Introduction—How do control system design engineers use models and simulation?

Pieter J. Mosterman

Panelists

- Lennart Ljung, *Linköping University*, Sweden
- Albert Benveniste, *INRIA/IRISA*, France
- Jonathan Sprinkle, *University of California, Berkeley*, USA

Introduction

- Let's look at a body electronics example
  - A power window

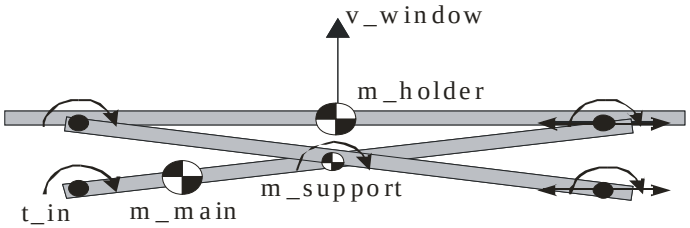
  - We have a few requirements:
    - We want the window to go up and down
    - We want it to start moving within 200 [ms] after pressing a button
    - We want it to open and close within 4 [s]

Plant Design – System Identification

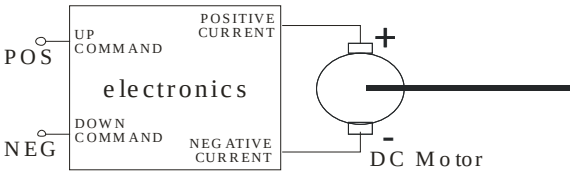
- Identify some door measurements

⟹ length, weight, heigth

- Devise a lift mechanism



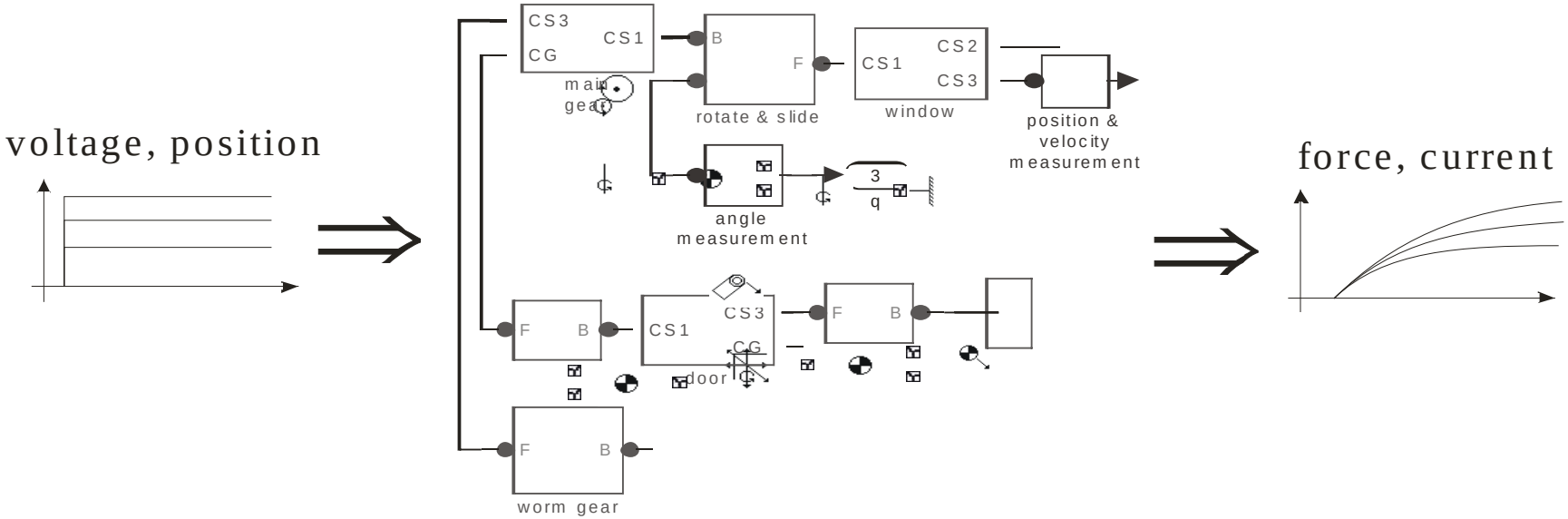
- Select a preliminary actuator



Dynamic System Identification

- Obtain coefficients of dynamic behavior
  - A power window

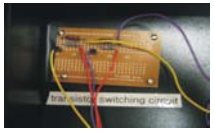
  - Design dynamic model



Hardware Design

- Actuator
  - DC motor

  - Signal conditioning

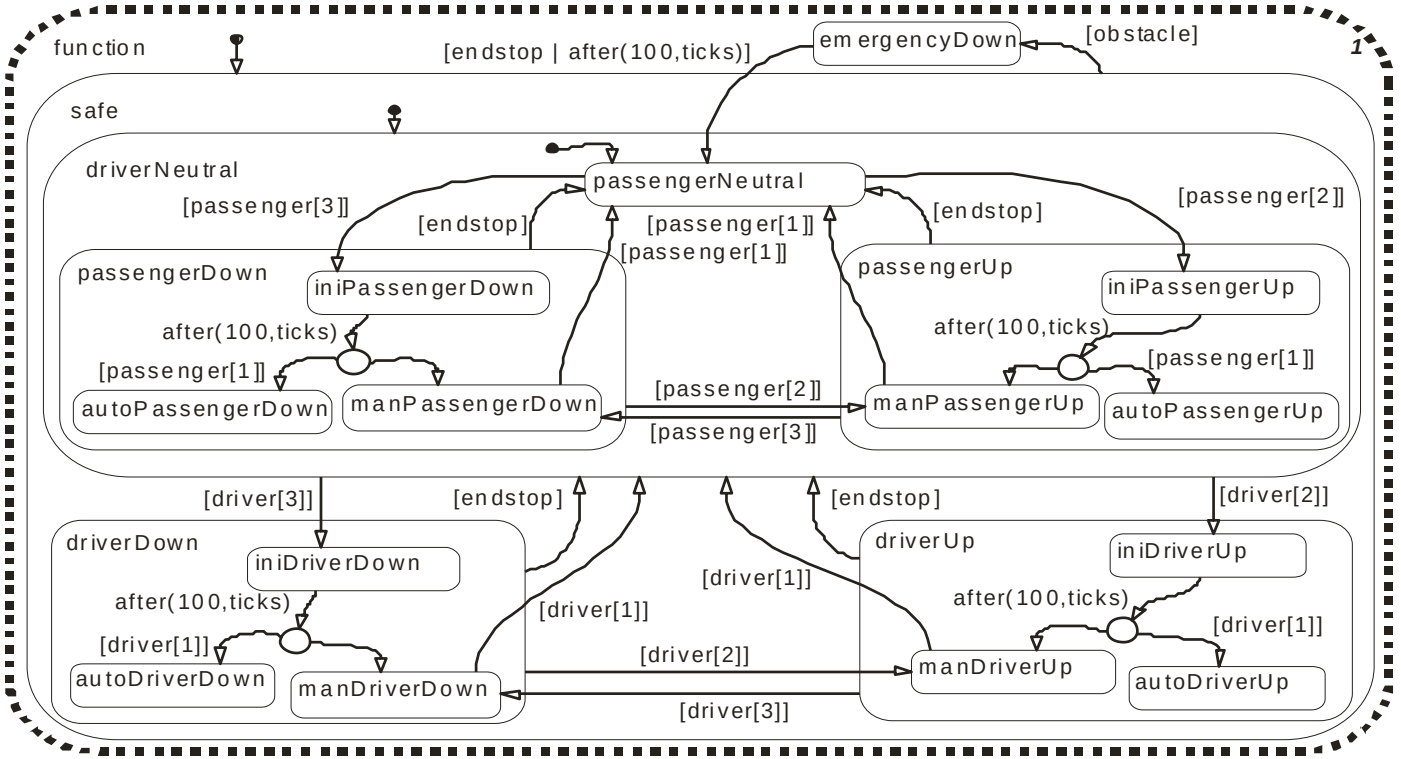


- Sensor
  - Armature current

  - Window force

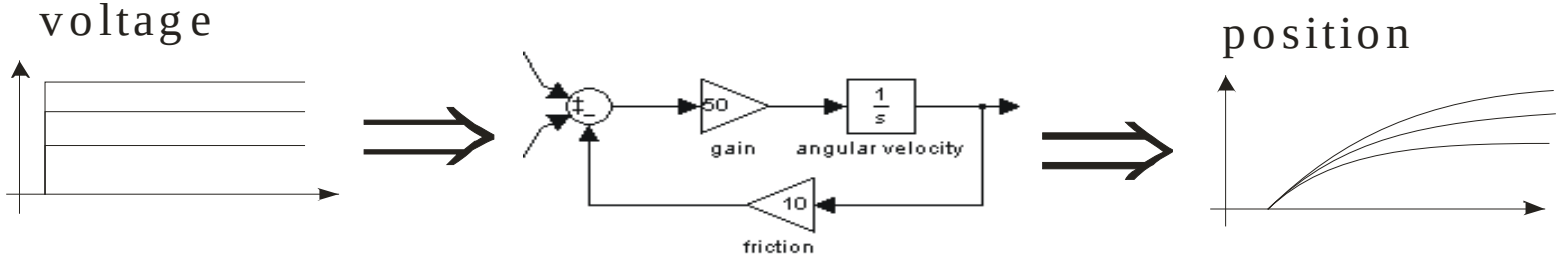
Initial Control Design

- State transition behavior of the window control



Feedback Control Loop

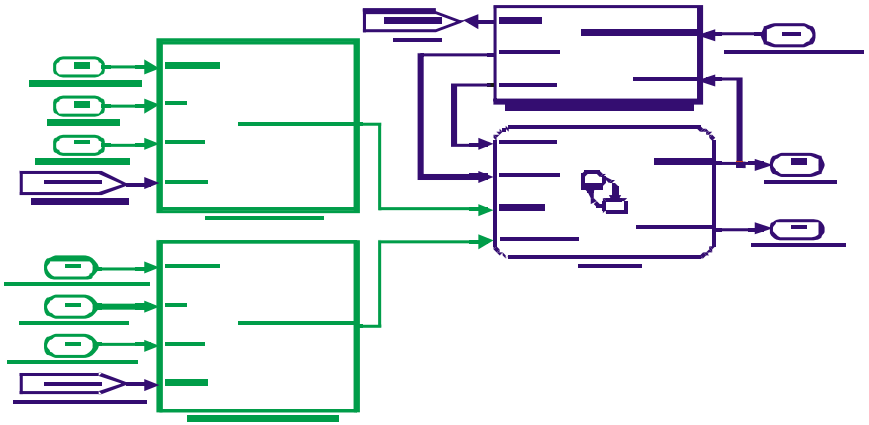
- Control the velocity of window movement
  - Continuous-time model of the physics



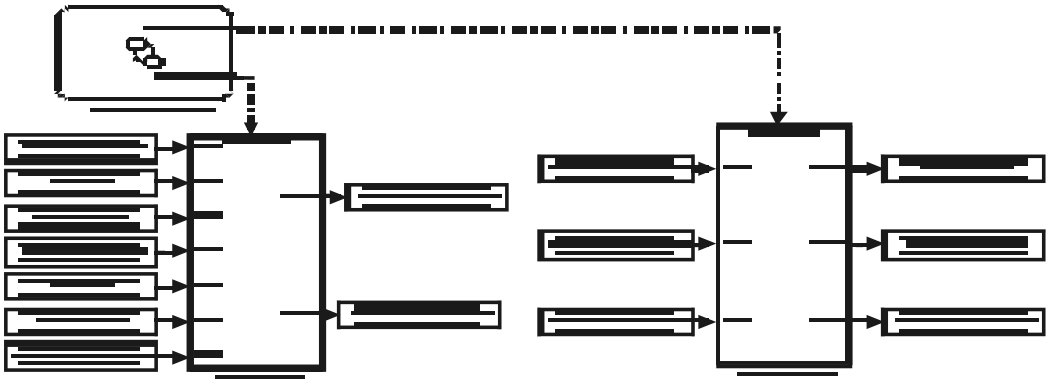
  - Synthesize a controller

Control Implementation

- Assign sample times for discrete-time computer

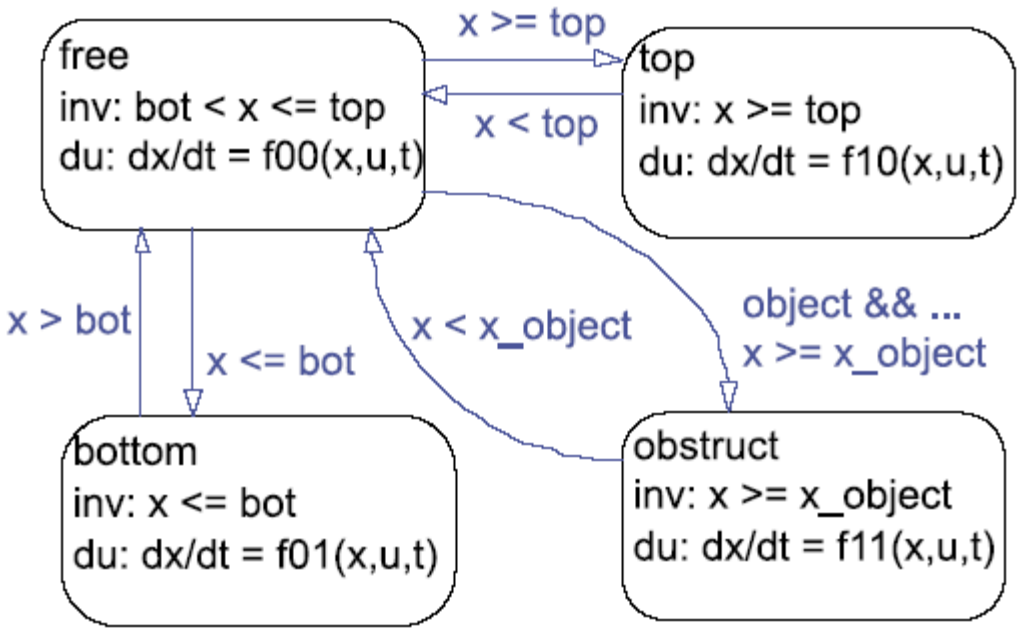


- Assign computations to tasks
  - Include (thread) scheduler



Verify the Design

- Generate Hybrid Automaton
  - Make models explicit



Generate Code

- For example, C code

```
28 /* Function prototypes for chart <S1>/control */
29 static void exit_internal_c2_s2_safe(SFpower_window_con_rtw_c2InstanceStruct
30 *chartInstance);
31 static void
32 exit_internal_c2_s7_driverNeutral(SFpower_window_con_rtw_c2InstanceStruct
33 *chartInstance);
34
35 #define IN_NO_ACTIVE_CHILD (0)
36 #define IN_c2_s1_emergencyDown 1
37 #define IN_c2_s2_safe 2
38 #define IN_c2_s3_driverDown 1
39 #define IN_c2_s7_driverNeutral 2
591 /* Logic: '<S3>/either' */
592 rtb_either = power_window_con_B.passenger_control_b
593 || power_window_con_B.passenger_control_a;
594
595 /* Logic: '<S13>/allow_action' incorporates:
596 * Inport: '<Root>/driver_up'
597 * Logic: '<S13>/overrule'
598 */
599 rtb_temp34 = power_window_con_U.driver_up
600 && (!(rtb_either));
```

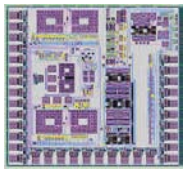
- Many different code formats (‘targets’)
  - Emulate fixed point
  - Real-time
  - Instrumented for debugging
  - Highly optimized

Go (back) to hardware

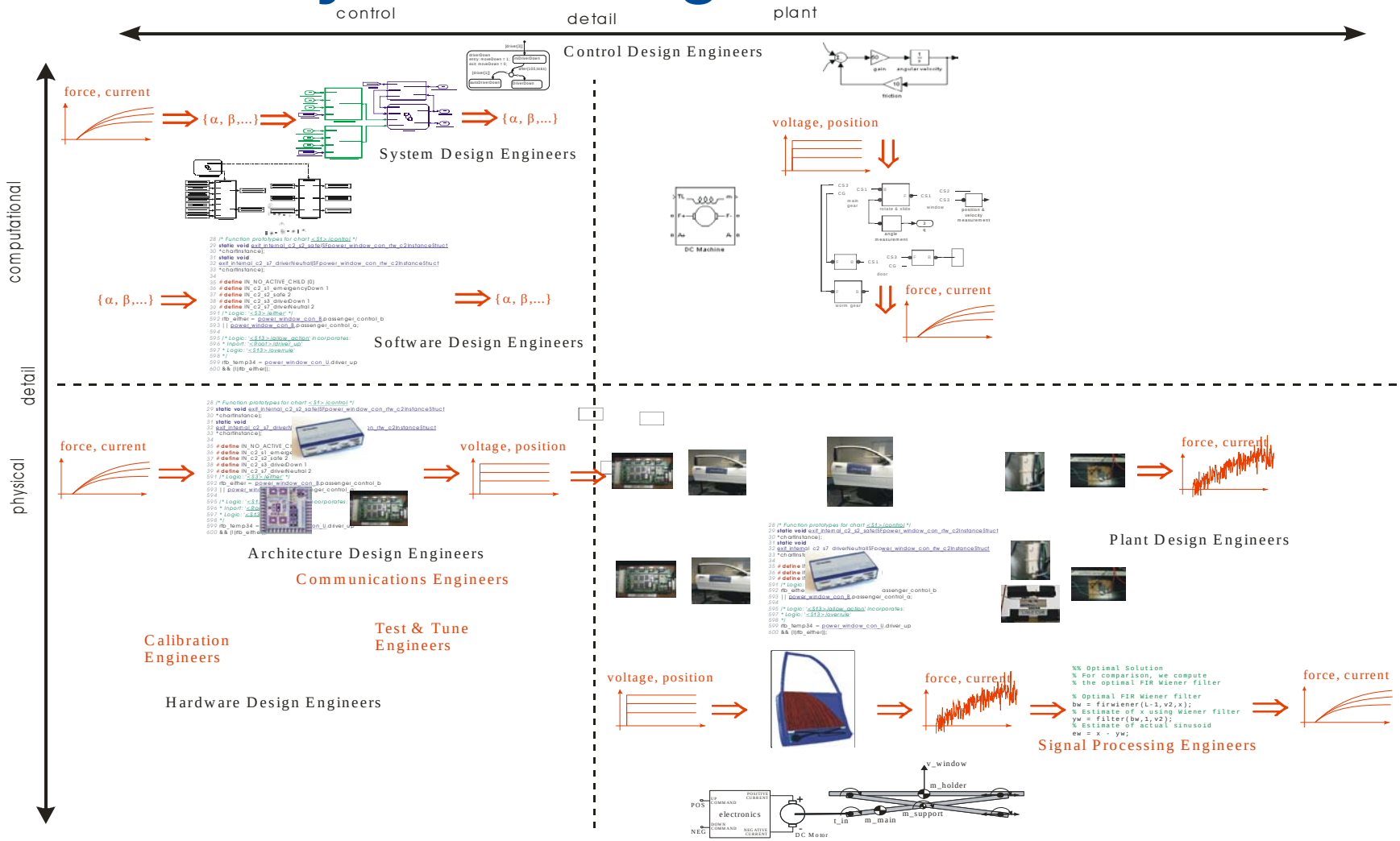
- Different hardware implementations
  - General purpose PC

  - Rapid real-time prototype platform (includes I/O)

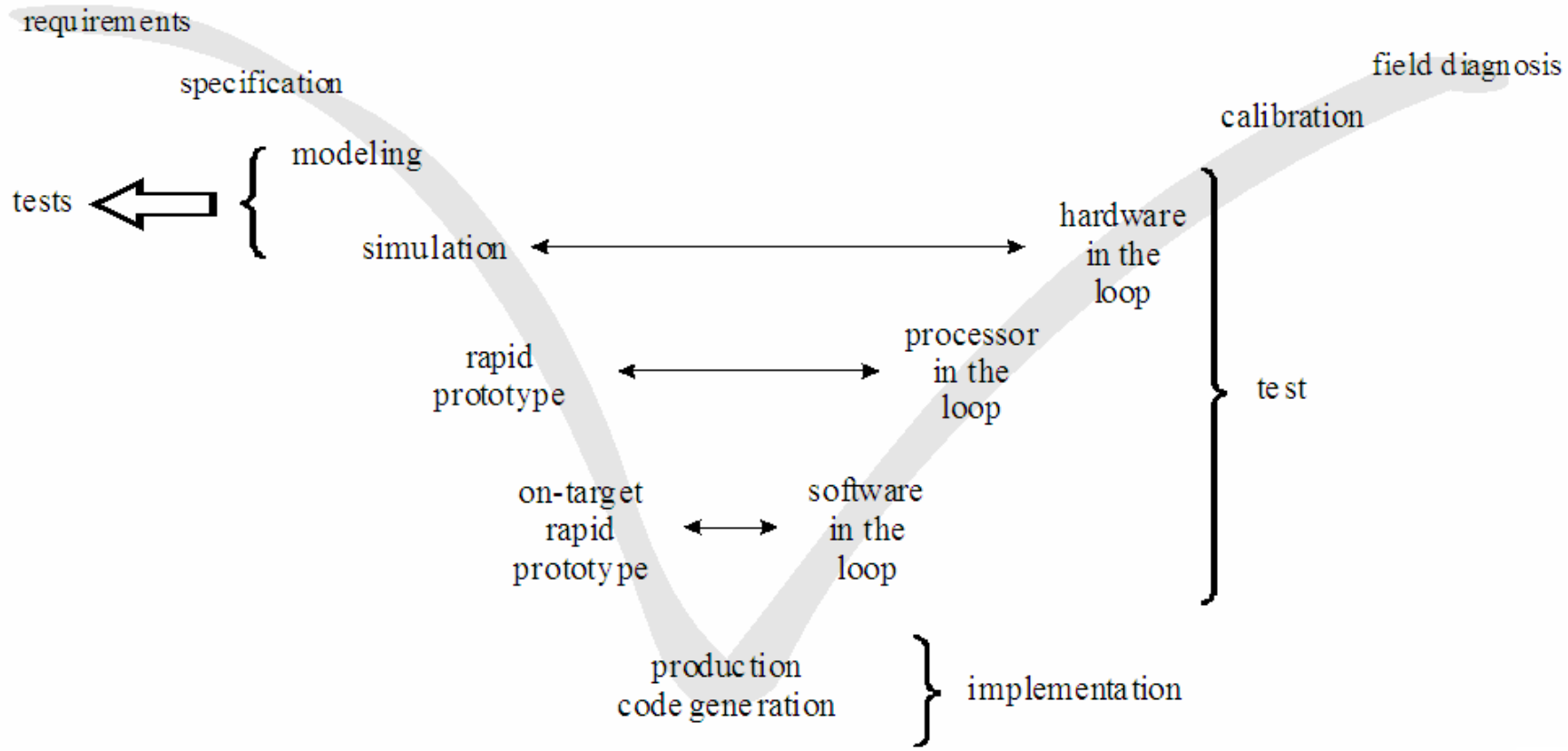
  - Production (fixed point) hardware target



Control System Design Overview



An Industrial Design Process



Pieter J. Mosterman, Sameer Prabhu, and Tom Erkkinen,
"An Industrial Embedded Control System Design Process," in *Proceedings of The Inaugural CDEN Design Conference* (CDEN'04), July 29 - 30, Montreal, Quebec, Canada, 2004.

'Issues'

- Multi-Formalism Questions
  - Is there an underlying **set of semantic notions** onto which a sufficiently broad set of modeling languages used in the control system design process can be mapped?
  - Is there an **API for a general computing machine** that would be sufficient to combine models in different formalisms typically used in control system design.
  - What is the best approach to quickly **generating different modeling formalisms**? Libraries? Meta-modeling? Programming against an API?
  - Is there an **optimal formalism to translate** between modeling formalisms? In particular, what is the preferred formalism to go from controller model to embedded and/or real-time code for different targets?

More 'Issues'

- Industry Related Questions
  - Is there a need for users in industrial control system design to configure and/or restrict usage of modeling languages and how is this best achieved? How can **style guides** be enforced?
  - What needs are there to support **enterprise-wide modeling**? What requirements are sufficient to guarantee **composability of models**? Is composability a requirement?
  - Is there a need to allow users to **configure their tools** specifically for controller design? How should this be supported?

Further 'Issues'

- Model Transformation Questions
  - Is there a way to get to **explicit models** that are used in control law synthesis (e.g., ordinary differential equations, state transition diagrams, hybrid automata) from models that are designed using more intuitive and practicable methods (e.g., implicit differential and algebraic equations, integrated legacy code, guarded equations).
  - Are there methods to generate **models from scenarios**? In other words, can we derive declarative models from axiomatic specifications?
  - Are modern-day **model reduction** techniques sufficient to handle the complexity of industrial models so control **synthesis** methods can be applied?
- Behavior Generation Questions
  - Is **simulation** a sufficiently powerful technology for control system