Co-simulation of Distributed Embedded Real-time Control Systems

Marcel Verhoef [1] **Peter Visser** [2] **Jozef Hooman** [3] **Jan Broenink** [2]

[1] Chess and Radboud University Nijmegen
[2] University of Twente, Dept EE-M-CS, Control Engineering Group
[3] Embedded Systems Institute and Radboud University Nijmegen

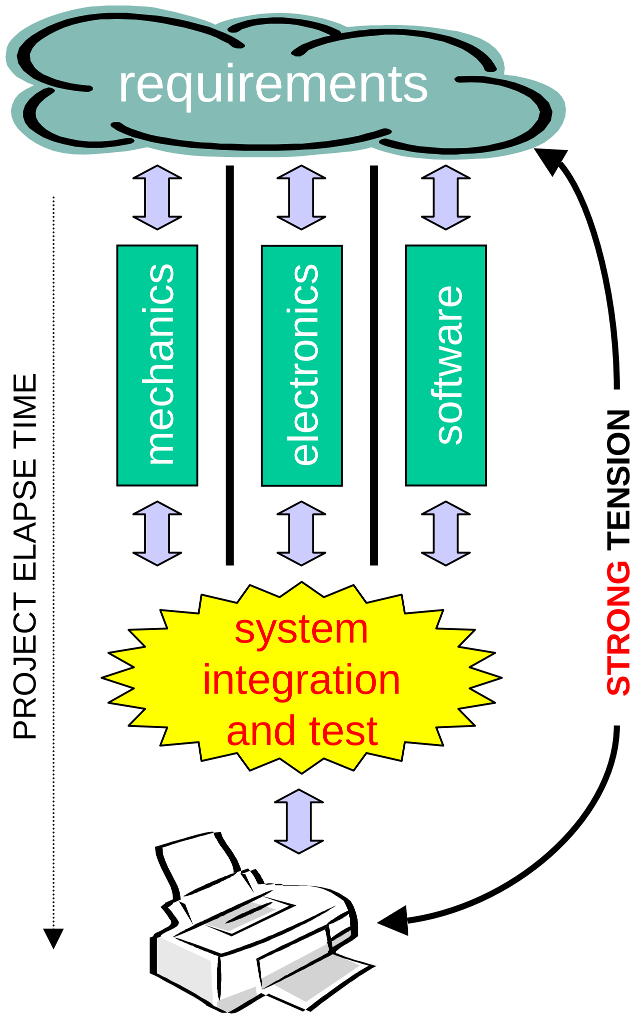
Agenda

- Context and motivation
- Basic techniques: Bond-graphs and VDM++
- Case study : Water tank level controller
- Tool support and integrated operational semantics
- Results and conclusions
- Current and future work

Beyond the Ordinary: Design of Embedded Real-time Control

- BODERC project @ ESI
- Sept 2002 - Apr 2007
- Multi-disciplinary design
  - mechanics
  - electronics
  - software
- High-tech systems focus
- Early life cycle trade-offs
- Industry as a laboratory
- http://www.esi.nl/boderc

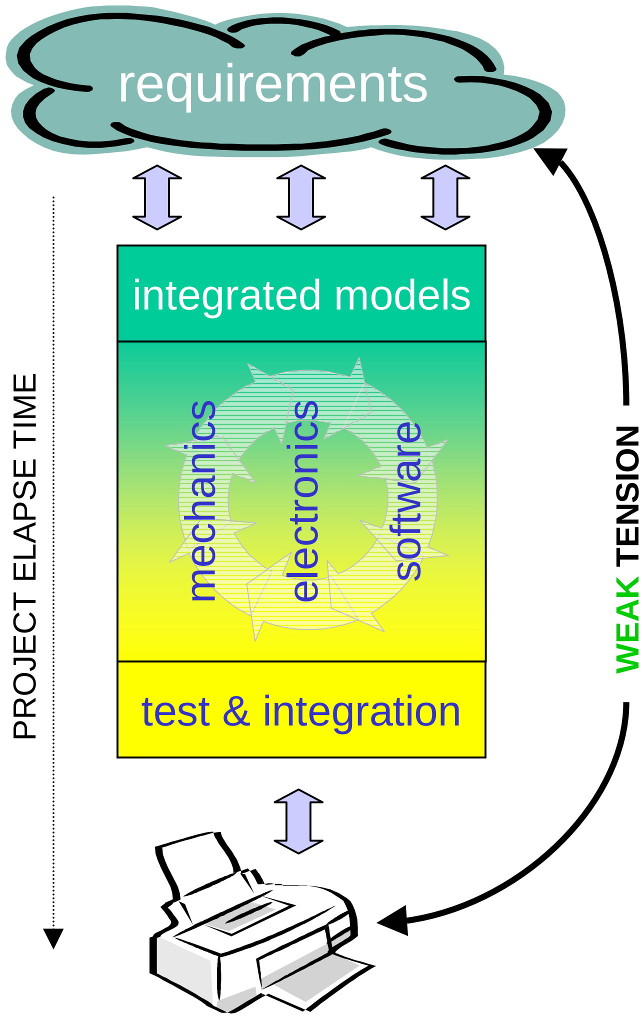
Design of High-Tech Systems - State of Practice



- design is typically mono-disciplinary organised
- domain specific methods and custom tools are used
- out-of-phase development and system-level focus lacking
- cross-cutting concerns postponed to the integration phase
- late validation & feedback

" INTEGRATION HELL "

Multi-disciplinary Systems Design - The Vision



- system level approach
- model-driven design
- integrated models & tools
- rapid evaluation
- early feedback
- support design dialogue
- continuous integration
- continuous validation
- less effort overall
- higher quality

The Challenge - Integrated Design Models (1)

- Notations and analysis techniques used by the disciplines are fundamentally different
  - mechanics : finite element methods
  - electronics : differential or difference equations
  - software : labelled transition systems
- Is a common notation feasible* at all?

  * [Henzinger & Sifakis, FM 2006 key note, LNCS 4085, pp 1-15]

The Challenge - Integrated Design Models (2)

- scope of discipline specific tools is widening
  - Matlab Simulink $\rightarrow$ Stateflow, Real-Time Workshop, TrueTime
  - Rhapsody $\rightarrow$ Simulink
  - UML $\rightarrow$ SysML
- bigger piece of the pie $\neq$ satisfy all stakeholders
- problems : poor abstraction, restrictive MoCs
- novel actor-based techniques* : Ptolemy-II
- problems : disruptive approach, poor semantics

* [http://ptolemy.eecs.berkeley.edu]

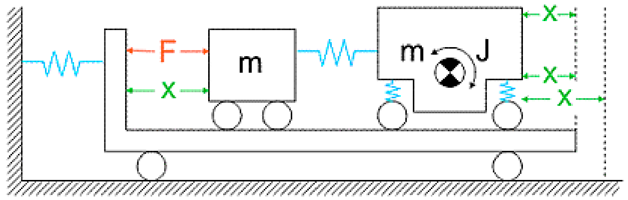
Our approach - Integrated Design Models (3)

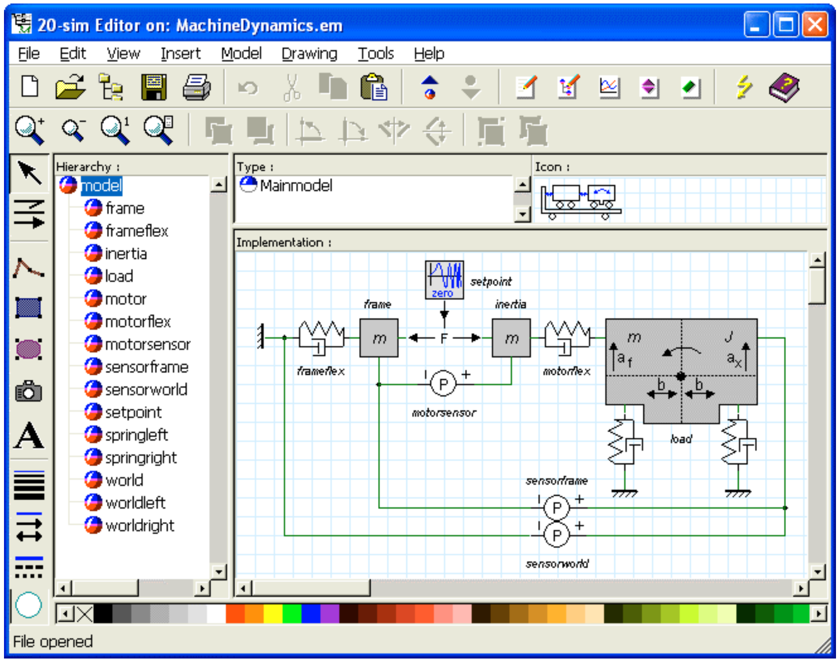
- Cross the continuous time - discrete event divide
- Select a well-defined (formal) notation on either side
- Explore semantic integration of those notations
- Implement tool support for these extensions
- Investigate models by (reliable) co-simulation

- Expected benefits:
  - good abstraction facilities on both sides of the divide
  - supports light-weight modelling required in early stages
  - few a-priori MoC specific restrictions → avoid design bias
  - fits in design flow → low acceptance threshold for industrial uptake

Continuous Time Realm - Bond Graphs

- dynamic systems modelling, physics domain independent
  - mechanics
  - electronics
  - pneumatics
- graphical notation: Bond graphs*
- formal analysis for algebraic loops and differential causalities
- model validation through simulation and visualisation
- industry grade tool support http://www.20sim.com

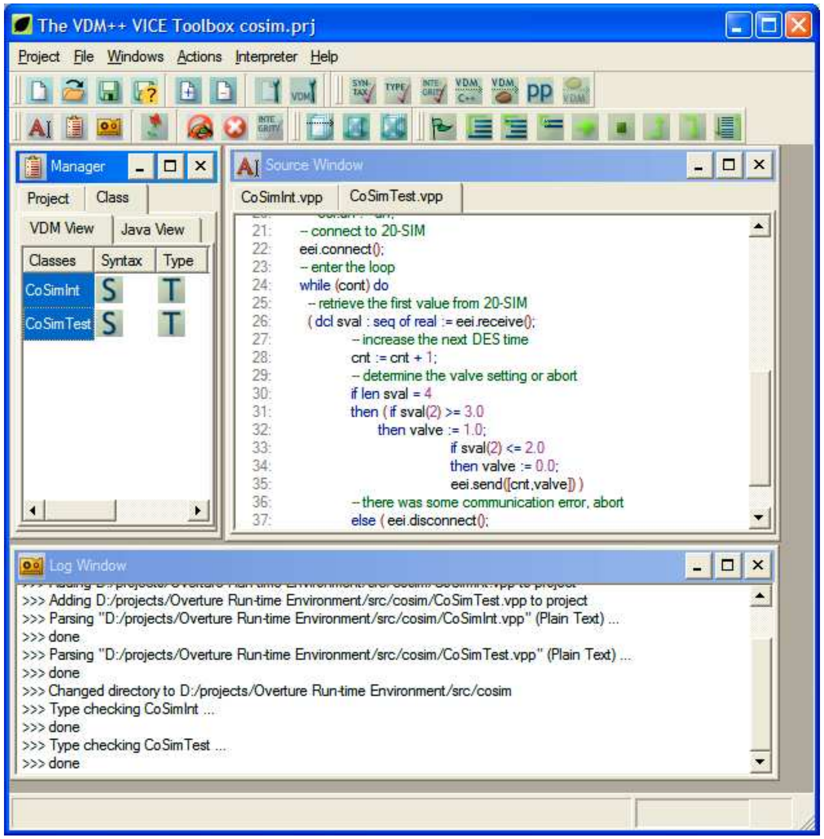




* [Gawthrop, Bevan, IEEE Control Systems Magazine, April 2007, pp 24 - 45]

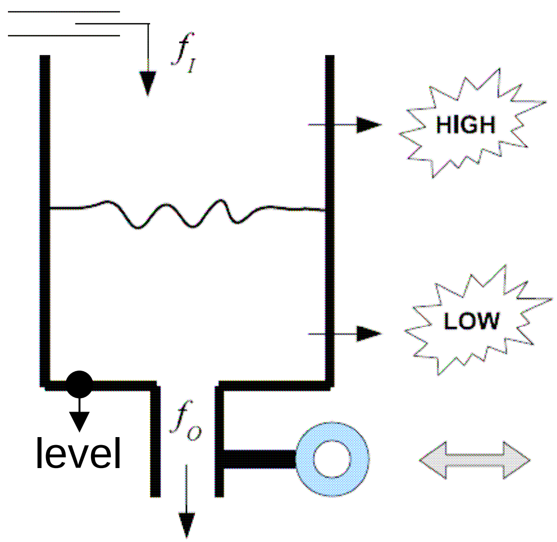
Discrete Event Realm - VDM++

- object-oriented formal model-based specification language
- concurrency through threads
- round-trip engineering UML
- formal analysis of static and run-time (type) correctness
- model validation through prototyping & structured testing
- industrial grade tool support http://www.vdmtools.jp/en
- VICE extension* for real time, scheduling and deployment



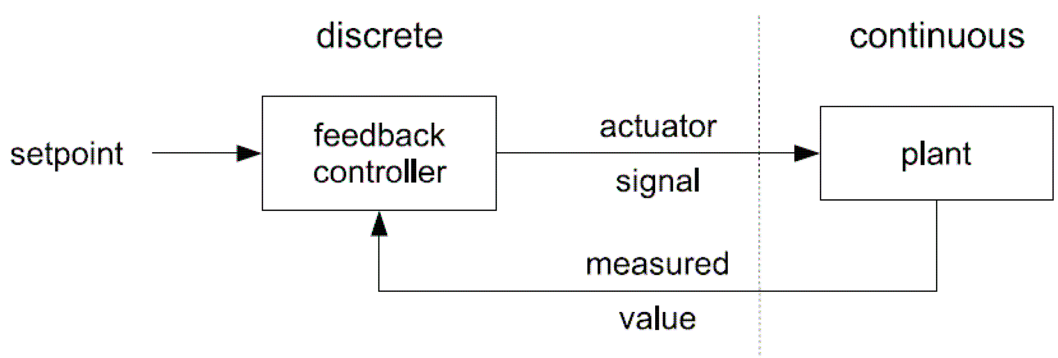
* [Verhoef, Larsen, Hooman, FM 2006, LNCS 4085, pp 145 - 162]

Our Approach by Example - water tank case (1)

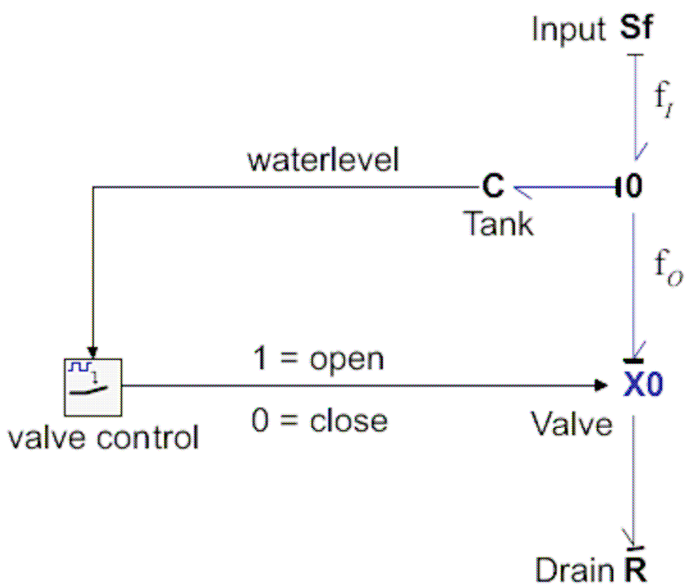


$$\frac{dV}{dt} = f_I - f_O$$

$$f_O = \begin{cases} \frac{\rho \cdot g}{A \cdot R} \cdot V & \text{if valve} = \text{open} \\ 0 & \text{if valve} = \text{closed} \end{cases}$$



Our Approach by Example - water tank case (2)

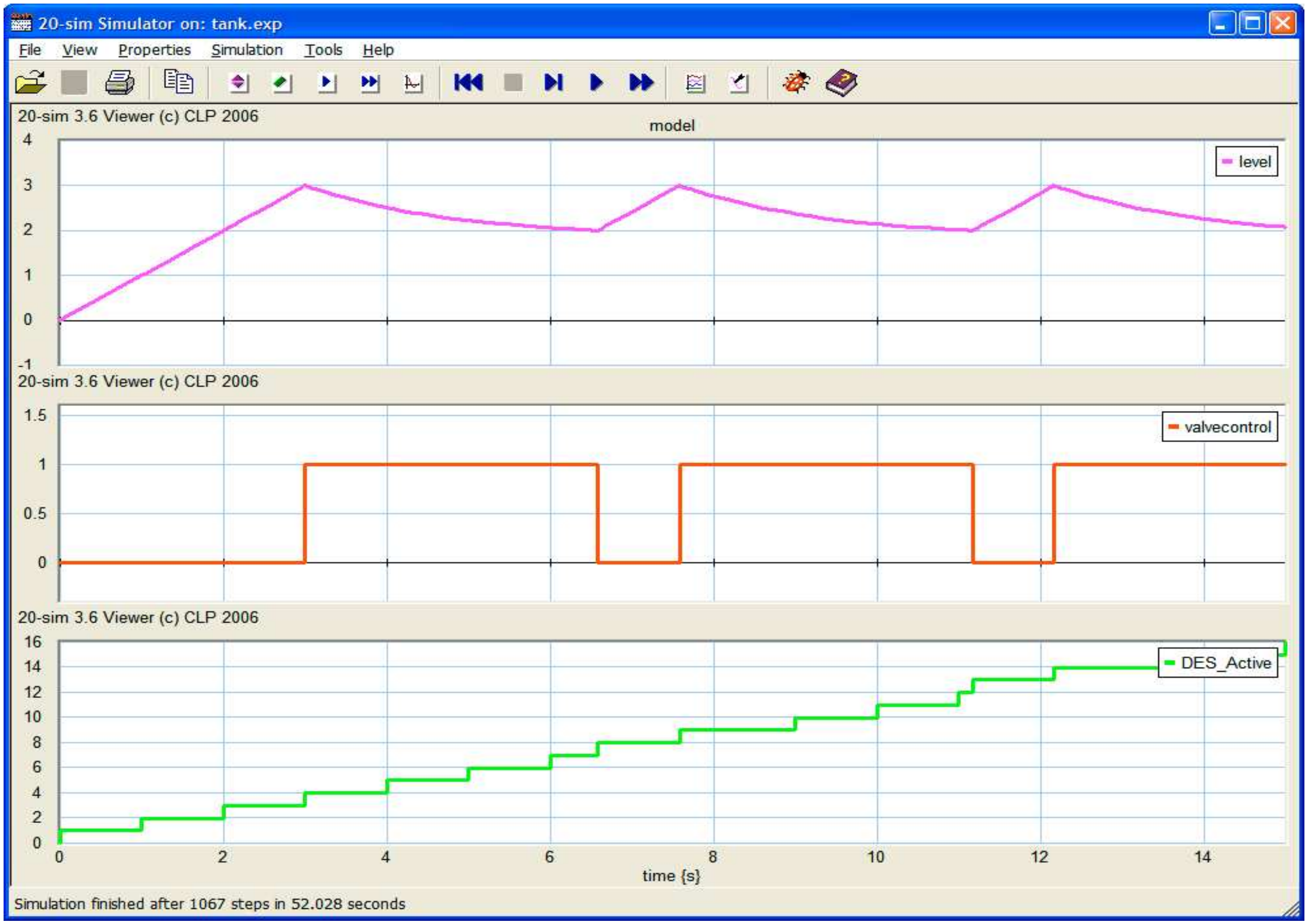



```
variables
  real volume, level;
parameters
  real area = 1.0;
  real gravity = 9.81;
  real density = 1.0;
equations
// p.e = pressure, p.f = flow rate
// integrate flow to obtain volume
  volume = int(p.f);
  level = volume / area;
  p.e = gravity * level * density;
```

Our Approach by Example - water tank case (3)

```
class Controller
instance variables
  static public level : real;
  static public valve : bool := false  -- default is closed
operations
  static public async open: () ==> ()
  open () == duration(0.05) valve := true;
  static public async close: () ==> ()
  close () == cycles(1000) valve := false;
  loop: () ==> ()
  loop () ==
    if level >= 3 then valve := true -- check high water mark
    else if level <= 2 then valve := false; -- check low water mark
threads
  periodic(1.0,0,0,1.0)(loop)
sync
  mutex(open, close, loop)
end Controller
```

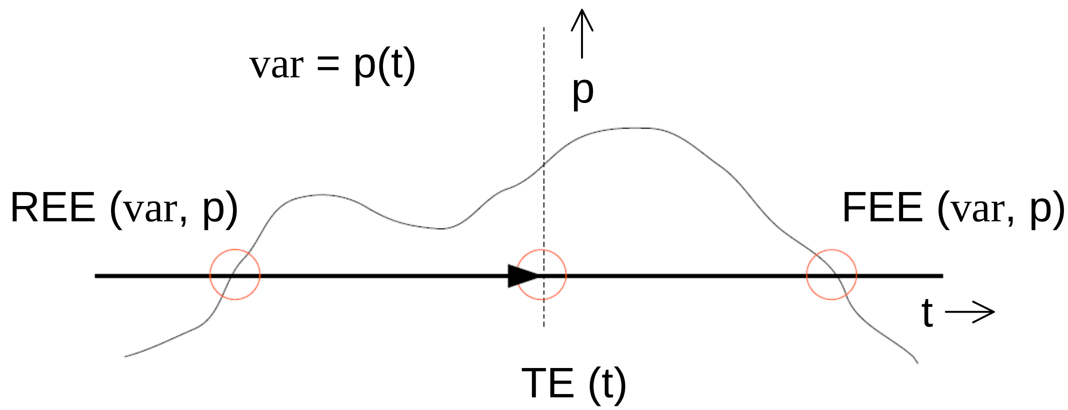
Our Approach by Example - water tank case (4)



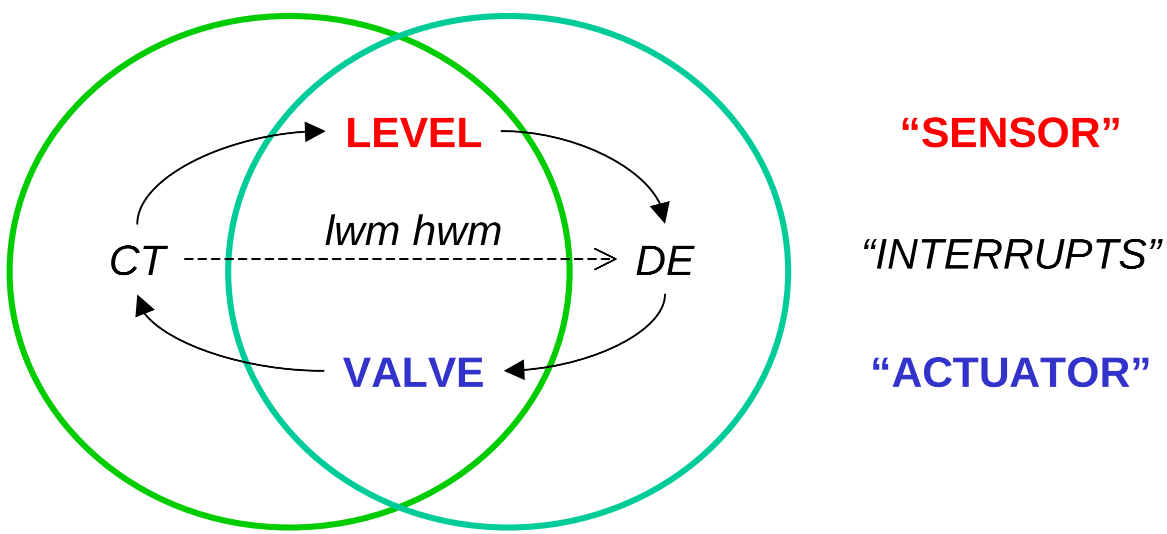
Integrated Operational Semantics (1)

- Continuous Time model
  - sets of differential equations
  - approximate solution numerically
  - discrete integration over some time interval
  - many "solver" algorithms available e.g. Euler
  - CT shares state variables with DE model
  - capture state events: zero-crossing detection
  - capture time events: proceed to time t > now

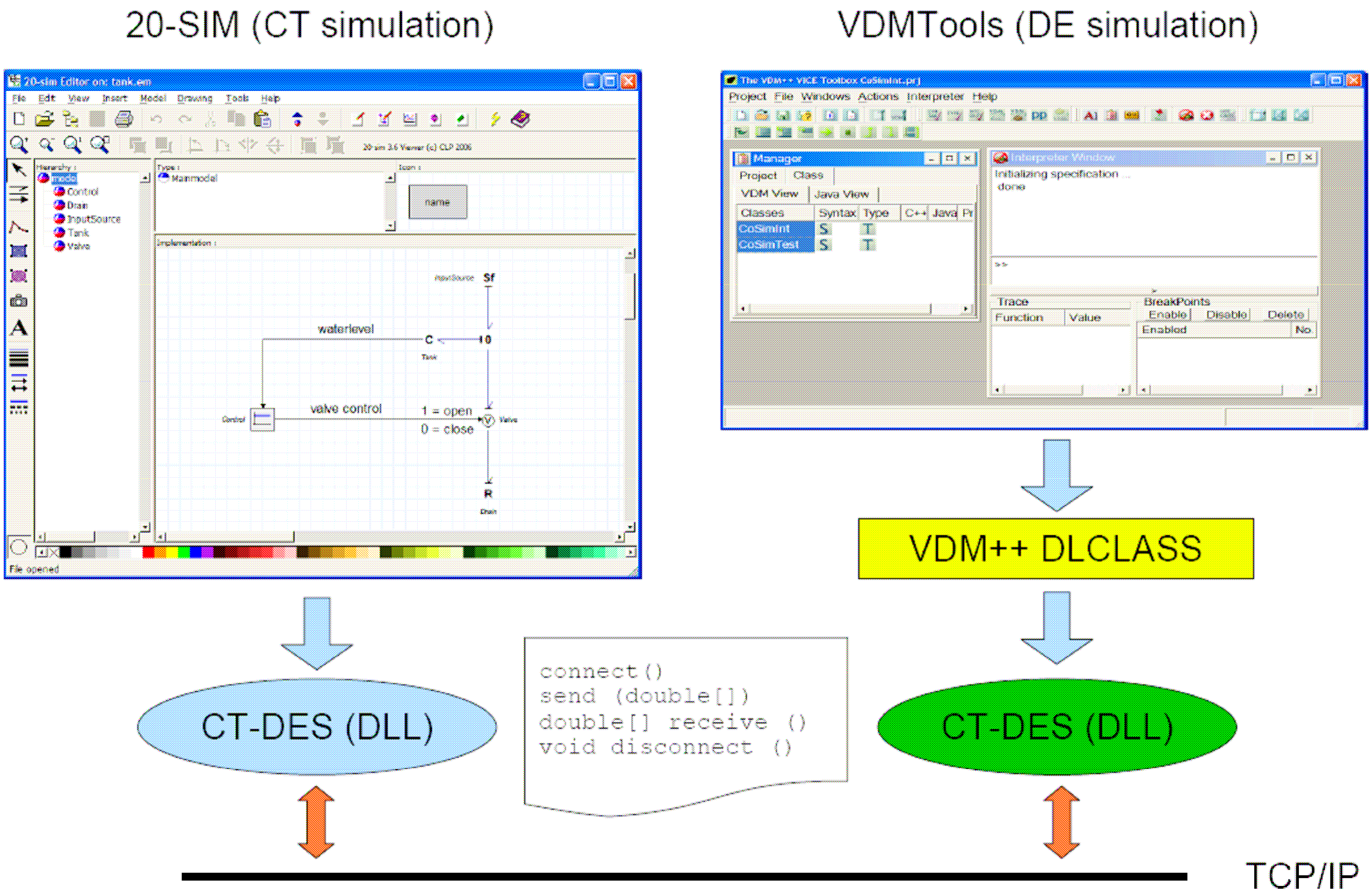
Integrated Operational Semantics (2)



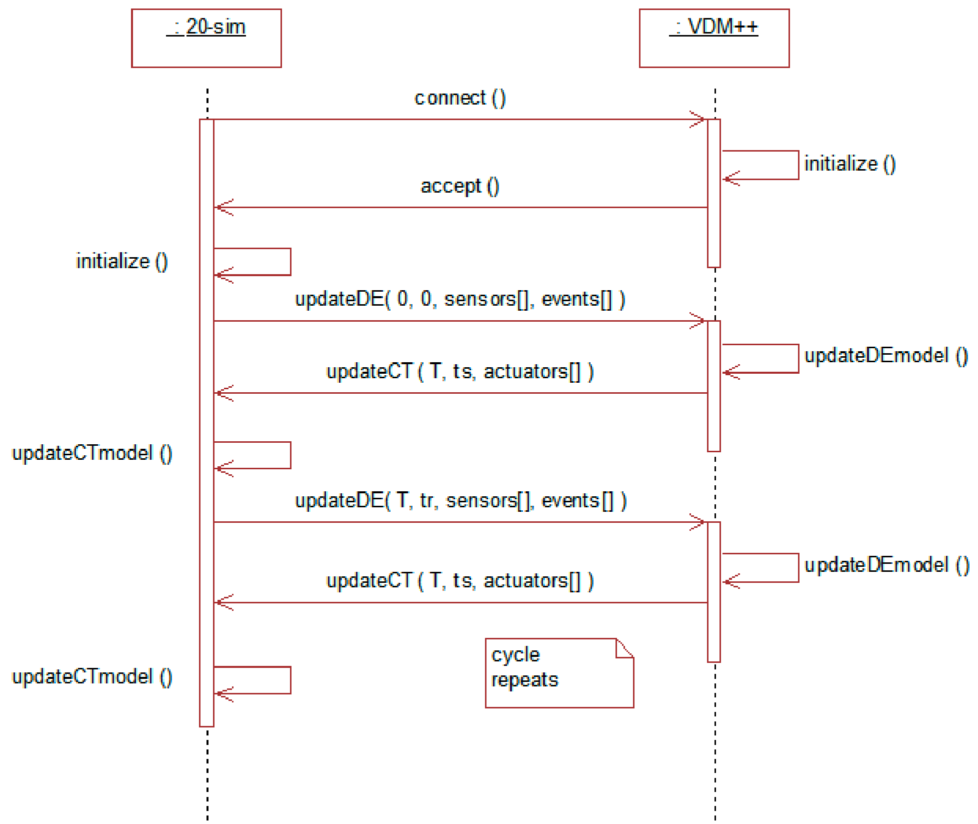
lwm = FEE (level, 2.0) *hwm* = REE (level, 3.0)



Tool Support (1)



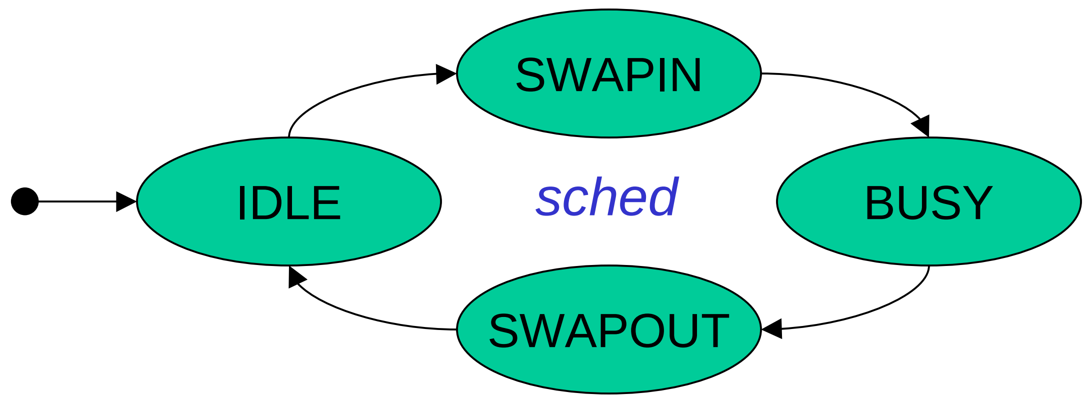
Tool Support (2)




```
sensor[1] = cpu1.Controller‘level
actuator[1] = cpu1.Controller‘valve
event[1] = REE(level,3.0) -> cpu1.Controller‘open
event[2] = FEE(level,2.0) -> cpu1.Controller‘close
event[3] = TE(15.0) -> abort
```

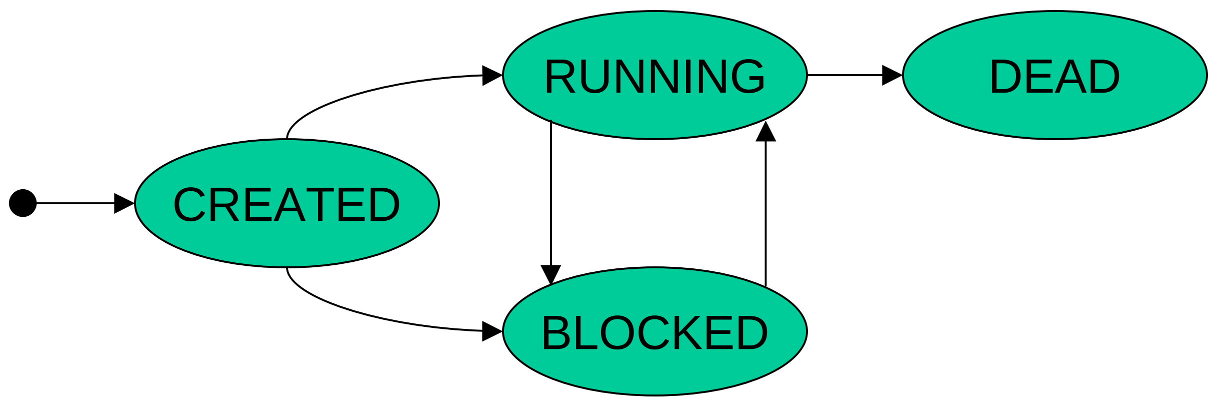
Integrated Operational Semantics (3)

- Discrete Event model
  - assume given a set of resources R
    { cpu_1, cpu_2, cpu_3, bus_1, bus_2 }
  - assume given an architecture
    $bus_1 \rightarrow \{cpu_1, cpu_2\}$, $bus_2 \rightarrow \{cpu_2, cpu_3\}$
  - each resource has a scheduling state *ss*



Integrated Operational Semantics (4)

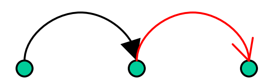
- Discrete Event model
  - each resource $r \in R$ has a set of tasks $r.T$ and an active task $r.\mathit{at} \in r.T \vee r.\mathit{at} = \mathbf{nil}$
    - cpu $\rightarrow$ *threads*
    - bus $\rightarrow$ *messages*
  - each task $t \in r.T$ has an execution state *es*



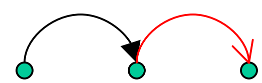
Integrated Operational Semantics (5)

- Discrete Event model
  - each active task r.ta ≠ **nil** can
    - either execute a state transition
    - or execute a time transition

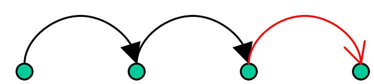
`x := 10`



`duration (100) x := 10`



`cycles (1000) (x := 10; y := 20)`



caveat: `duration (0)` is a valid time transition

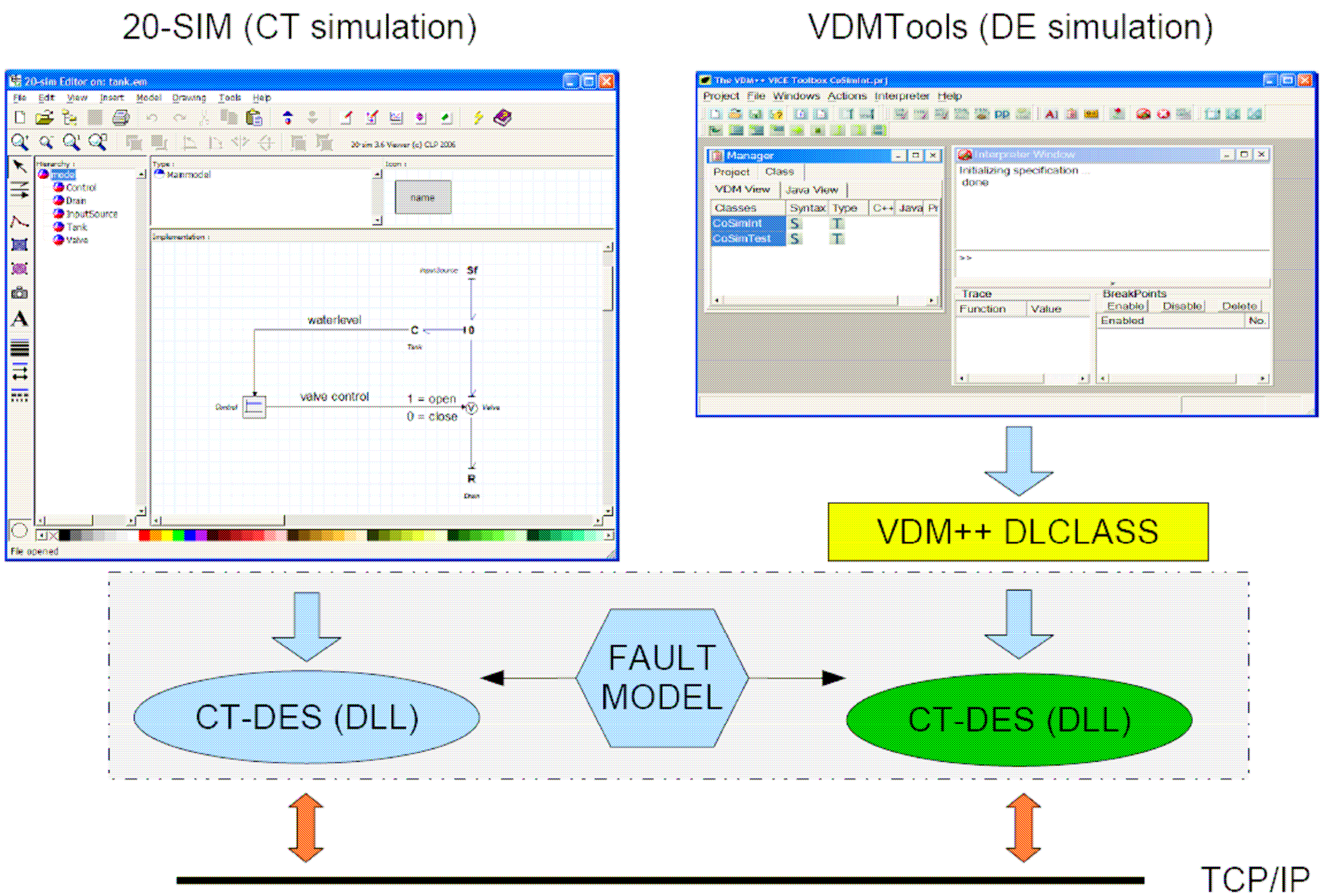
Integrated Operational Semantics (6)

- process state transactions until all resources are either idle or need to make a time transition
- determine the smallest DE time step t_{req} over all R
- CT solver is asked to move to $t + t_{req}$
- CT solver reaches $t + t_{rel}$ with $t_{rel} \leq t_{req}$
- time on all resources is updated to $t + t_{rel}$
- events are handled (if any occurred)
- guards and scheduler are re-evaluated (if affected)
- repeat until abort time event is reached

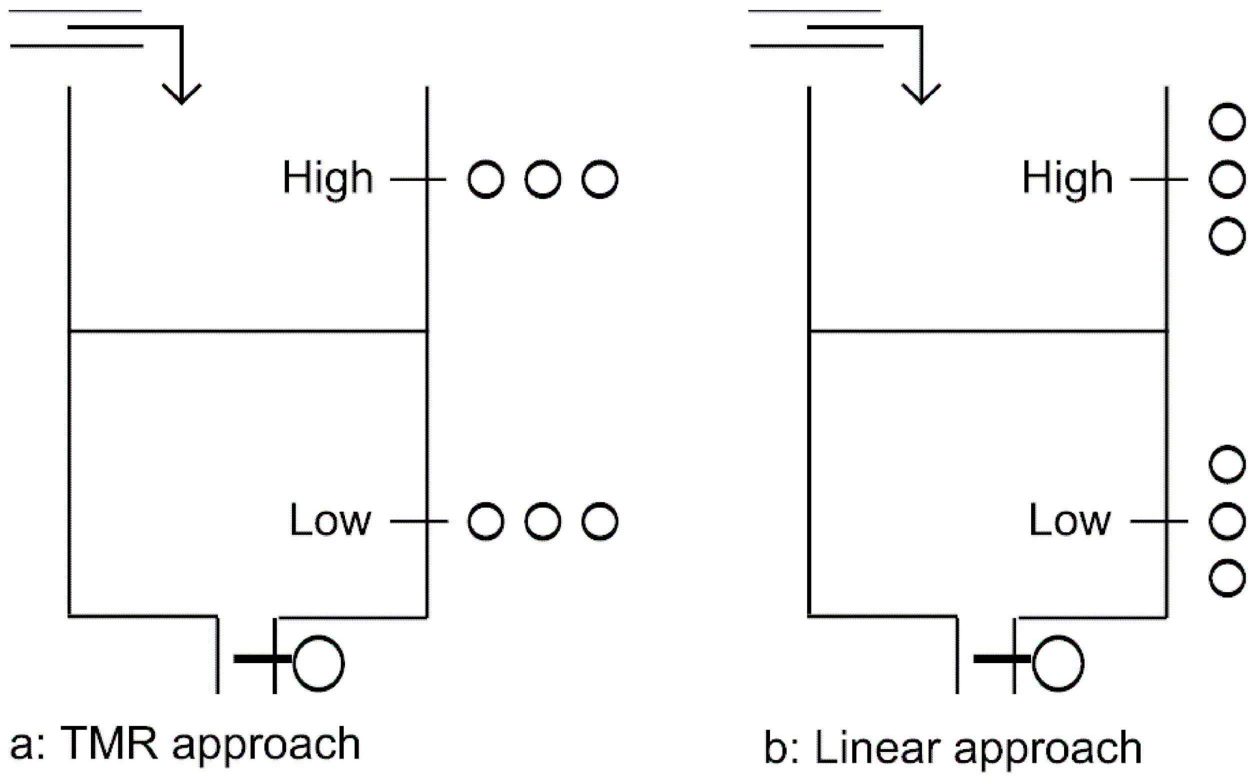
Results and conclusions

- Coupling does not restrict tools or add complexity
- Co-simulation enables cross-discipline dialogue
- Small model size due to abstraction on both sides
- Evaluation of design options requires low effort
- Discipline specific analysis on models is still feasible
- Generic integrated operational semantics
- Heterogeneous simulation is within reach
- Case studies: light-weight models can be accurate

Future Work (1)



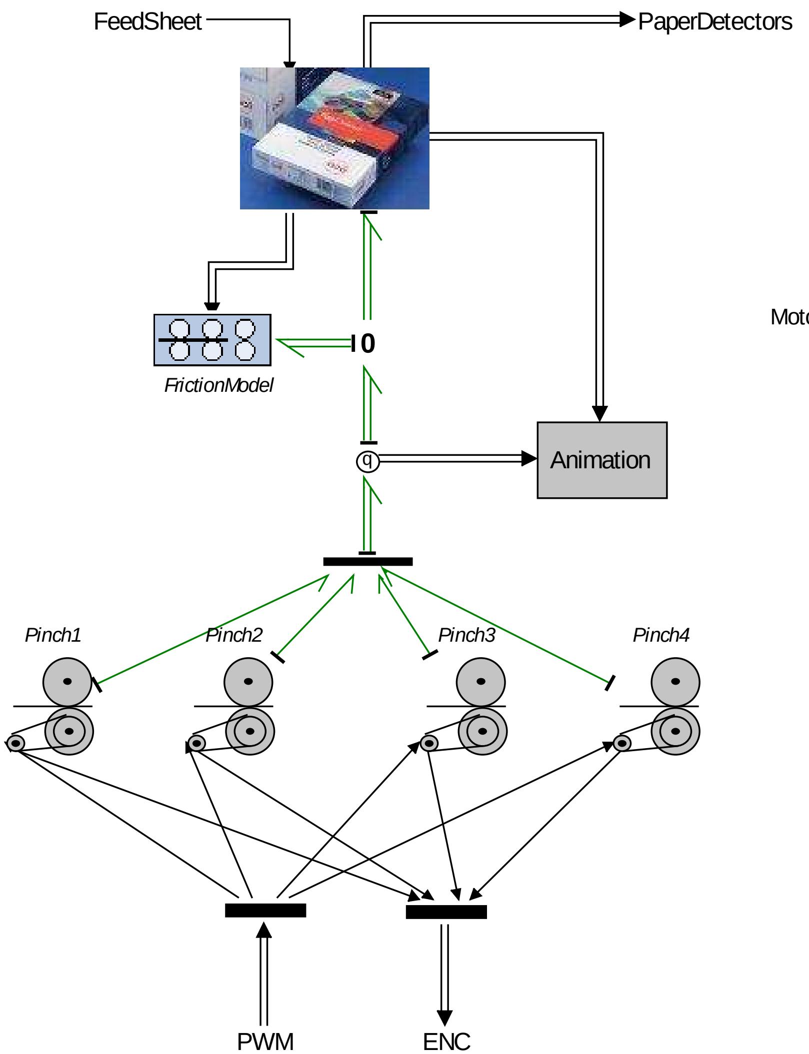
Future Work (2)

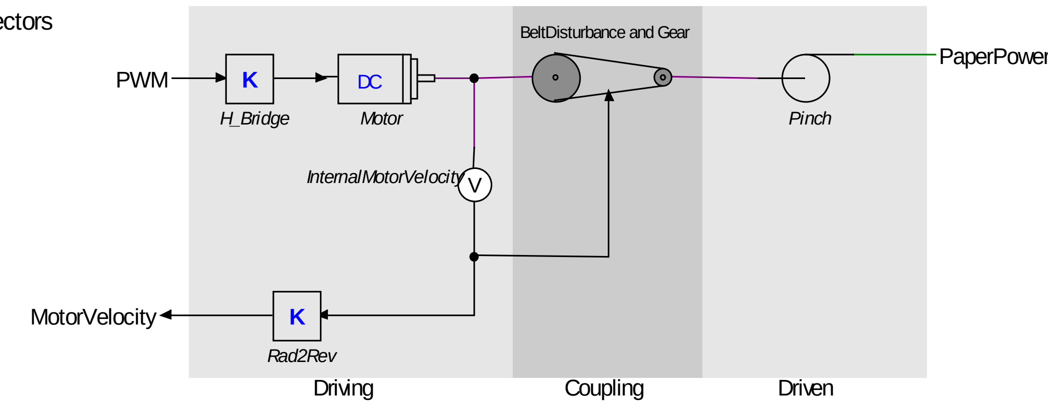


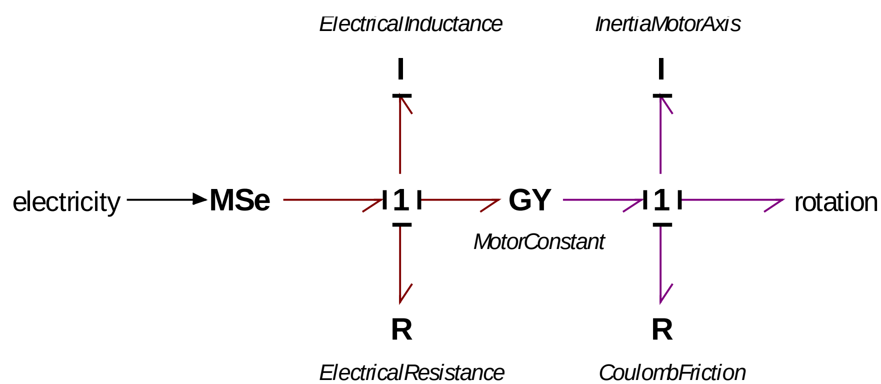
[Andrews, Verhoef, Fitzgerald, DSN 2007]

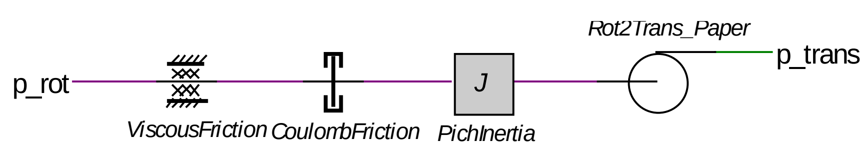
Printer paper path - case study (1)

Printer paper path - case study (2)

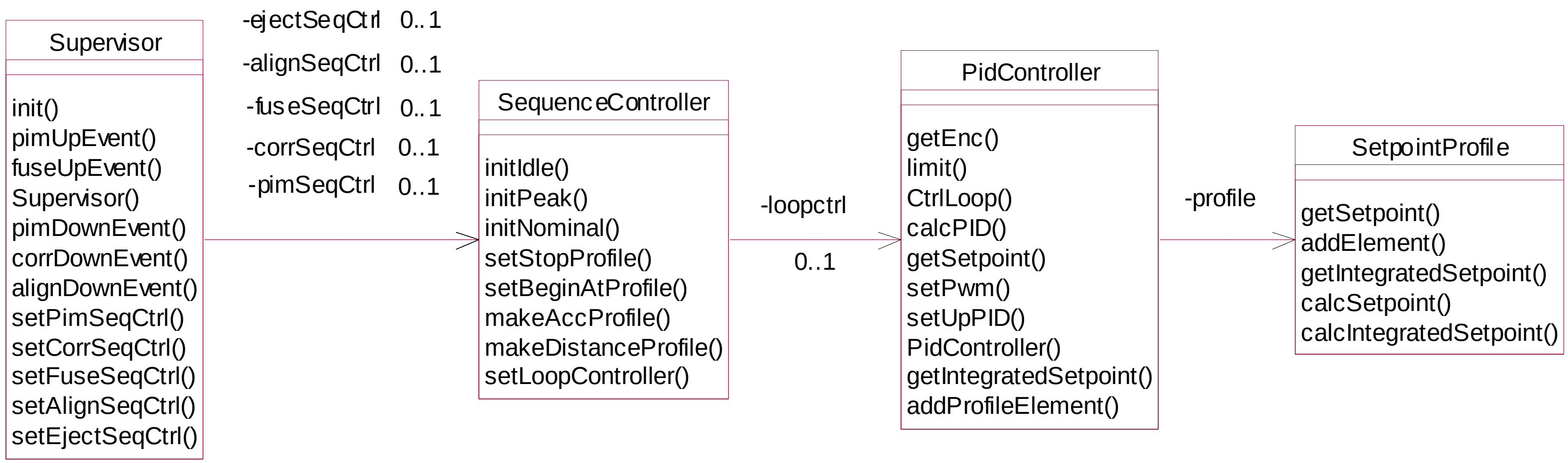






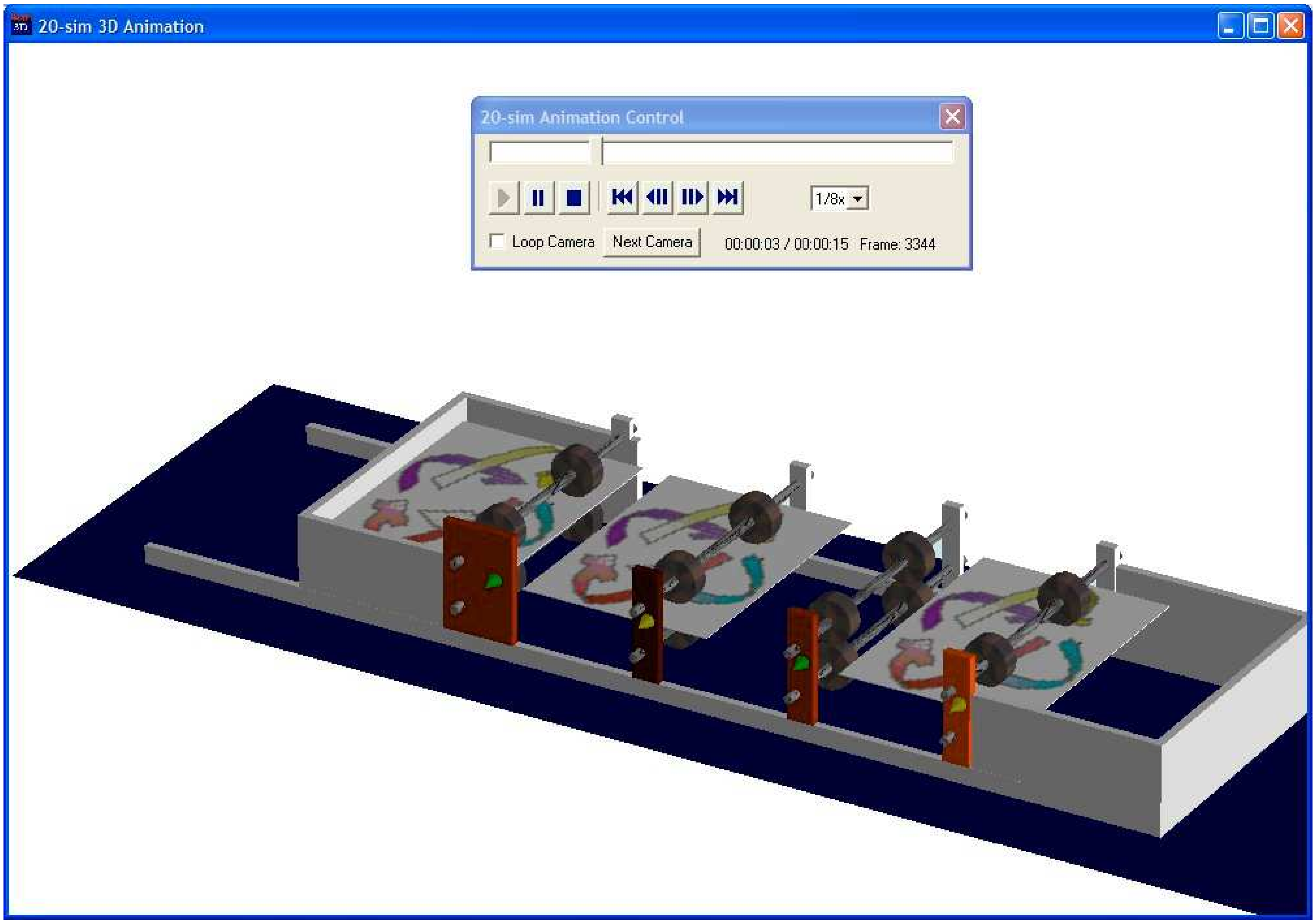


Printer paper path - case study (3)



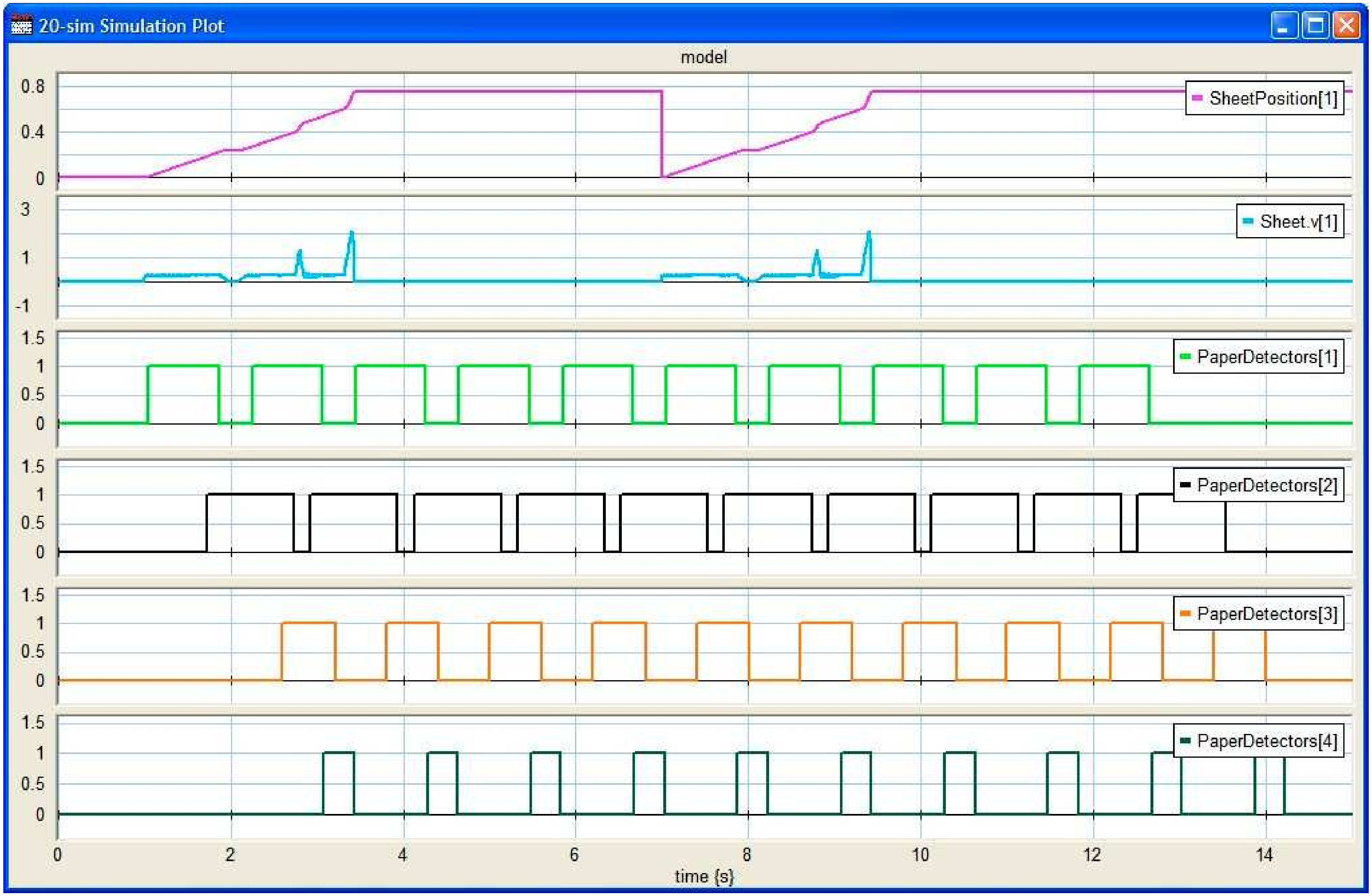

```
  public CtrlLoop: () ==> ()
  CtrlLoop () ==
    -- first retrieve the current encoder value
    ( dcl measured_value : real := ENCODER_GAIN * getEnc();
      -- output the previous value if we are time synchronous
      if hold then setPwm(curr_profile, curr_setpoint, curr_error, next_output);
      -- calculate the new pwm control value
      if feedback
      then next_output := limit(calcPID(measured_value))
      else next_output := limit(getSetpoint());
      -- output the new value directly if we are not time synchronous
      -- otherwise wait until the next period is due
      if not hold then setPwm(curr_profile, curr_setpoint, curr_error, next_output) )

thread
  -- run the controller at 1 kHz and assume no jitter
  periodic (0.001, 0, 0, 0.001)(CtrlLoop)
```

Printer paper path - case study (4)



Printer paper path - case study (5)

Printer paper path - case study (6)



Printer paper path - case study (7)