Design Space Exploration through Co-modelling and Co-Simulation: The Pacemaker Challenge

MARTIN MANSFIELD

CARL GAMBLE, JOHN FITZGERALD, PETER GORM LARSEN

Newcastle University

AARHUS UNIVERSITET

Outline

1. Background: Co-modelling and Co-simulation
2. Pacemaker Co-model
3. Observations
4. Development Opportunities

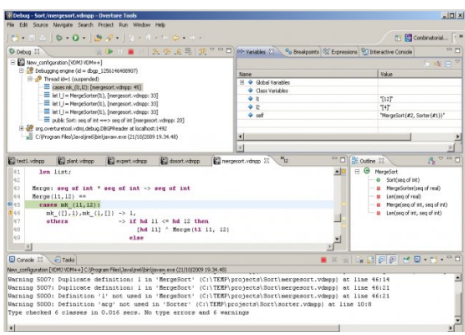
Background: Co-modelling

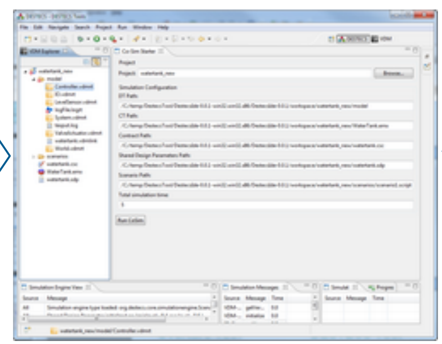
- Problem decomposition into disciplines
  (control, mechanical, electrical, software, human factors)
- Concurrent engineering required to improve time to market
- Important properties are multidisciplinary
- and so weaknesses are exposed late (integration)
- How to cross the boundaries between disciplines?

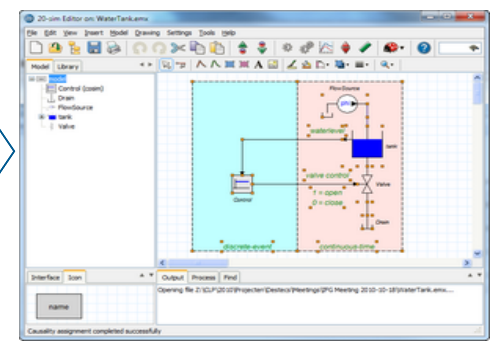
Background: Co-modelling

Discrete Event Model ⇔ Contract ⇔ Continuous Time Model

Overture	Crescendo	20-sim



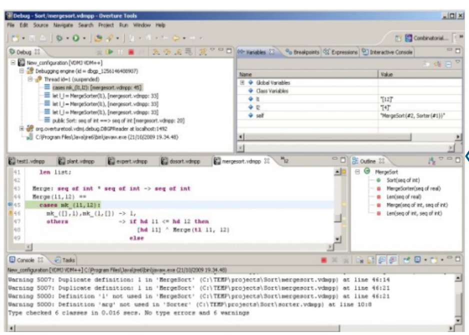


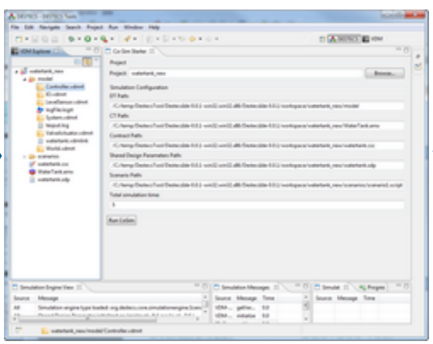


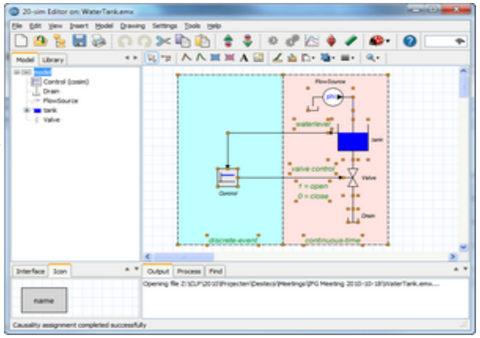
Background: Co-simulation

Discrete Event Simulator ⟷ Co-simulation Engine ⟷ Continuous Time Solver

Overture	Crescendo	20-sim







VDM Pacemaker History

- Basic VDM model (Overture Tool)
- VDM-SL → VDM++ → VDM-RT
- Generated simulation traces
- Coarse environment model
- Fair selection of pacing modes

(Macedo, Larsen, Fitzgerald)

2008

- Co-model (Crescendo Tool)
- Simple environment model
- Few pacing modes

(Gamble, Mansfield, Fitzgerald)

2012

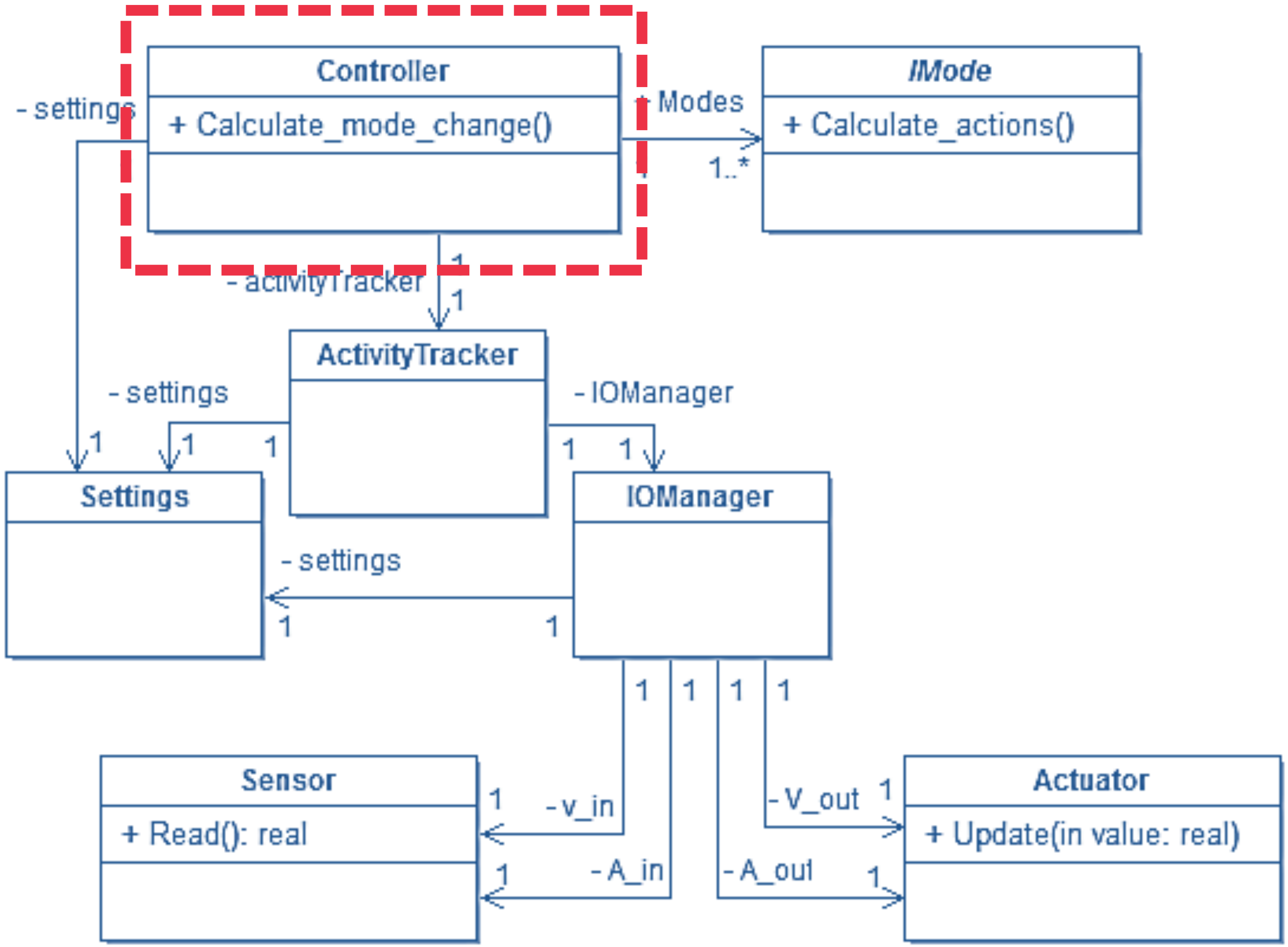
Latest Co-model

- Object of our study was a proof-of-concept for design space exploration for the pacemaker, based on co-models that could be extended

> 5.5 Noise Response
>
> In the presence of continuous noise the device response shall be asynchronous pacing.

- Mode changes can be challenging in embedded systems design
- Model mode change on noise detection for DDD to DOO and AAI to AOO
- We explore alternative configurations of noise detection

Discrete-event Model

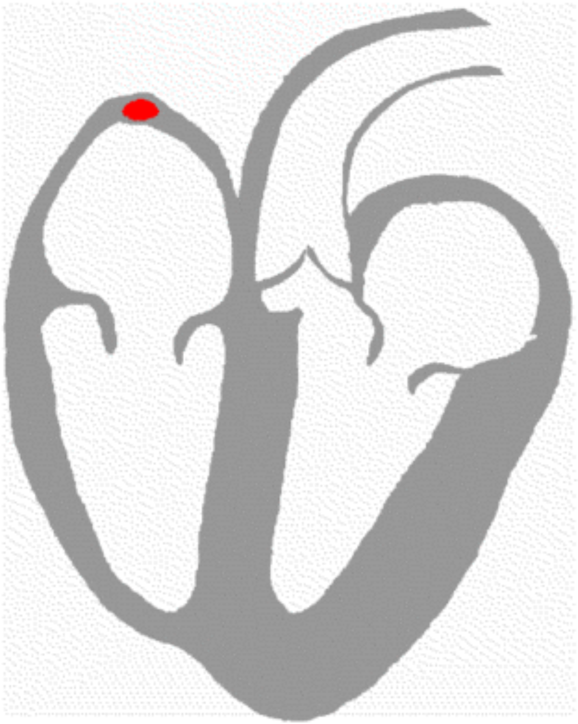


Discrete-event Model

```
types
  public Mode = <AAI> | <AOO> | <DDD> | <DOO>;

instance variables
  activity     : ActivityTracker;
  settings     : Settings;
  current_mode : Mode;

operations
  Step : () ==> ()
  Step() == (
    -- update timers
    activity.Step();

    -- Calculate if mode change is required
    Calculate_Mode_Change();

    -- delegate control to current mode
    modes(current_mode).Step();
  );

-- Run simulation at 1000Hz
thread periodic(1E6,0,0,0)(Step);
```

- Periodic thread runs control loop at 1KHz
- CT environment is sampled, with values stored for computation
- Control is delegated to the active mode subclass to calculate appropriate sensing/pacing behaviour

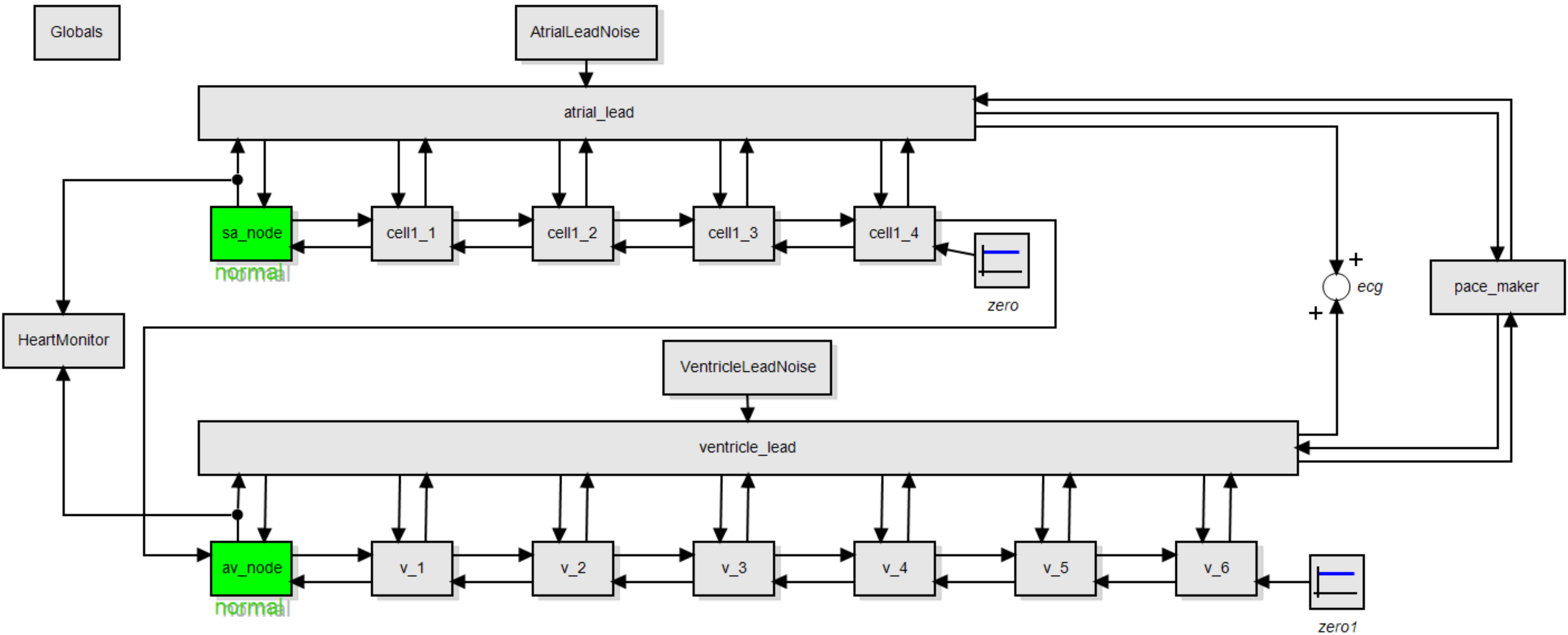
The Heart



- Heart pumping rhythm driven by electrical "action potentials" travelling through the fibres
- Heart arrhythmia (faults) stem from blockages and delays of the action potentials
  - Sinus bradycardia – slow pacing from the SA node
  - heart block – slow or absent transmission of action potential from atria to ventricles.
- An ECG output is not necessarily the goal of a good heart model

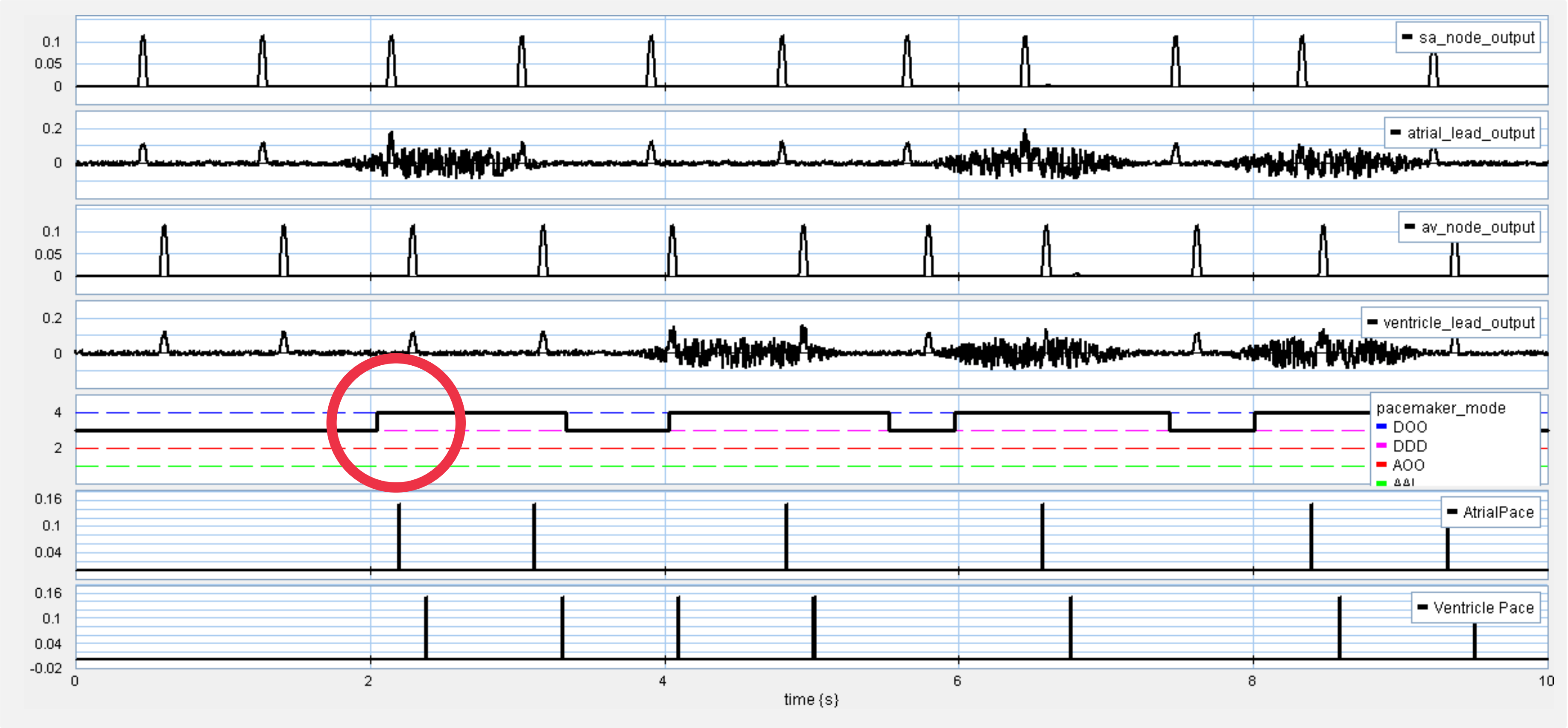
Illustration taken from https://commons.wikimedia.org/wiki/File:ECG_principle_slow.gif

Continuous-time Model



- Blocks correspond to systems of differential equations
- Model structured to allow different lead placements
- Sinus bradycardia and heart block modelled as faults

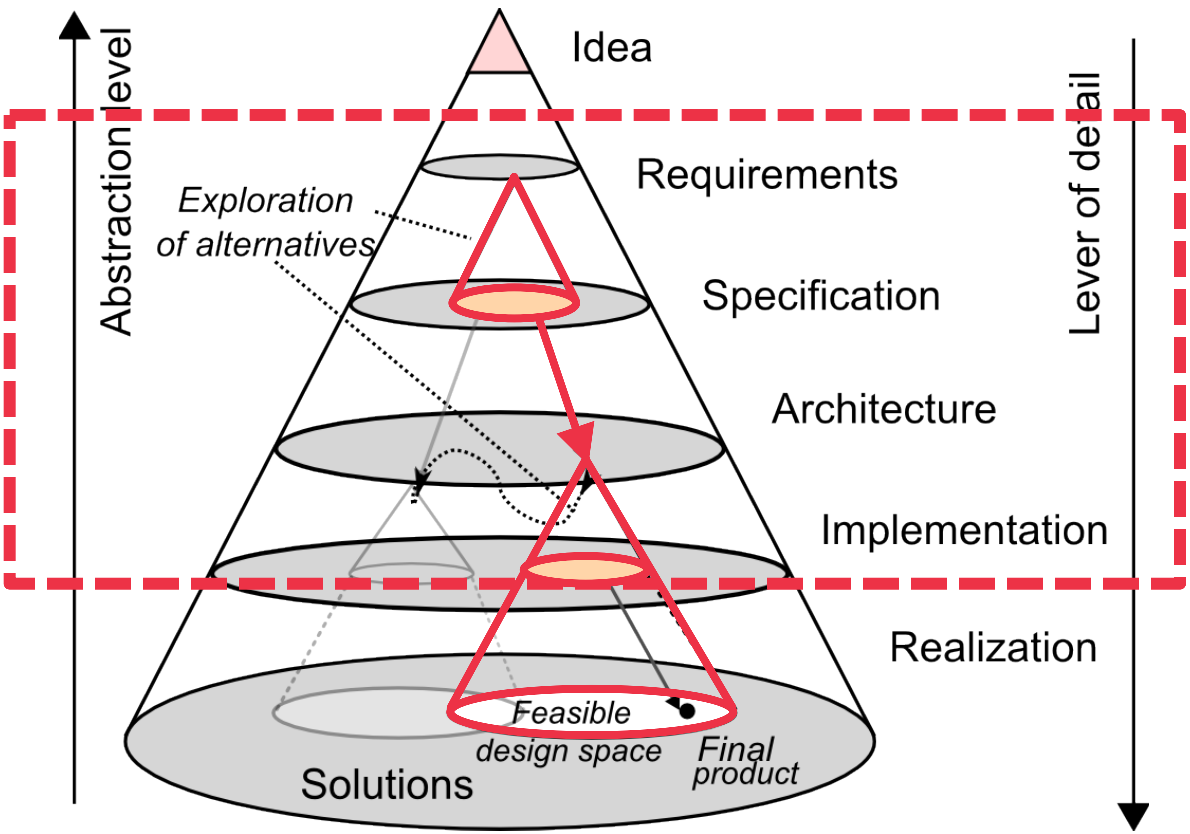
Co-simulation Output



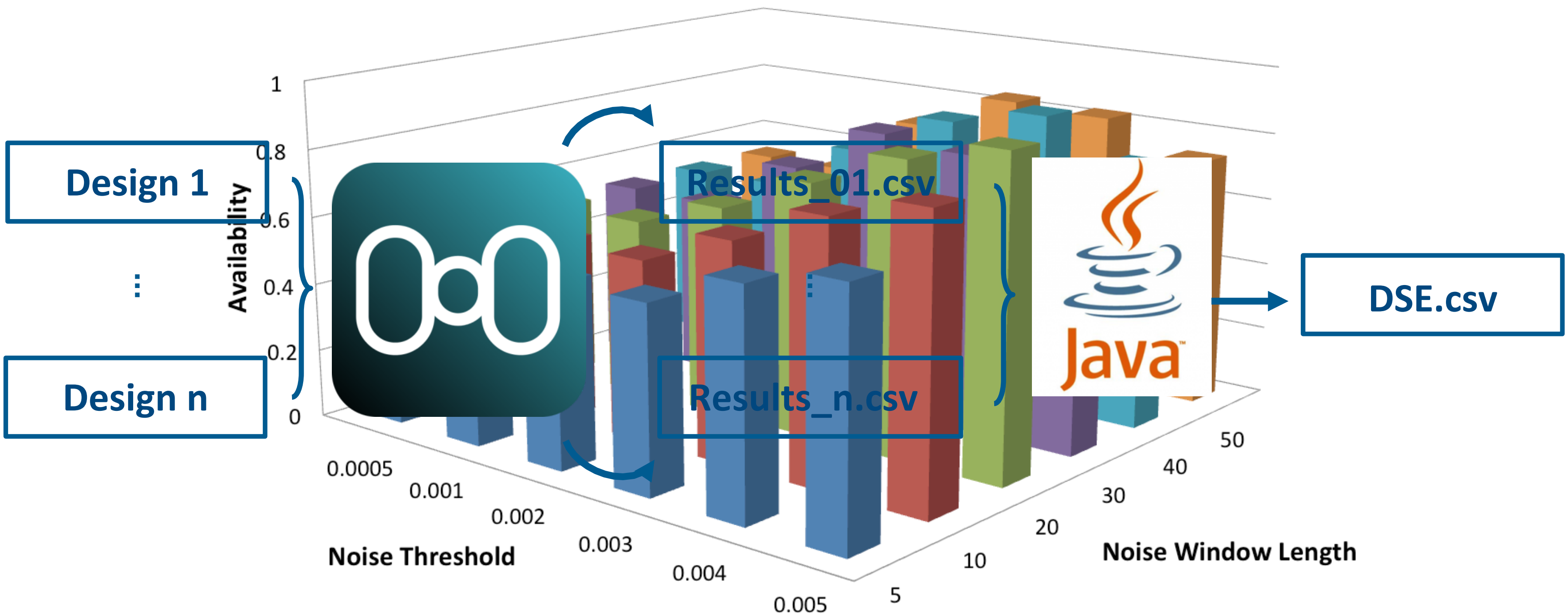
- Mode change due to noise in leads

Design Space Exploration

- Restrict potential designs to feasible solutions
- Visualisation and space management
- Quantification and qualification of design outcomes



Pacemaker DSE



- A design is a tuple of (noise threshold, window size)
- Summarised as proportion of run time in the "correct" mode

DSE: Closing the Loop

Open loop-Brute force

Default in Crescendo, explores the complete deign space

Closed loop-Space Culling

Useful where we know something about relation between design space and objective space

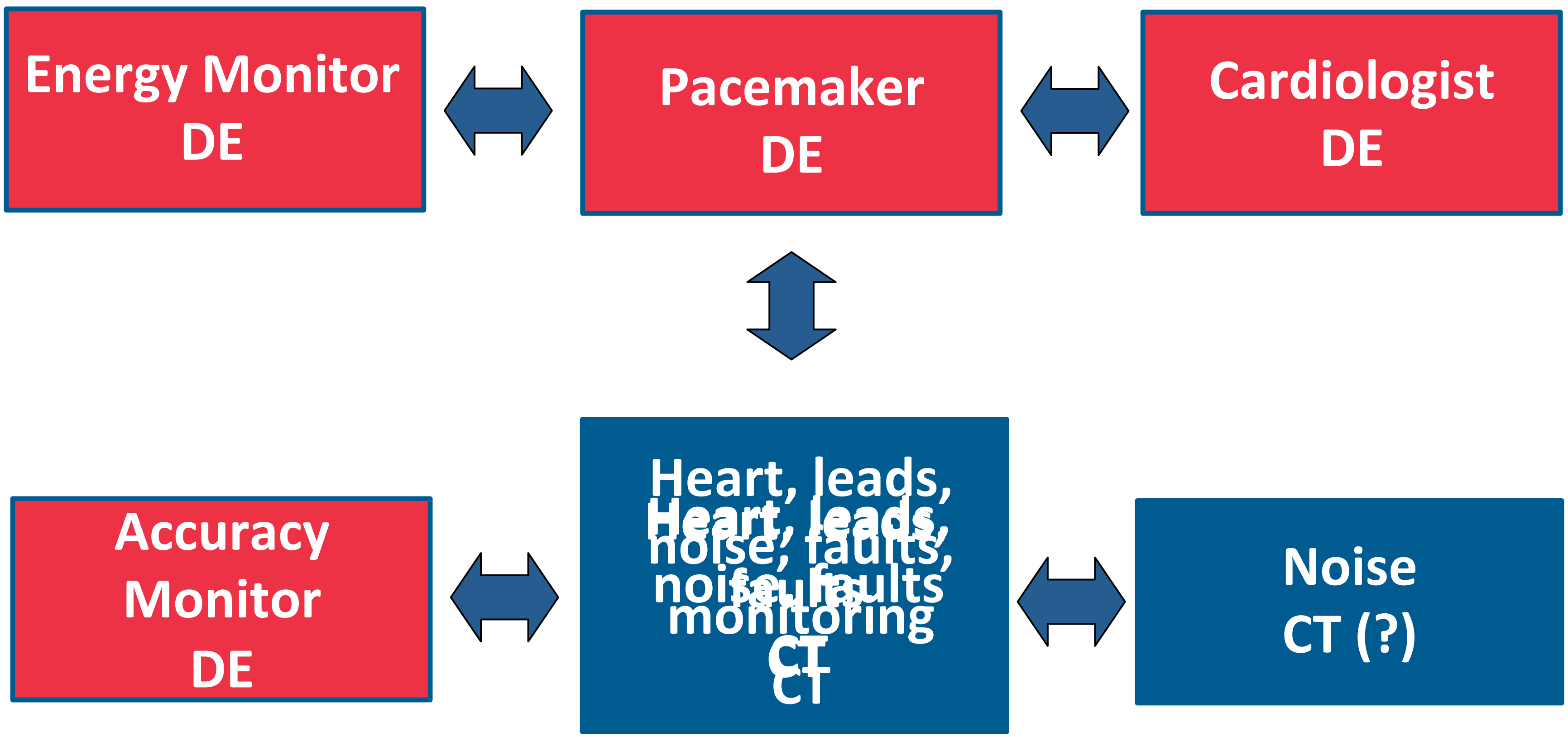
Closed loop-Genetic Algorithm

Utilising objective functions / Pareto optimisation to select the best design parameters

Observations

- Produced a proof-of-concept co-model and demonstrated co-simulation
- One model provides ability to observe effects of cyber or physical design decisions on system behaviour as a whole
- A small subset of pacing functionality has been implemented
  - Lacking specification to justify additional features
- Faults are implemented in CT environment, but introducing faulty controller implementations would be useful
  - Single event upsets can be an issue with embedded controllers

Future Work: Multi-models



Overture: What Next?

1 Year

- Better repertoire of DE-side fault models (SEU example): Could overture have support for manipulating memory values without editing controller code?

5 Years

- Hardware in the loop: interface to use VDM-RT to control hardware

- A complete pacemaker development, requirements to code, fully VDM-based, fully documented

10 Years

- VDM-RT formal semantics – permits demonstration of static analysis, e.g. of timing properties.

Design Space Exploration through Co-modelling and Co-Simulation: The Pacemaker Challenge

Thank you for listening.

martin.mansfield@ncl.ac.uk