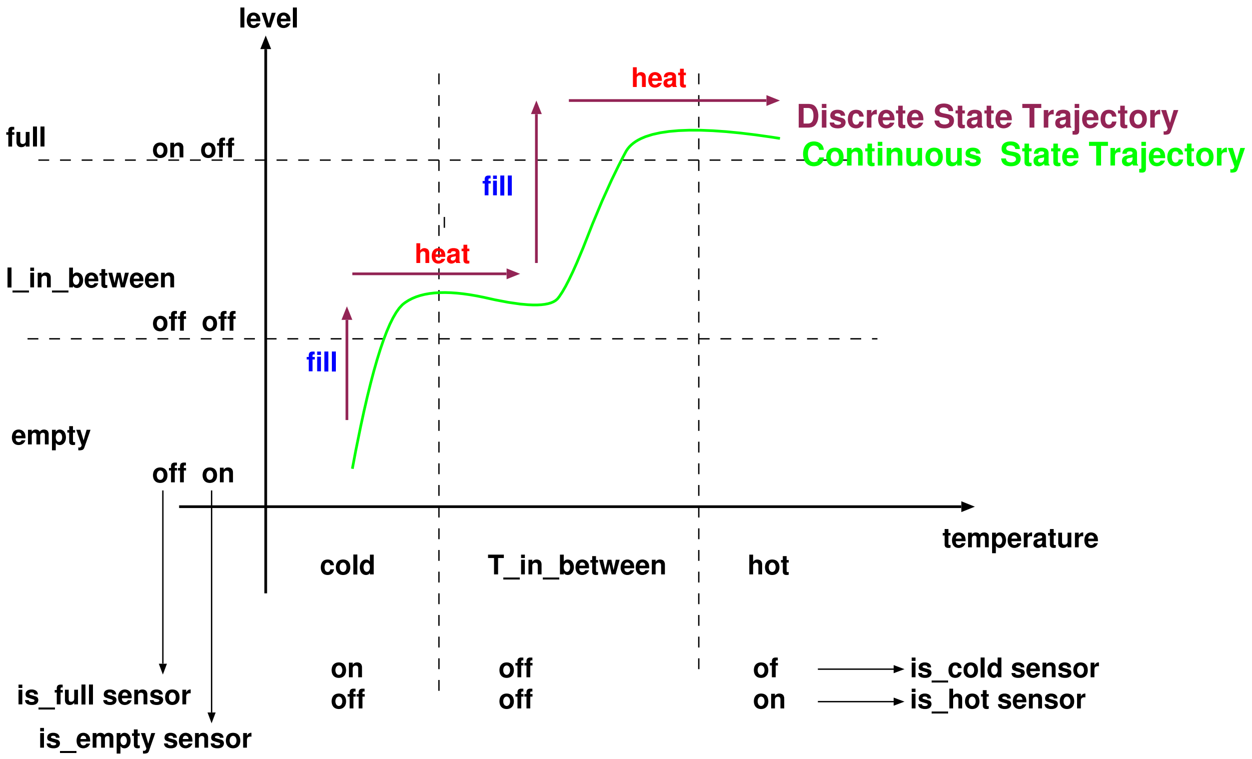
- $is\_low, is\_high, is\_cold, is\_hot$

$$\left\{ \begin{array}{rcl} \frac{dT}{dt} & = & \frac{1}{l}\left[\frac{W}{c\rho A} - \phi T\right] \\ \frac{dl}{dt} & = & \phi \\ is\_low & = & (l < l_{low}) \\ is\_high & = & (l > l_{high}) \\ is\_cold & = & (T < T_{cold}) \\ is\_hot & = & (T > T_{hot}) \end{array} \right.$$

Abstraction: high-level (discrete) view, FSA formalism



Levels of abstraction: trajectories (behaviour)



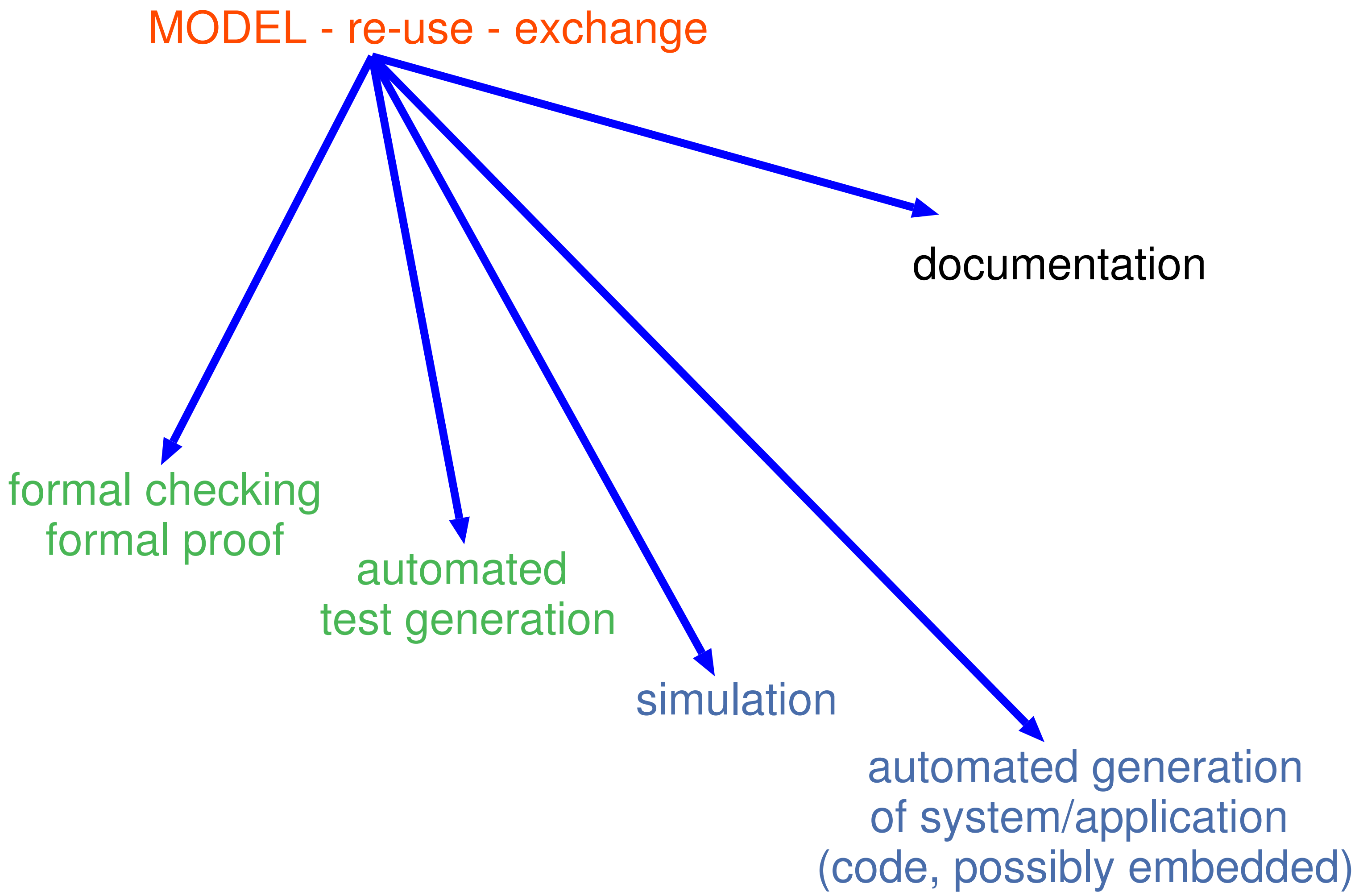
different level of accuracy

Levels of abstraction: behaviour morphism

$$\begin{tikzcd}
\text{model} & M\_t \arrow[r, "\text{abstraction}"'] \arrow[d, "\text{simulation}"'] & M\_d \arrow[d] \\
\text{trajectory} & traj\_t \arrow[r] & traj\_d
\end{tikzcd}$$

detailed
(technical) level

abstract
(decision) level



MODEL - re-use - exchange

documentation

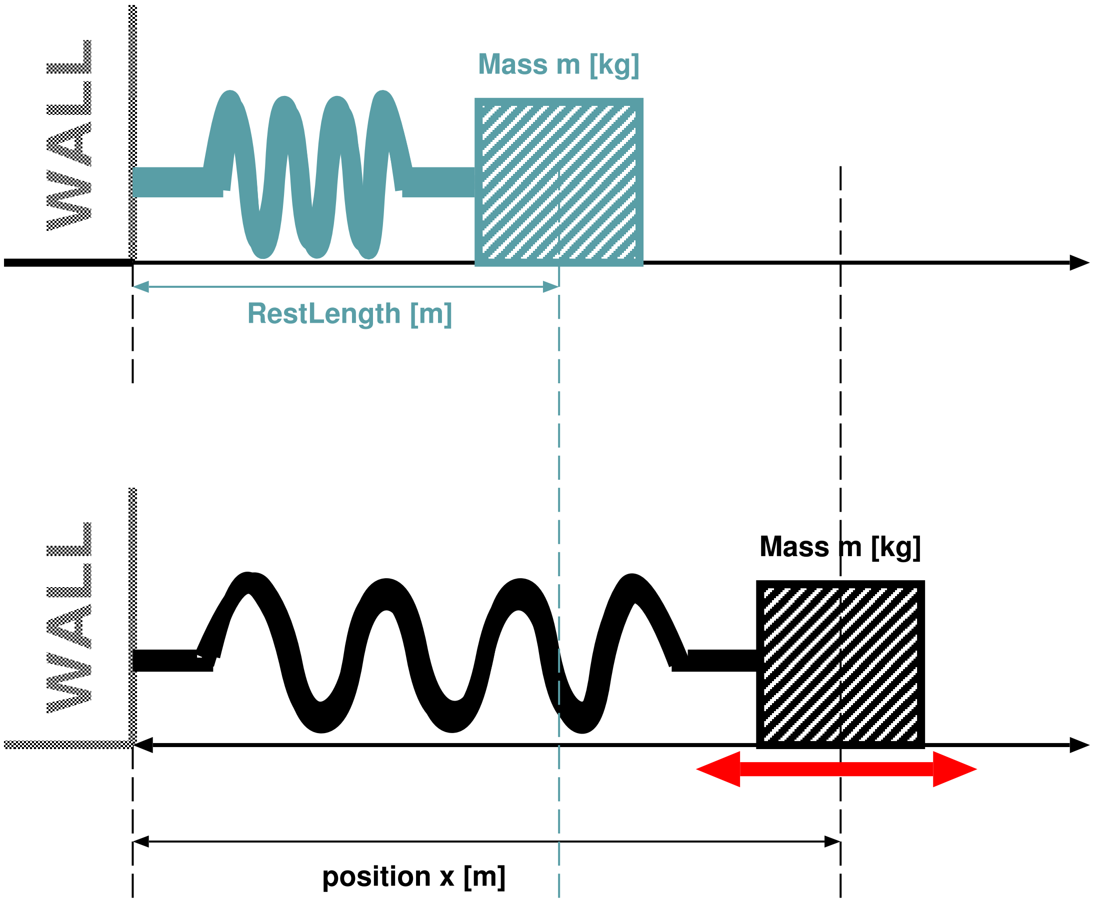
formal checking
formal proof

automated
test generation

simulation

automated generation
of system/application
(code, possibly embedded)

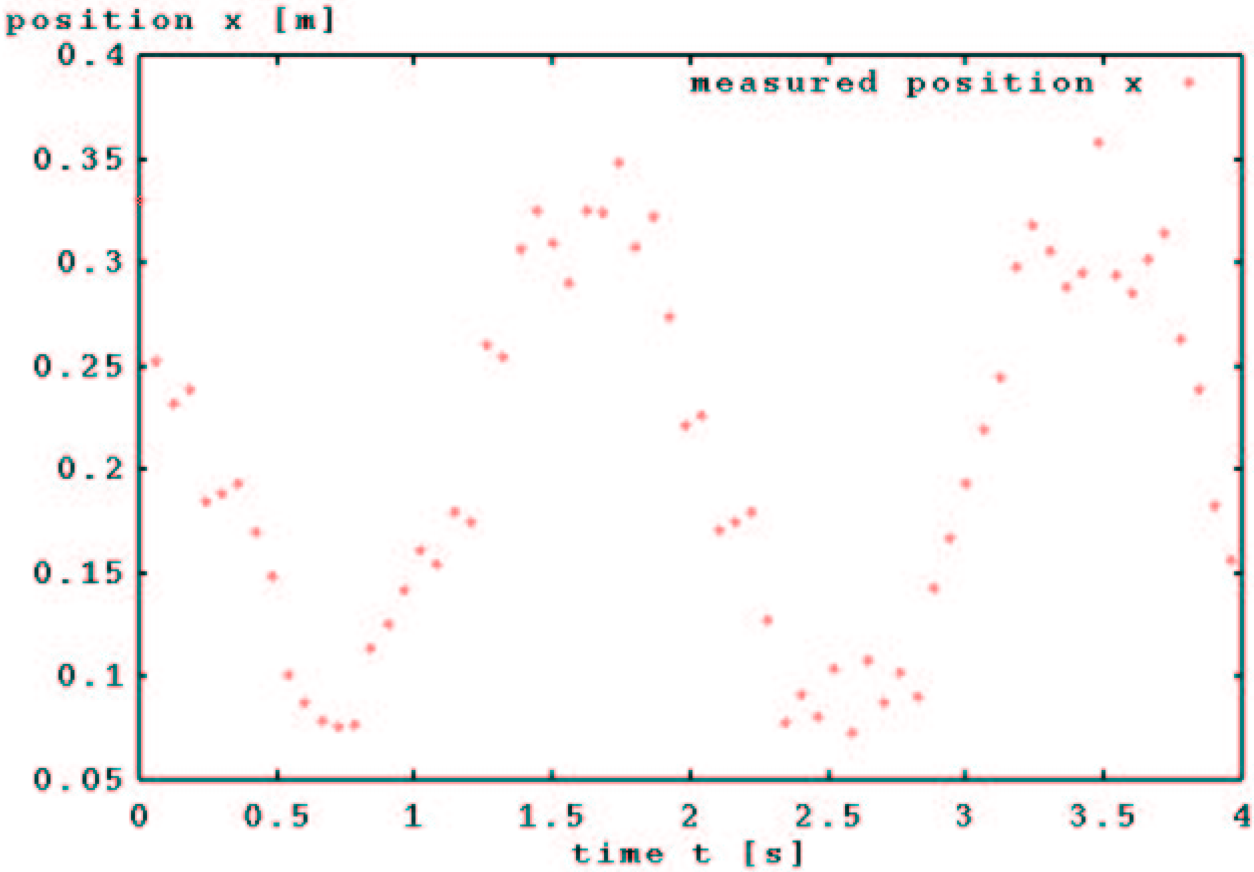
A Modelling and Simulation Exercise: the Mass-Spring system



Knowledge Sources

- A Priori Knowledge: Laws of Physics
- Goals, Intentions: Predict trajectory given Initial Conditions, “optimise” behaviour, . . .
    1. Analysis
    2. Design
    3. Control
- Measurement Data

Measured Data



Experimental Frame

- Room Temperature, Humidity, ...
- Frictionless, Ideal Spring, ...

Structure Characterisation

- $n-1$-order polynomial will perfectly fit n data points
- Ideal Spring: Feature = maximum amplitude constant
- Spring with Damping: Feature = amplitute decreases

Building the model from a-priori knowledge

Newton's Law

$$F = M\frac{d^2\Delta x}{dt}$$

Ideal Spring

$$F = -K\Delta x$$

$$\downarrow$$

$$\frac{d^2\Delta x}{dt^2} = -\frac{K}{M}\Delta x$$

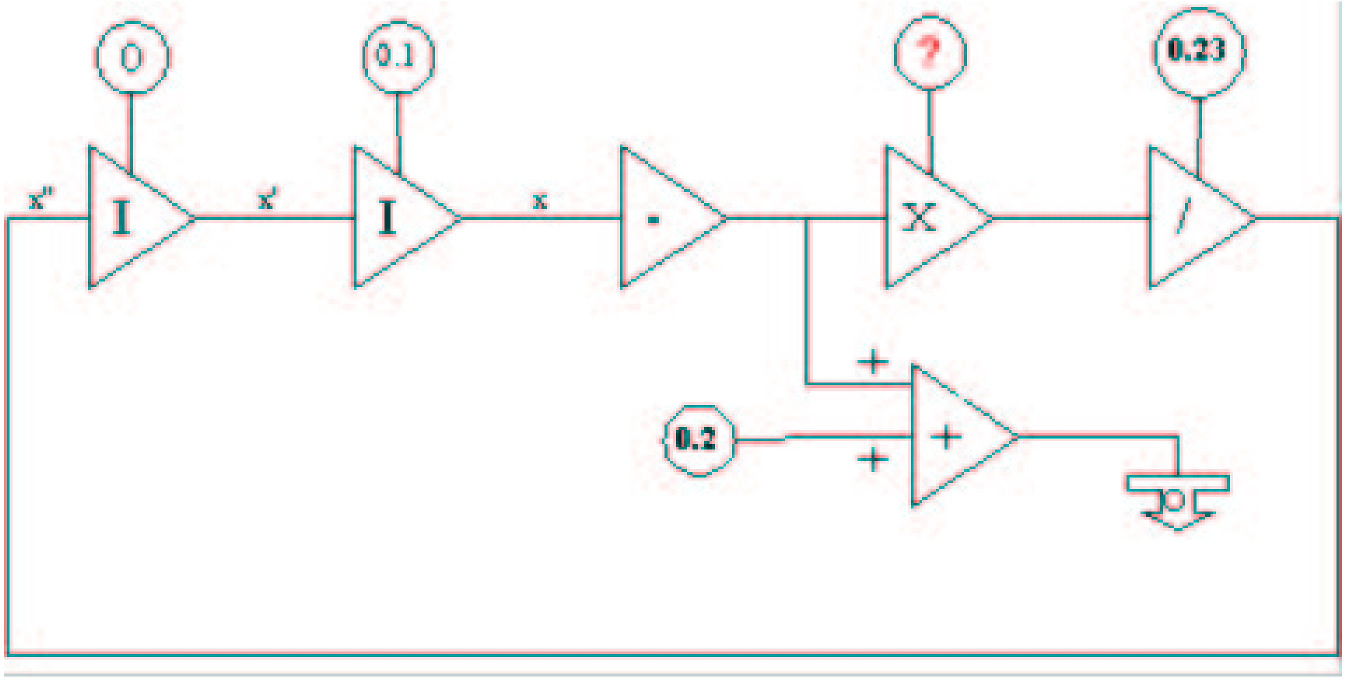
```
CLASS Spring "Ideal Spring": DAEmodel :=
{
 OBJ F_left: ForceTerminal,
 OBJ F_right: ForceTerminal,

 OBJ RestLength: LengthParameter,
 OBJ SpringConstant: SCParameter,

 OBJ x: LengthState,
 OBJ v: SpeedState,

 F_left - F_right = - SpringConstant * (x - RestLength),
 DERIV([ x, [t,] ]) = v,

 EF_assert( x - RestLenght < RestLength/100),
},
```

From Model to Simulation

Block-diagrams

analog computers, Continuous System Modelling Program (CSMP)

- From (algebraic) equation to Block Diagram
- Higher order differential equations



Time-slicing simulator pseudo-code

```
Main program:

 FOREACH block IN system
  IF block is integrator
   initialise block's output with initial condition (IC)
  ELSE
   initialise output with 0

 READ system network (graph) structure

 READ integrator configuration info
  step_size
  communication_interval

 READ experiment info
  end_time
```

Time-slicing simulator pseudo-code (ctd.)

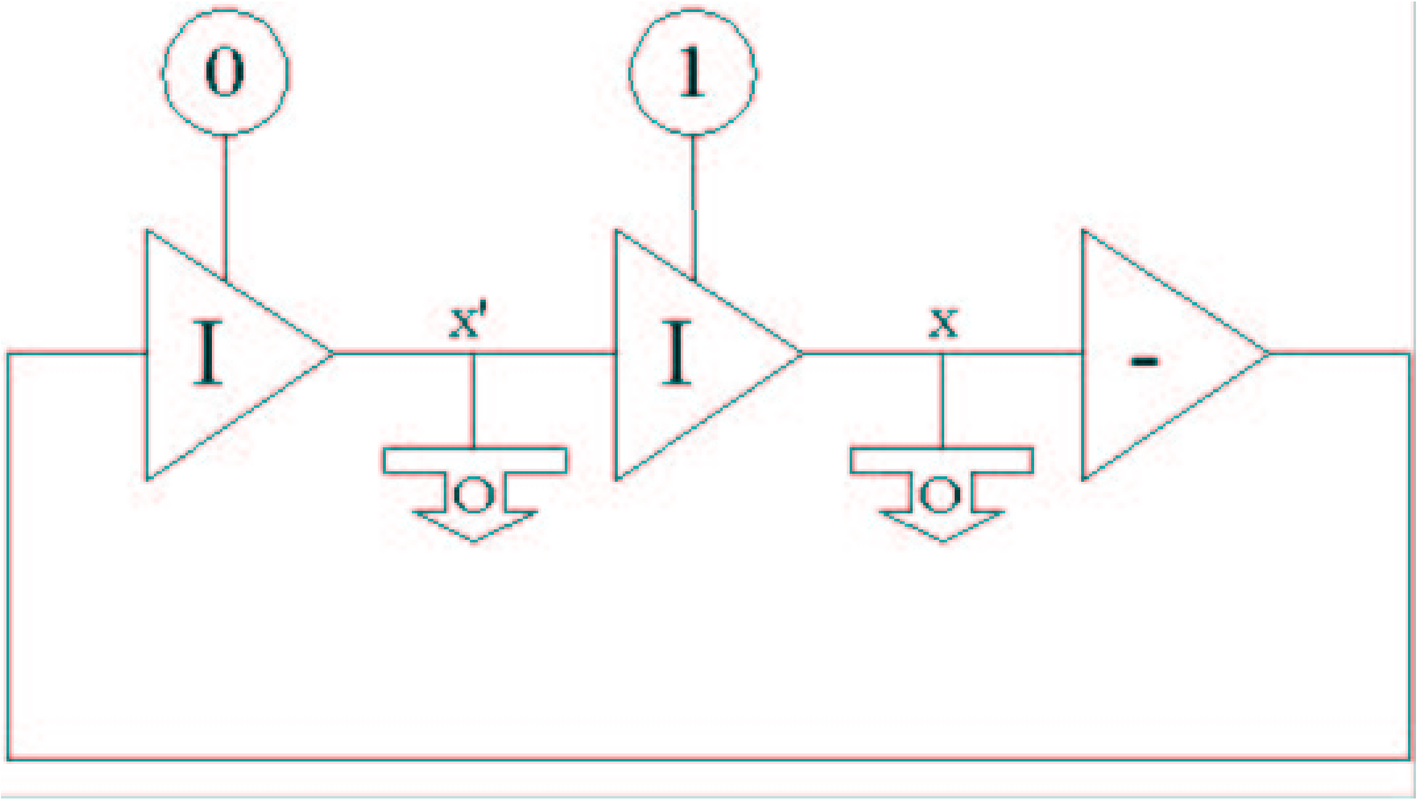
```
Simulation Kernel Loop:
 WHILE (NOT End_of_simulation) DO
  Update_blocks
  Output time and state variables

Update_blocks:
 FOREACH block IN system
  given current block inputs
  (get input from output of influencer)
  Calculate_block_output for block
 increment current_time with stepsize

End_of_simulation:
 termination condition such as
  current_time >= end_time
  condition(state_values) == TRUE
```

```
Calculate_block_output: ([...] means optional)

  WeightedSum
    W, block_number, P1, e1[, P2, e2[, P3, e3]] ; --->
       output= SUM_i(Pi*ei)
  Summer
    +, block_number, P1, e1[, P2, e2[, P3, e3]] ; --->
       output = SUM_i(sign(Pi)*ei)
       (only the sign of Pi is used)
  Integrator
    I, block_number, IC, e1 ; --->
       output= previous_output +  step_size*e1
       (simple fixed-step Euler integration)
  Minus (Sign Inverter)
    -, block_number, e1 ; --->
       output= -e1.
  Multiplier
    X, block_number, e1, e2 ; --->
       output= e1*e2.
  Divider
    /, block_number, e1, e2 ; --->
       output= e1/e2.
  Constant
    K, block_number, P1 ; --->
       output= P1.
  Output
    O, block_number, e1 ; --->
       output= e1.
       (As a side-effect, the (time, e1) tuple is put
        on the output stream at every communication point).
```

Time Slicing: Evaluation Order

Blocks need to be sorted !

Time Slicing: Circle Test



```
; Circle Test for
; CSMP-style Time Slicing Simulator

$endtime = 100;
$timestep = ?;
$comminterval = 1.5;

I, 1, 0, 3        ; x’ is integral of x’’, IC=0
I, 2, 1, 1        ; x  is integral of x’,  IC=1
-, 3, 2           ; -x
O, 4, 1           ; output x’
O, 5, 2           ; output x
```

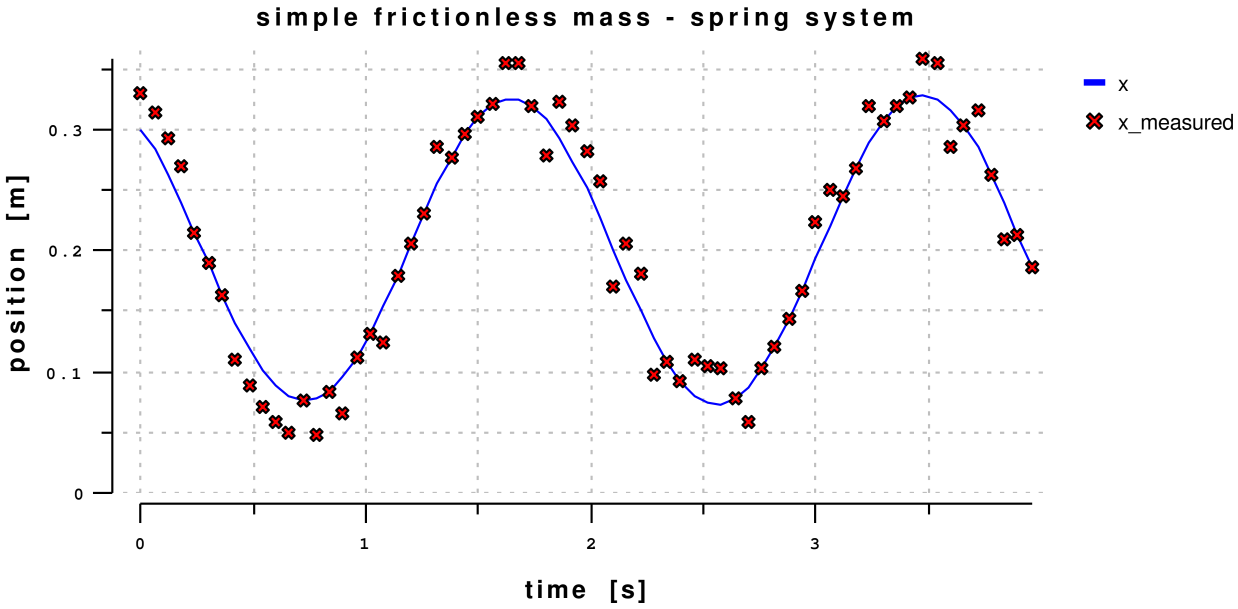
Experimentation

1. Model
2. Parameters (constant for each simulation run)
3. Initial Conditions
4. Input (file, interactive, real system)
5. Output (file, plot, real system)
6. Solver Configuration
7. Experiment type (simulation, optimization, parameter estimation $=$ model calibration)

Results Analysis

- Accuracy (numerical)
- Model Parameters
- Model Structure

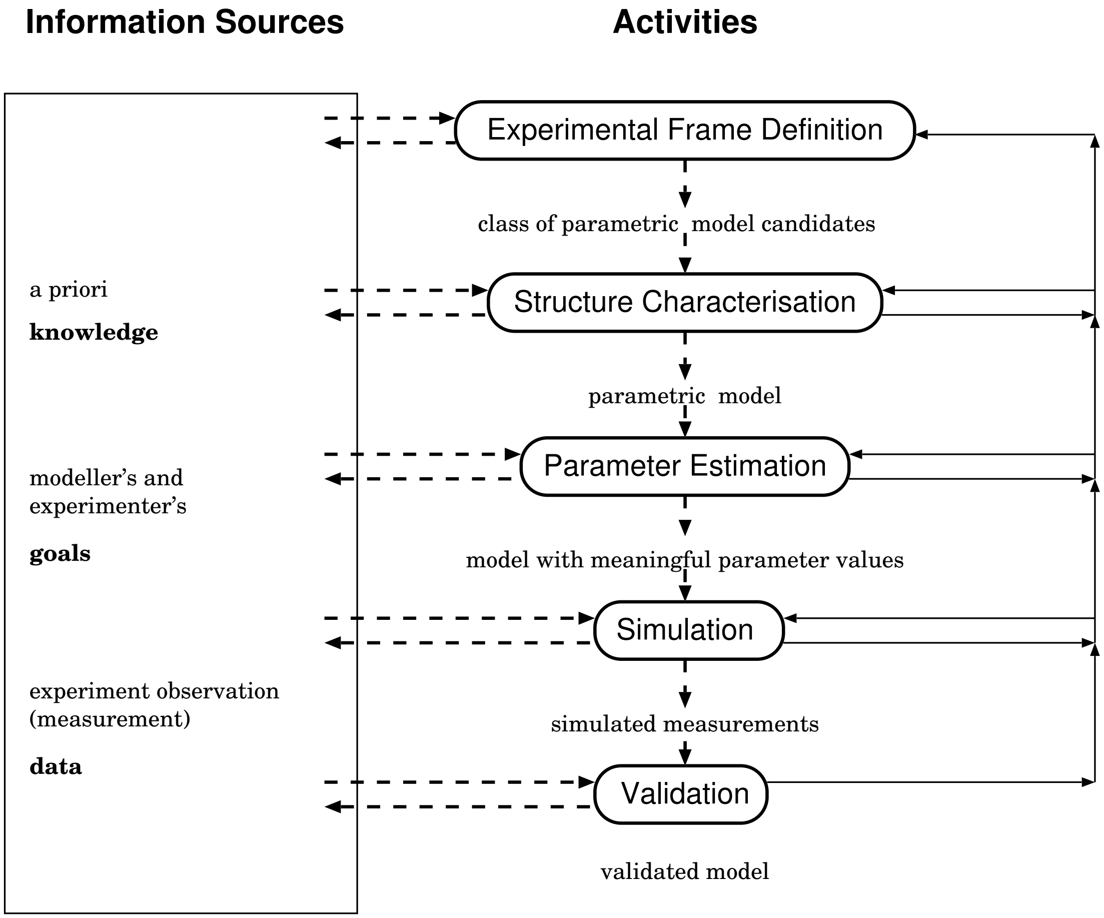
Model Calibration: Parameter Fit



From Here On . . .

- Virtual Experiments: simulation, optimisation, what-if, . . .
- Validation/Falsification

Modelling and Simulation *Process*



Information Sources

Activities

a priori

knowledge

modeller's and
experimenter's

goals

experiment observation
(measurement)

data

Experimental Frame Definition

class of parametric model candidates

Structure Characterisation

parametric model

Parameter Estimation

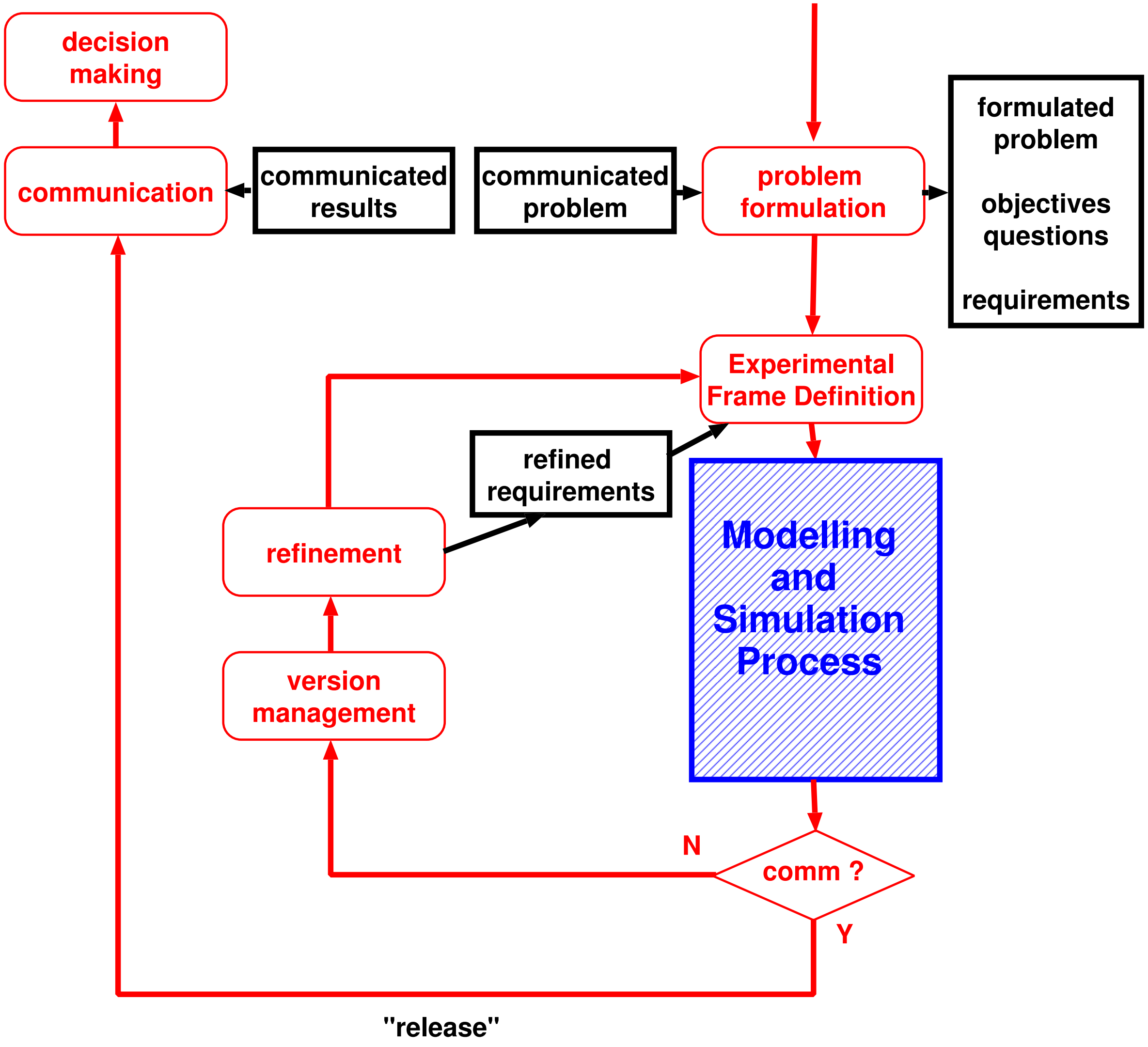
model with meaningful parameter values

Simulation

simulated measurements

Validation

validated model


decision making
communication
communicated results
communicated problem
problem formulation
formulated problem
objectives
questions
requirements
Experimental Frame Definition
refined requirements
refinement
Modelling and Simulation Process
version management
N
comm ?
Y
"release"