

Facilities for Testing Control Software

Pieter J. Schoenmakers
<tigr@ics.ele.tue.nl>

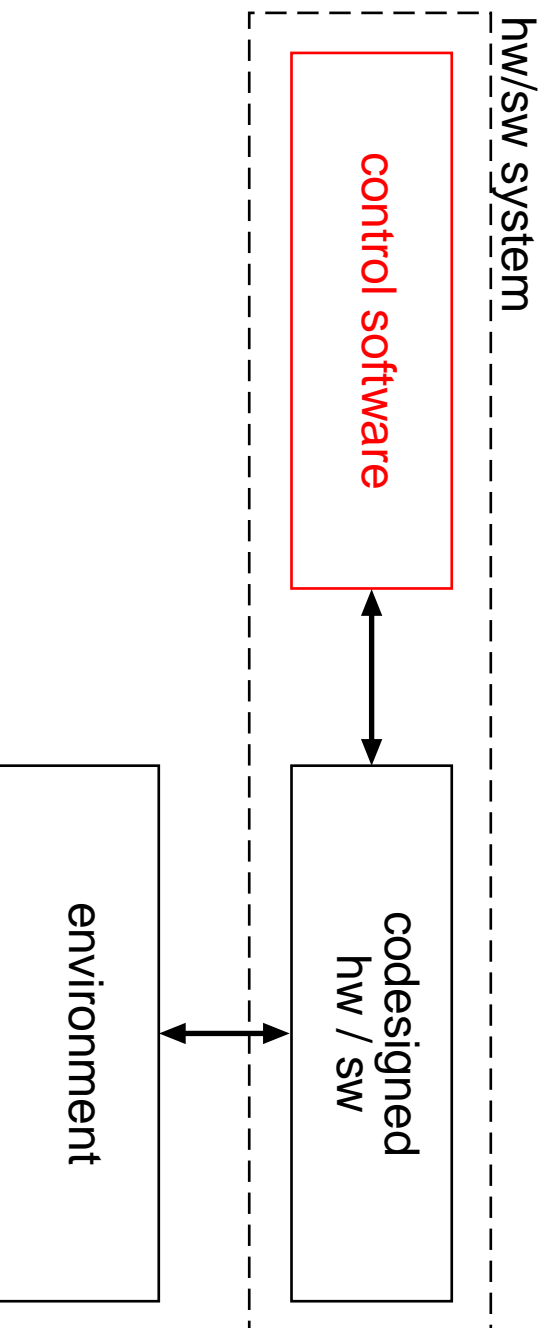
Eindhoven University of Technology

November 1997

Overview

- What is control software?
- What is the problem?
- The architecture
- The testing architecture
- Design for test
- Implementation
- Application
- Conclusion

Control Software



control software is

- high-level
- no stream / data processing
- developed independent of codesign process

Problem Statement

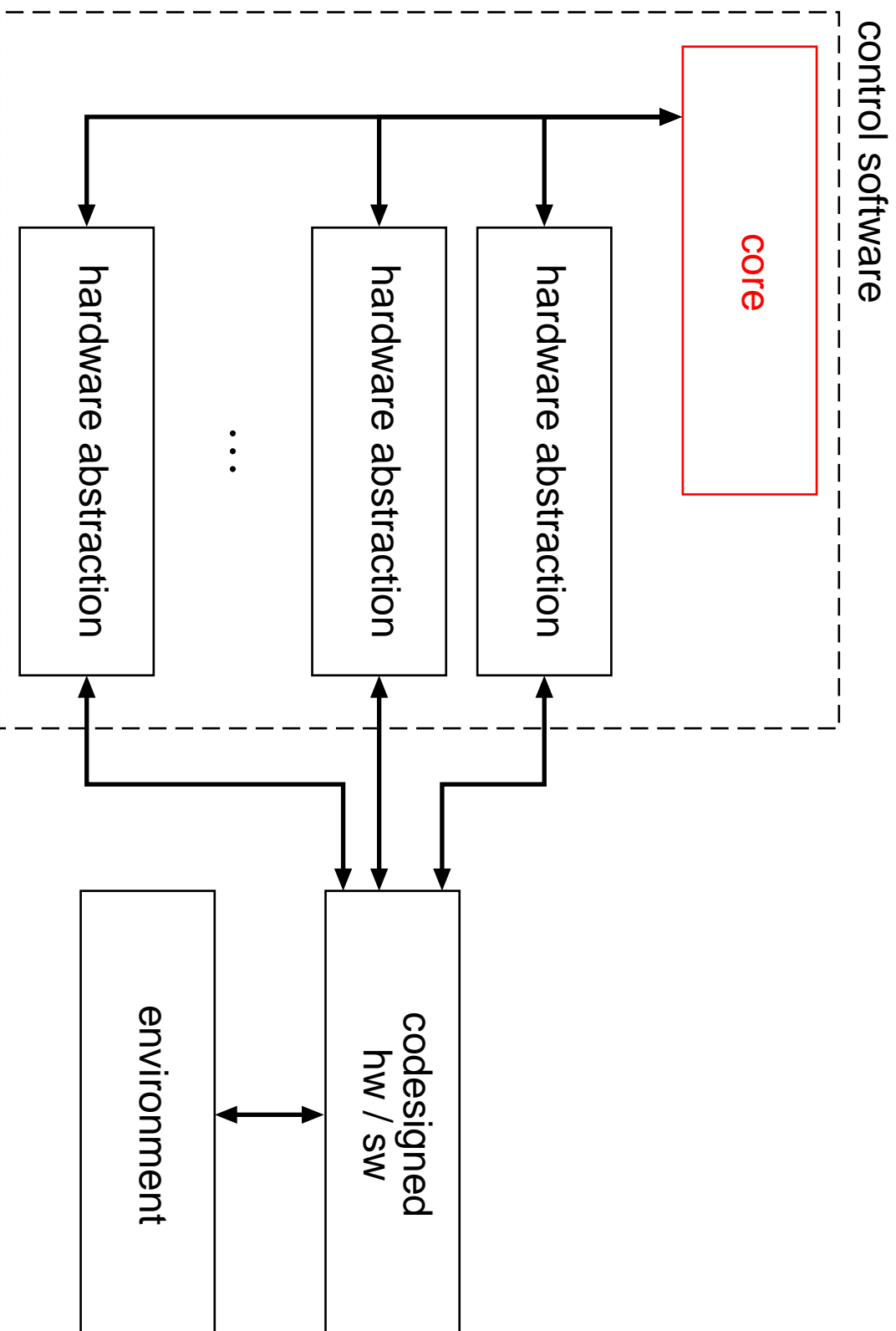
How to test control software?

- at (sub-) system level
- prior to hardware availability
- unattended
- independent of hardware / availability

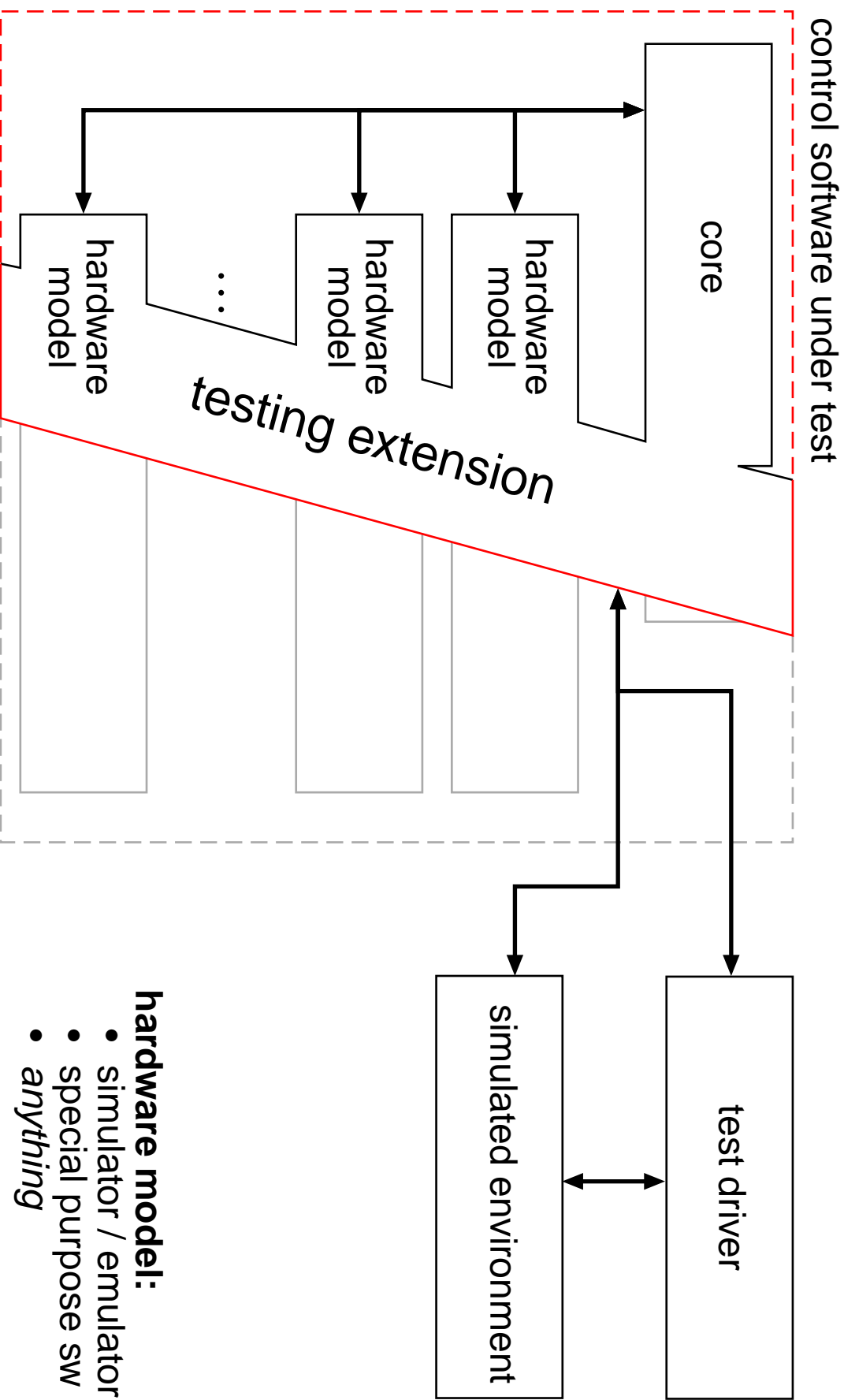
Generalizing

- how to test software that interacts with hardware
 - independent of hardware
 - unattended
- convert control software into 'normal' software
 - normal testing methods apply
 - normal tools can be used

The Architecture



The Testing Architecture



- hardware model:**
- simulator / emulator
 - special purpose sw
 - *anything*

Design for Test

Hardware

- test: the design
a particular instance
- testing facilities: part of design

Software

- test: the design
- testing facilities: *not* part of design

Implementation

premise: object oriented programming language

mechanism: ability to amend objects' behavior

- modify / add behavior replace / add methods
- support complex behavior add instance variables
- no recompilation

facilities: driver and framework

- test driver
- extensible simulated environment
- testing extensions framework

Application

proof of concept: elevator control

- service s systems of f floors using e elevators
- simulated environment
 - cage movement (constant speed, infinite acceleration)
 - inhabited by p persons
 - push buttons
 - observe an elevator or floor
- test execution
 - scenario generation (no record or playback)
 - outcome: conditions, invariants

Conclusion

method

- testing software that normally interacts with hardware
- independent of the hardware
 - usual sw testing methods apply
 - independent of source availability
- testing extensions are not part of the design

models

- hardware models
 - flexible
 - simple models suffice for high abstraction testing
- testing is only as good as the models
- models need to be tested as well