Time, Events and Components in Automotive Embedded Control

Karl-Erik Årzén
Lund University
Sweden

LUND INSTITUTE OF TECHNOLOGY
Lund University

ARTIST2

Beyond Autosar, Innsbruck 23-24 March, 2006

Outline

- Trends in Automotive Systems and Consequences for Automotive Control
- Controller Timing
- Analysis Tools
  - Jitter Margin
  - Jitterbug
  - TrueTime
- Controller Components

Disclaimer

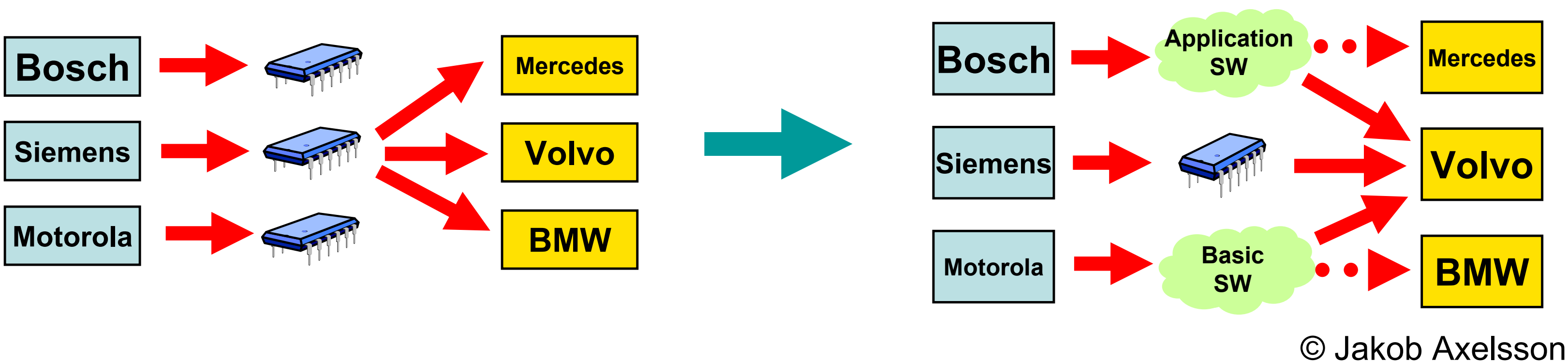
- At several places I will refer to Autosar.
- Autosar = AUTomotive ”Open” System Architecture
- Most of the technical documents are confidential!!
- Hence, my knowledge is only second-hand

The Role of Control

- Advanced control is absolutely essential in modern cars
  - Powertrain, emissions, vehicle dynamics, safety systems, …
  - ECU rather than CPU
- Control gives performance, safety, and low emissions
- The quality and performance of the control systems must be a top priority

Automotive Trends

- From federated to integrated systems

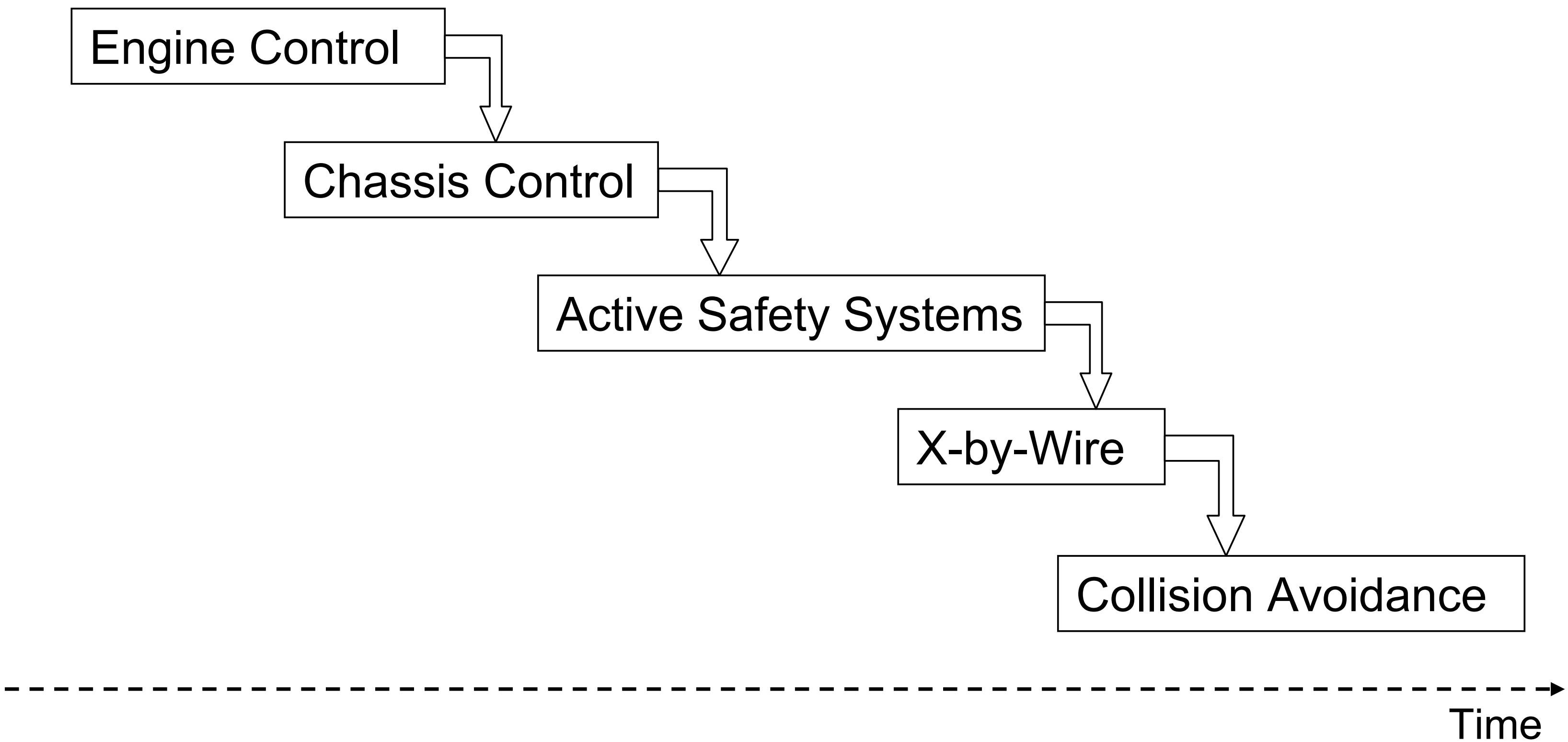


- One system and supplier / ECU

- Several systems / ECU
- Automotive manufacturers become HW / SW integrators

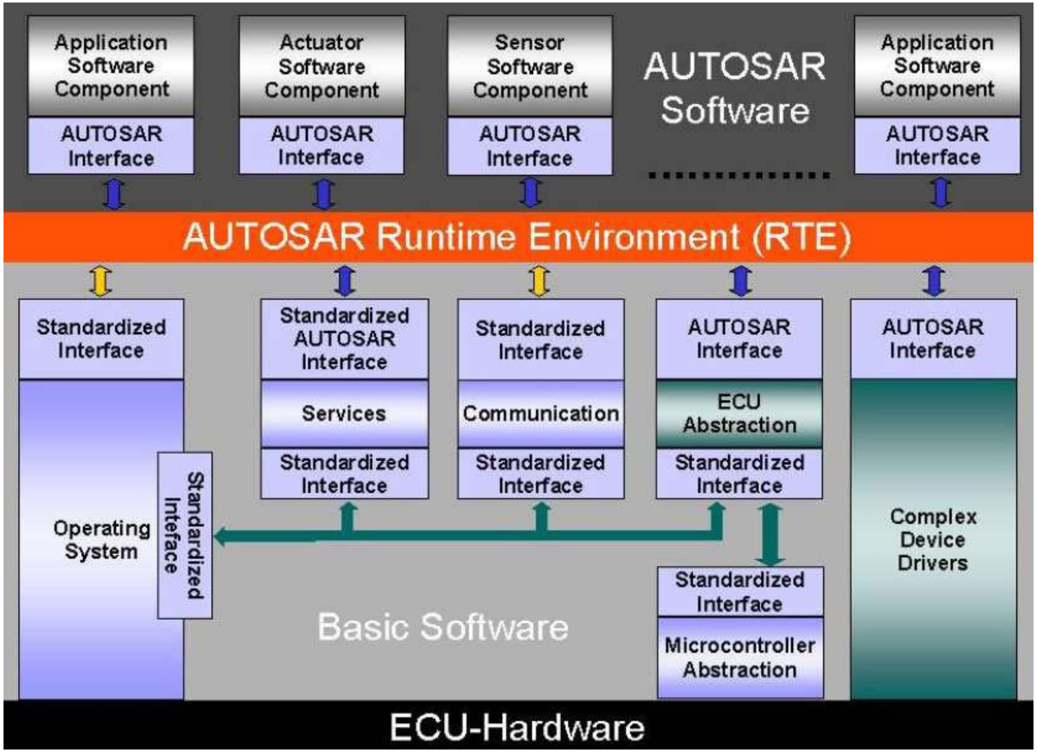
Automotive Trends

- Increased functionality and complexity



Automotive Trends

- Standarized architectures and support for reuse
  - Autosar
  - Component technology

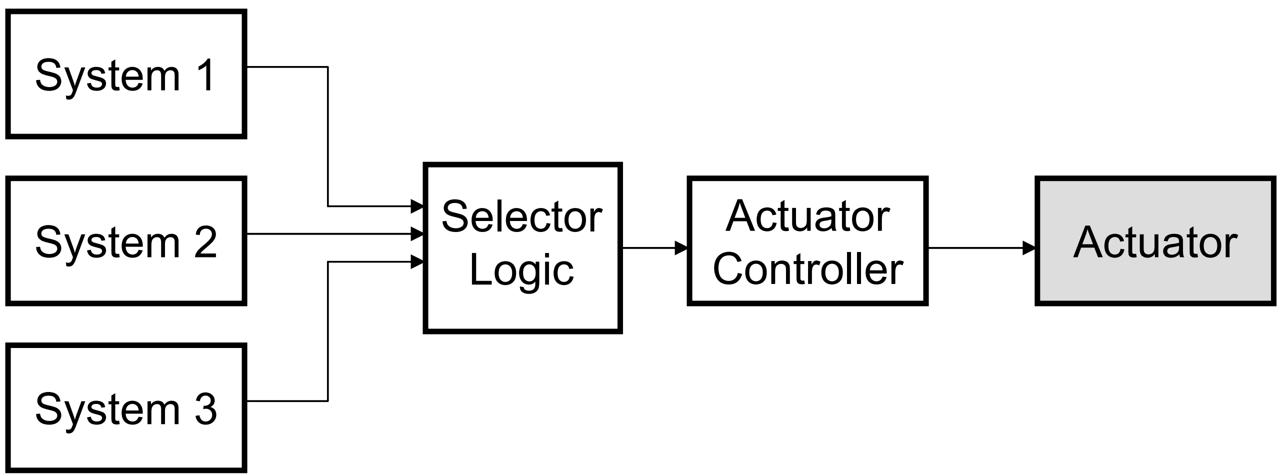


Consequences for Control

- A sensor will be used by several systems
  – part of the vehicle platform, or
  – part of one system but made available to other systems, possibly using middleware techniques
- Sensor components will be special

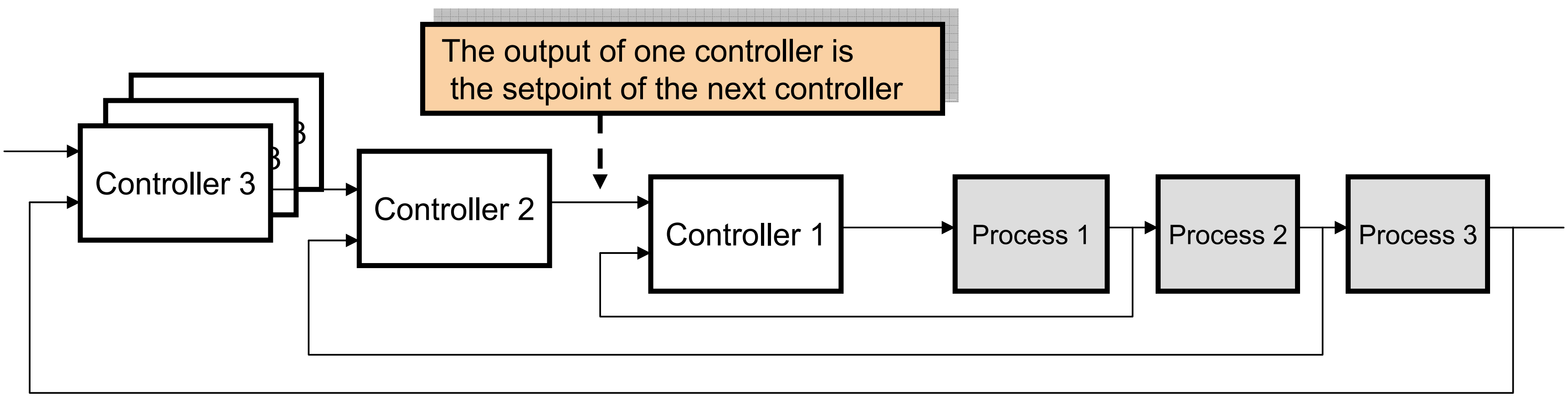
Consequences for Control

- The same actuator will be used by several systems
  - Brakes will be used by intelligent cruise control, lane following system, collision avoidance system, ESP, ….
- Actuator components will be special



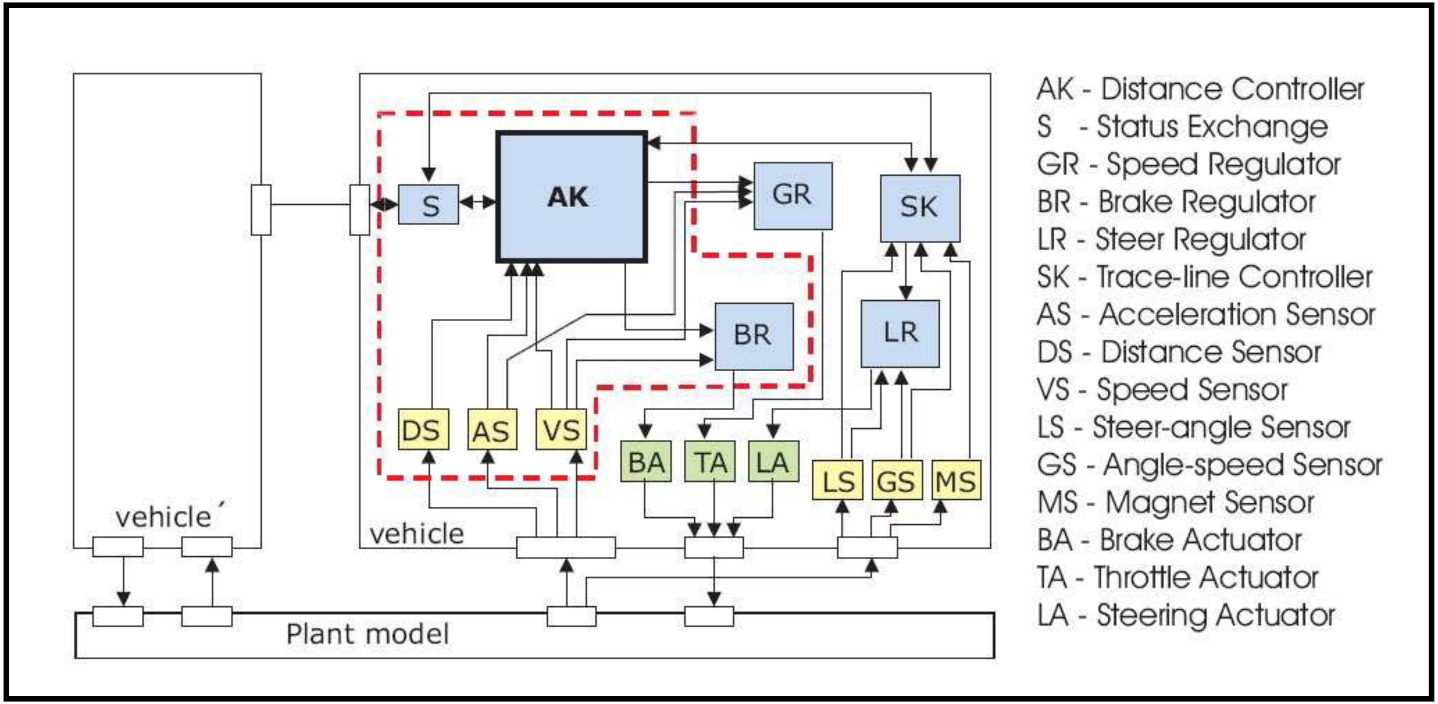
Consequences for Control

- Cascaded control structures will dominate
  - hierarchical, layered
- The different controller components will be part of different systems residing on the same or on separate ECUs



Example

- Platooning (PATH project)



© Werner Damm et al

Outline

- Trends in Automotive Systems and Consequences for Automotive Control
- Controller Timing
- Analysis Tools
  - Jitter Margin
  - Jitterbug
  - TrueTime
- Controller Components

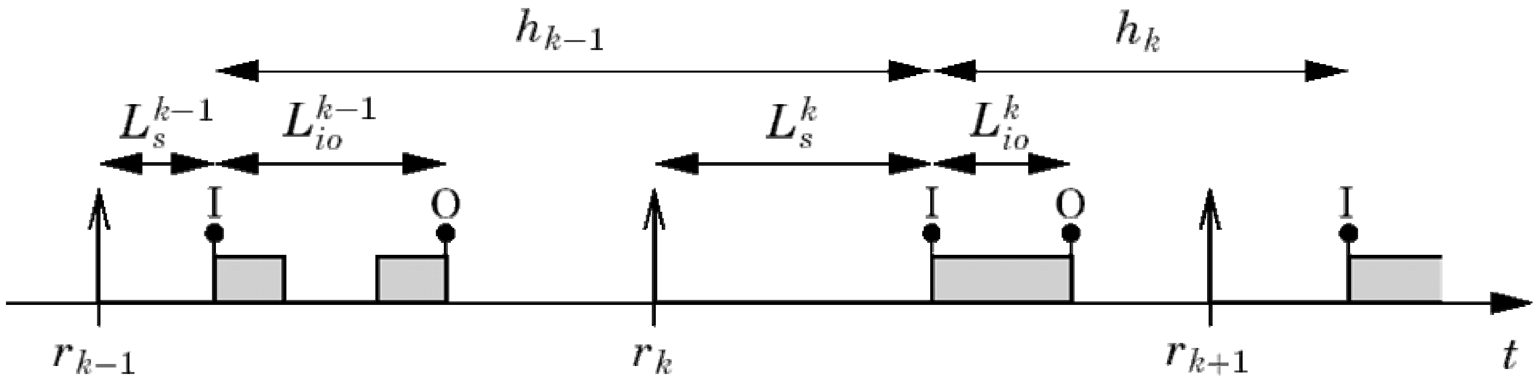
Control Loop Timing

- Classical control assumes deterministic sampling
  - in most cases periodic (not engine control)
  - too long sampling interval or too much jitter cause poor performance or instability
    - but, anomalies exist
- Classical control assumes negligible or constant input-output latencies
  - if the latency is small compared to the sampling interval it can be ignored
  - if the latency is constant it can be included in the control design
  - too long latency or too much jitter cause poor performance or instability
    - but, anomalies exist

Networked Embedded Control Timing

- Embedded control systems with limited computing resources may cause temporal non-determinism
  - multiple tasks competing for computing resources
  - preemption by higher-priority tasks, blocking when accessing shared resources, varying computation times, non-deterministic kernel primitives, priority inversion, …
- Networked control systems with limited communication resources may cause temporal non-determinism
  - network interface delay, queuing delay, transmission delay, propagation delay, link layer resending delay, transport layer ACK delay, ...
  - lost packets

Timing Model



- Task released at $r_k = hk$
- Sampling latency L_s
- Sampling jitter $$J_s \stackrel{\text{def}}{=} L_s^{max} - L_s^{min}$$
- Sampling interval jitter $$J_h \stackrel{\text{def}}{=} h^{max} - h^{min}$$
- Input-output latency jitter $$J_{io} \stackrel{\text{def}}{=} L_{io}^{max} - L_{io}^{min}$$

Time-Triggered vs Event-Triggered

- A time-triggered approach with a global clock maximizes the temporal determinism
  – time-triggered computations
  – time-triggered communication
- The time-triggered approach also has other advantages
  – e.g. fault handling

Time-Triggered vs Event-Triggered

- However, maximizing the temporal determinism may degrade control performance
- The time-triggered approach has disadvantages
  - e.g. inflexibility
- There is no simple answer to the question of whether a time-triggered or an event-triggered approach is best, not even if one only considers control performance

Latency vs Jitter

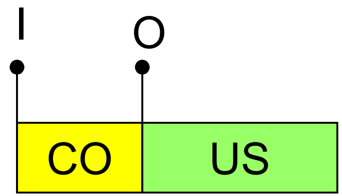
- Both input-output latency and jitter typically degrade control performance
- Jitter can be removed by buffering → longer latency
  - time-triggered approach
- It is easier to compensate for constant than random delays

Which is worse – latency or jitter?

Reducing Latency

- Minimize the interval between sampling and output
- Split up code in two parts: CalculateOutput and UpdateState

```
y = ADin();
u = CalculateOutput(y,yref);
DAout(u);
UpdateState(y,yref);
```



Reducing Latency

- General linear controller

```
x(k+1) = F x(k) + G y(k) + H r(k)
u(k) = C x(k) + D y(k) + E r(k)
```

- Code structure

```
ADin;
u := u1 + D*y + E*r;  // CalculateOutput
DAout(u);
x := F*x + G*y + H*r; // UpdateState
u1 := C*x;            // Precalculate
```

Outline

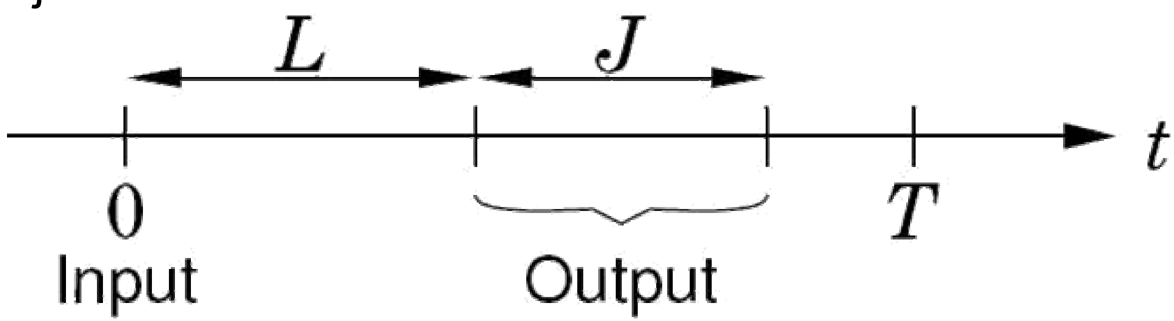
- Trends in Automotive Systems and Consequences for Automotive Control
- Controller Timing
- Analysis Tools
  - Jitter Margin
  - Jitterbug
  - TrueTime
- Controller Components

Jitter Margin

- A measure of how much time-varying input-output latency a control loop can tolerate before becoming unstable
- Extension of the phase margin / delay margin for constant latencies
- Defined by Anton Cervin based on results by Lincoln & Kao

Jitter Margin

- Assumptions:
  - periodic sampling (high prio/interrupt-driven)
  - arbitrarily time-varying latency
    - - constant part
    - - jitter



- Jitter margin $J_m(L)$: the largest J for which stability can be guaranteed given a value of L

Jitter Margin

- Based on small-gain theorem
  - sufficient only
  - not very conservative
  - only linear systems
- Graphical frequency domain test

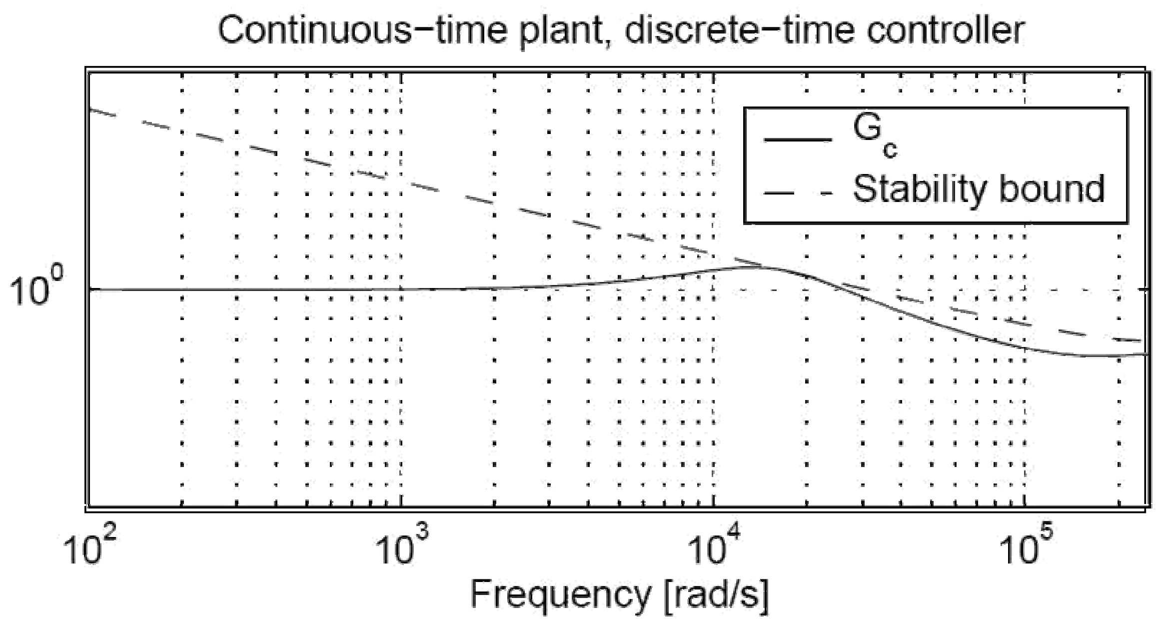
$$\left| \frac{P_{\text{alias}}(\omega) K(e^{i\omega})}{1 + P_{\text{ZOH}}(e^{i\omega}) K(e^{i\omega})} \right| < \frac{1}{\sqrt{J} \left| e^{i\omega} - 1 \right|}, \quad \forall \omega \in [0, \pi]$$

"Closed Loop System (complimentary sensitivity function)"

"Straight Line"

Jitter Margin

Graphical test:

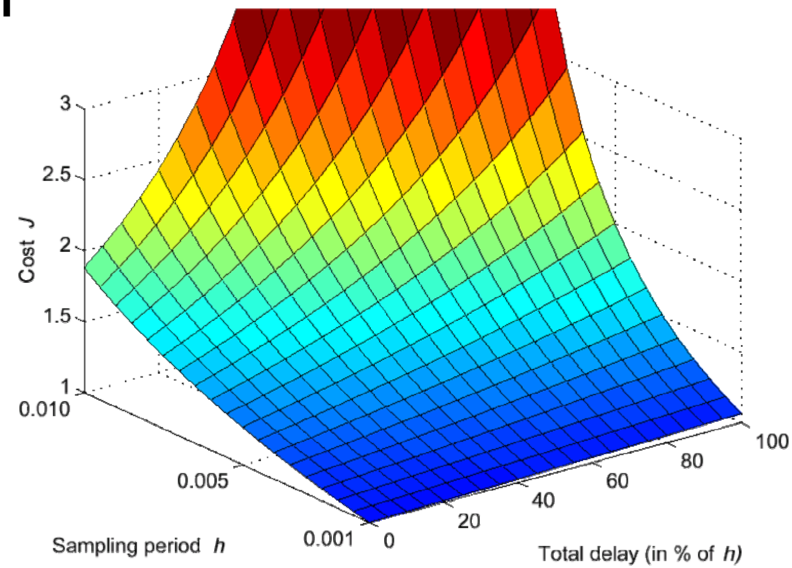


Jitter Margin Usage

- Scheduling
  – assigning realistic task deadlines
- Networking
  – selecting network protocols

Jitterbug

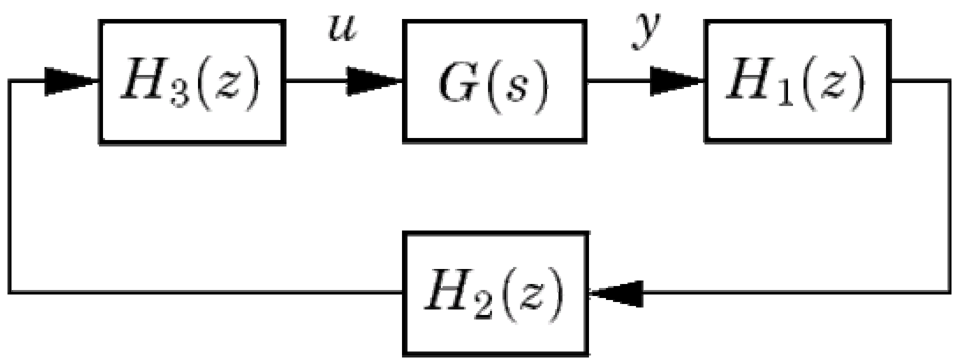
- Matlab-based toolbox for analysis of real-time control performance
- Evaluate effects of latencies, jitter, lost samples, aborted computations, etc on control performance
- Quadratic performance criterion function



Developed by Bo Lincoln and Anton Cervin

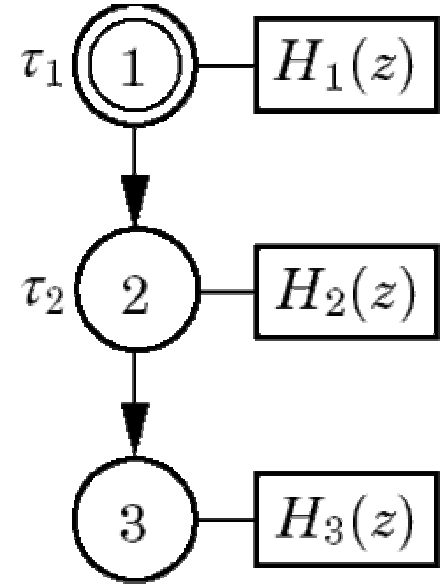
Jitterbug Analysis

- System described using a number of connected continuous-time and discrete-time blocks driven by white noise



Jitterbug Analysis

- The execution of the discrete blocks is described by a stochastic timing model expressed as an automaton
- Time intervals are represented by arbitrary probability density distributions



Jitterbug Performance Analysis

Process: $\boxed{P(s) = \frac{1}{s^2 - 1}}$

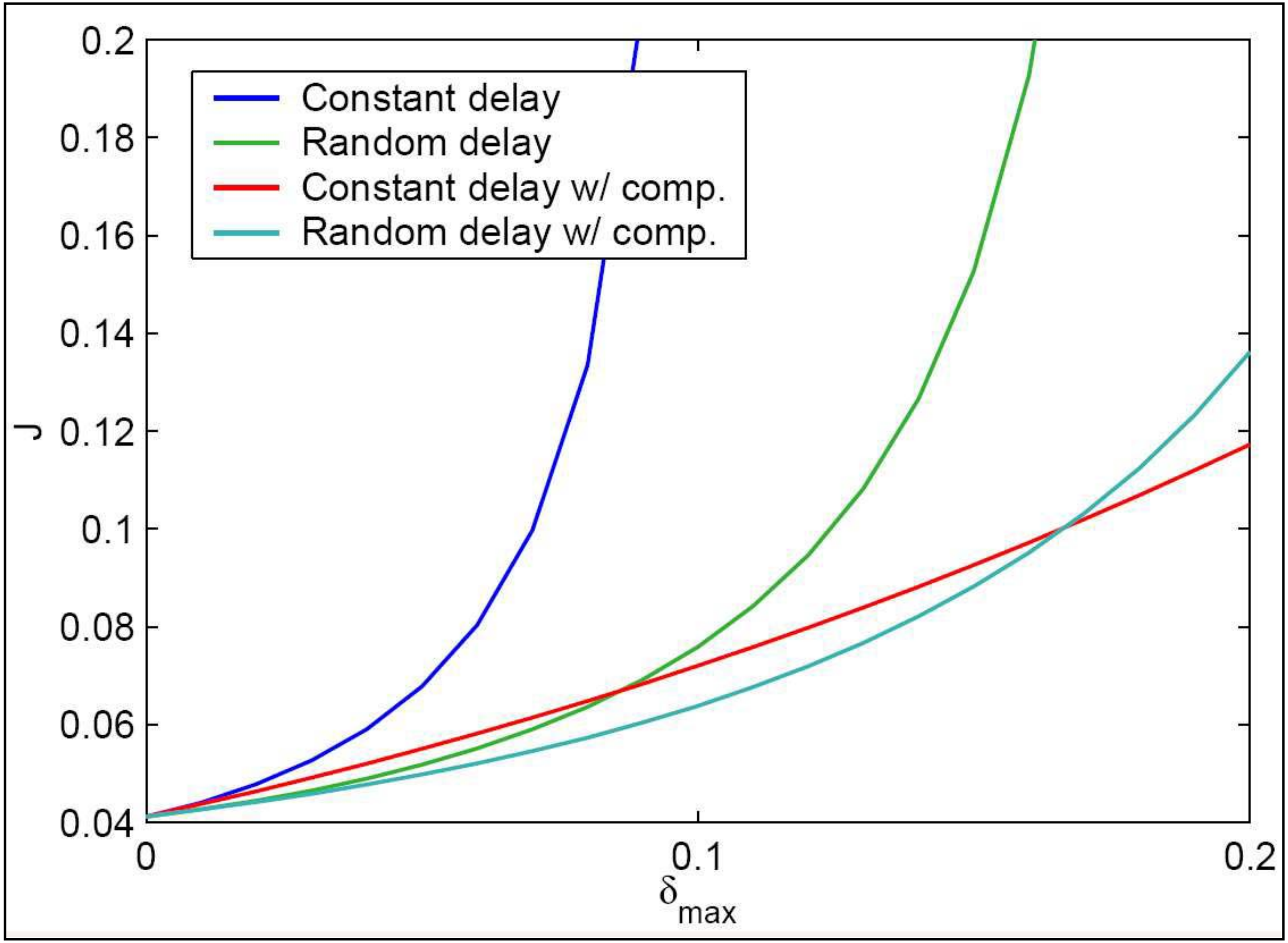
Cost function: $\boxed{J = E(y^2 + 0.001u^2)}$

LQG controller with h = 0.2

Compare four cases:

- Constant latency: $\delta = \delta_{\max}$
- Random latency: $\delta \in U(0, \delta_{\max})$
- Constant delay with latency compensation
- Random latency with average delay compensation

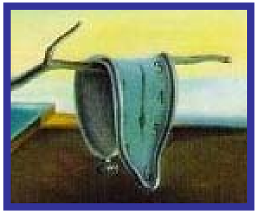
Results



Constant delay
Random delay
Constant delay w/ comp.
Random delay w/ comp.
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
J
0
0.1
0.2
δ_{max}

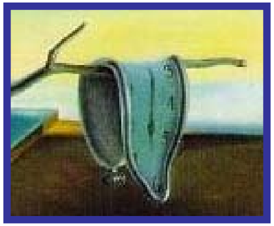
Performance Analysis

- Test batch with 32 processes with different dynamics
- Different scheduling models, including
  - Constant worst-case latency w compensation
  - Random latency w average compensation
- Random latency better in all realistic cases
- Speaks against a time-triggered approach
  - but, the desire for synchronized sampling speaks in favour



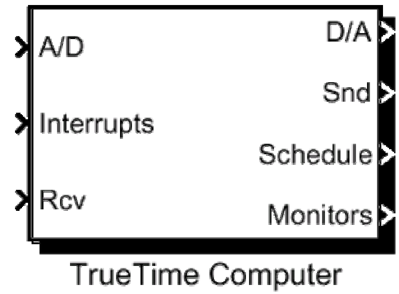
TrueTime

- Simulation of networked control loops under shared computing & communication resources
- Real-time kernels and networks in Matlab/Simulink
- Developed by Anton Cervin, Dan Henriksson, Johan Eker

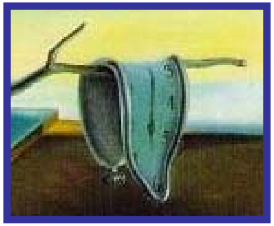


Computer Block

- Simulates an event-based real-time kernel
- Executes user-defined tasks and interrupt handlers
- Coded in Matlab, C++ or Simulink diagrams
- Arbitrary user-defined scheduling policies
- External interrupts and timers
- Support for common real-time primitives

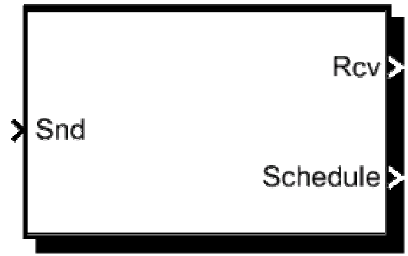


- Fixed priority
- EDF
- Cyclic executive



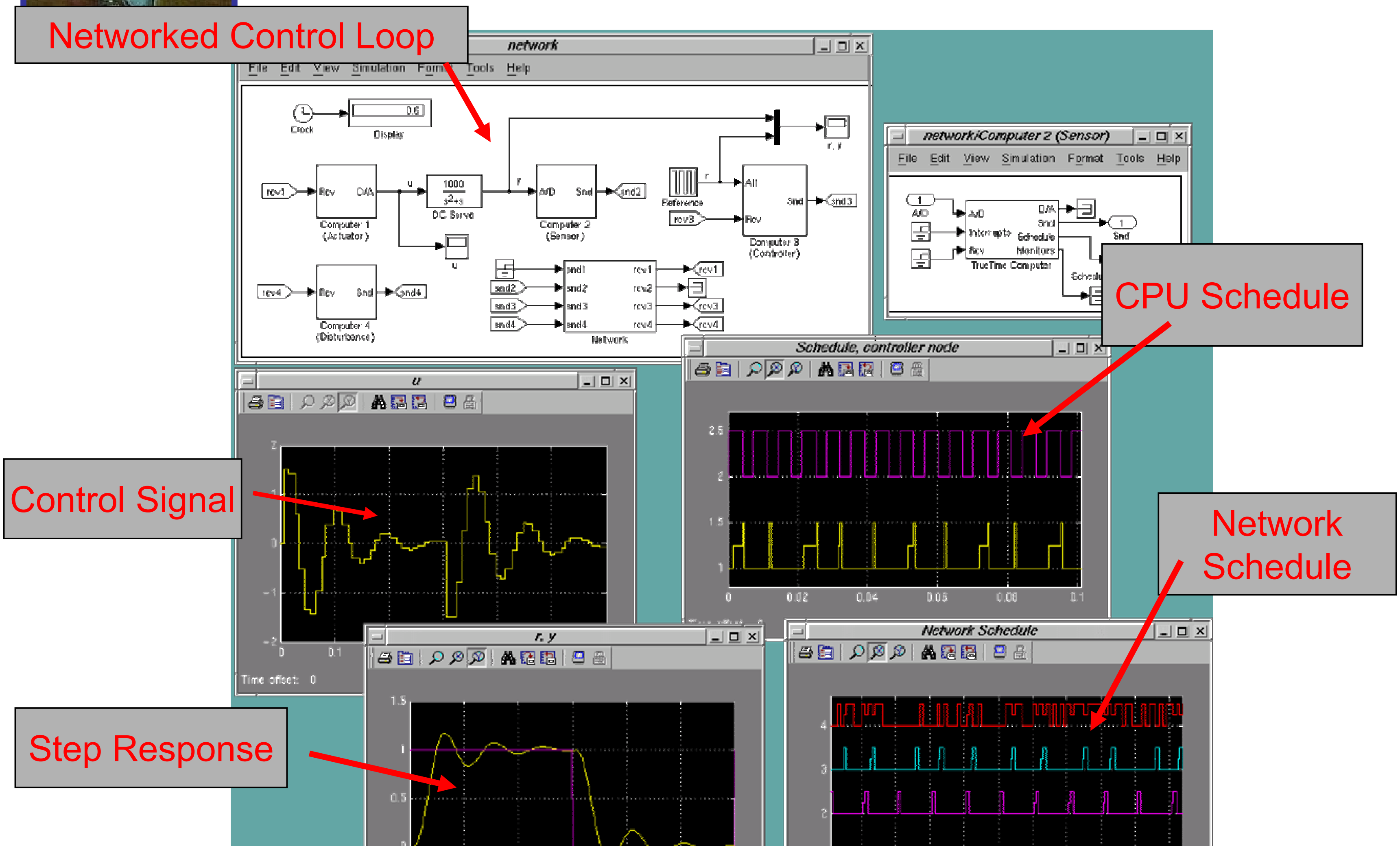
Network Block

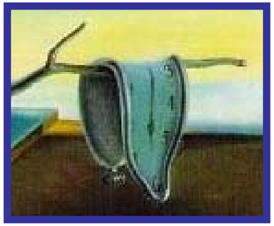
- A variety of pre-defined wired data-link layer protocols
  - CSMA/CD (Shared Ethernet)
  - Switched Ethernet
  - CAN
  - Round Robin
  - FDMA
  - TDMA
- Wireless network
  - WLAN
  - Zigbee



TrueTime Network

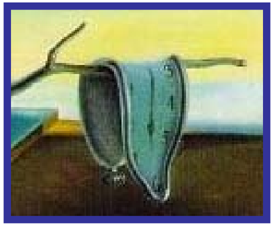
Screen Dump


Networked Control Loop
CPU Schedule
Control Signal
Network Schedule
Step Response
network
network/Computer 2 (Sensor)
Schedule, controller node
Network Schedule
Computer 1 (Actuator)
Computer 2 (Sensor)
Computer 3 (Controller)
Computer 4 (Disturbance)
DC Servo
Network
TrueTime Computer



TrueTime Possibilities

- Co-Simulation of:
  - computations inside the nodes
  - wired/wireless communication between nodes
  - sensor and actuator dynamics
  - mobile robot dynamics
  - dynamics of the environment
  - dynamics of the physical plant under control
  - the batteries in the nodes
  - local clocks with offset and drift
- For
  - embedded control
  - networked embedded control
  - sensor networks
  - mobile robots



Example Users

- Embedded Systems Institute (NL)
  - Integrated simulation of mechanics, electronics and RTOS tasks (VxWorks)
  - Copying machine

    "We found TrueTime to be a great tool for describing the timing behavior in a straightforward way"

- Robert Bosch GmbH
  - Extended the network block with Flexray and TTCAN
- > 1.100 downloads

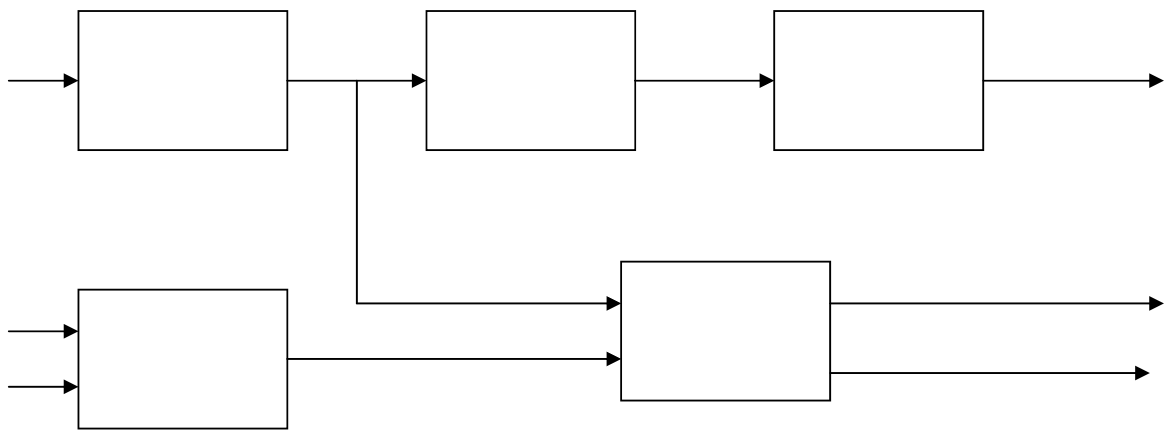
Outline

- Trends in Automotive Systems and Consequences for Automotive Control
- Controller Timing
- Analysis Tools
  - Jitter Margin
  - Jitterbug
  - TrueTime
- Controller Components



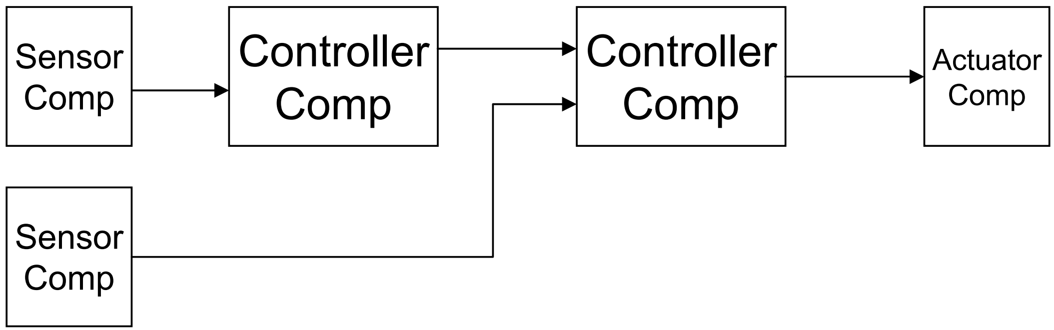
Controller Components

- Component models for embedded systems are often based on the ”pipe and filter” model



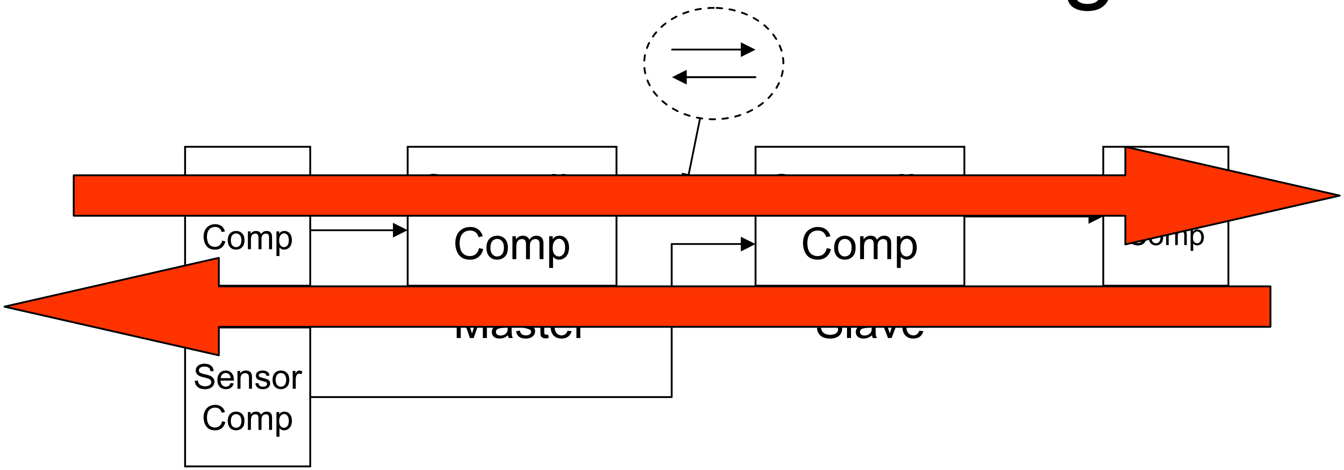
- Components (cp Simulink blocks)
- Logical signal flow
- However, not enough for controller components

Problem: Minimize Latency



- From sensor input to actuator output
- Solution:
  - Execute the CalculateOutput part of all the components according to the logical signal flow
  - Afterwards execute the UpdateState part of the components
- Two scans or sweeps

Problem: Bi-Directional Signal Flow



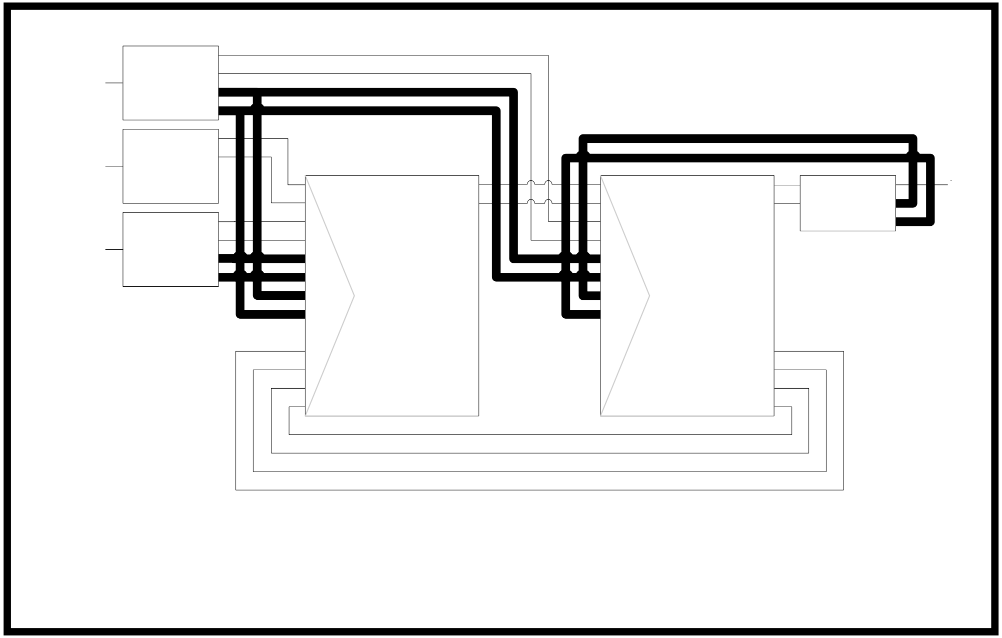
- Signals flow in both directions
- Due to actuator saturation and multiple controller modes the controller state in the master should not be updated until the slave has been updated
  - anti windup and bumpless mode changes
- Solution:
  - Two sweeps:
    - Forward (left to right) – execute CalculateOutput
    - Backward (right to left) – execute UpdateState

Implications

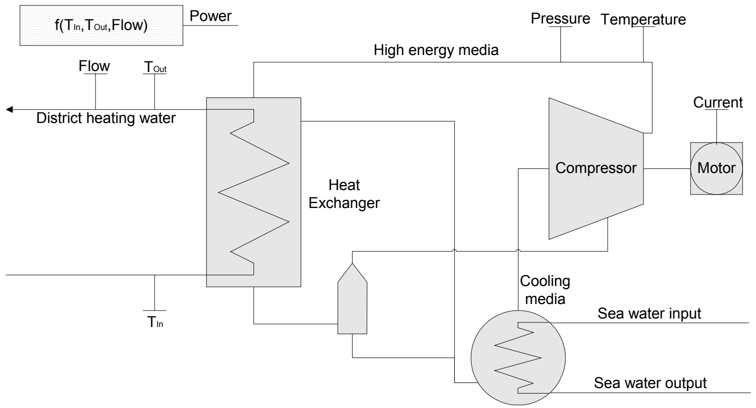
- It is not enough to only standardize on a certain component model
- Also
  - controller component interfaces + semantics
  - the execution structure
- If not
  - degraded control performance
  - reduced possibilities for ”plug and play”
  - software integration and interoperability more difficult
- Well-known in process automation
  - ABB’s control modules (ca 1988)

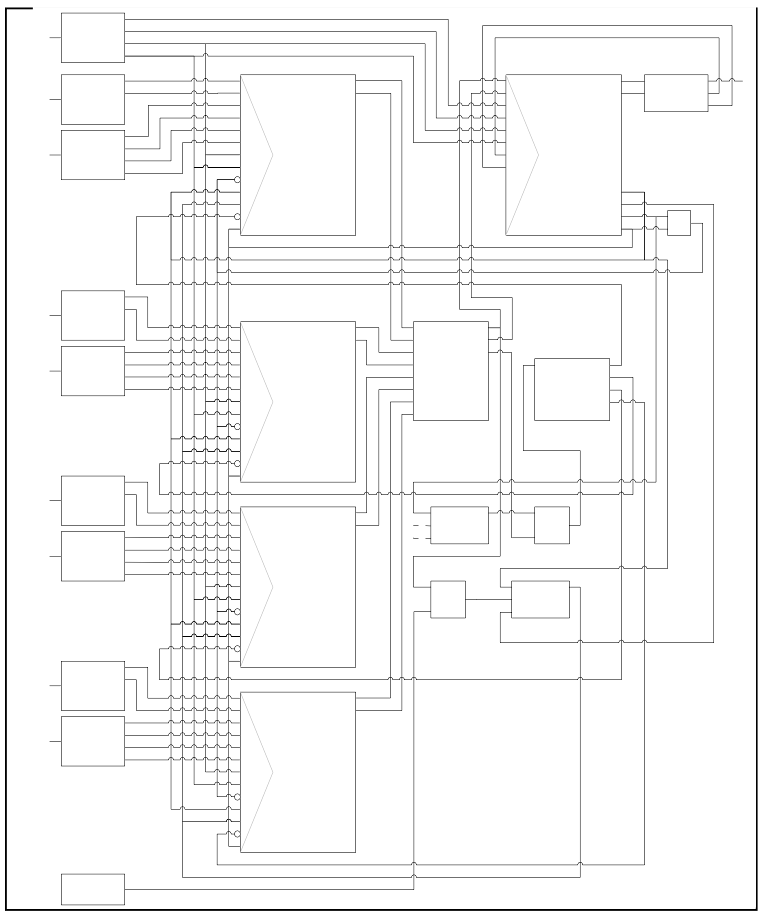
ABB Control Modules

- Main signal flow
  - Integrator anti wind-up
  - Bumpless changes
  - Signal quality
  - Range info



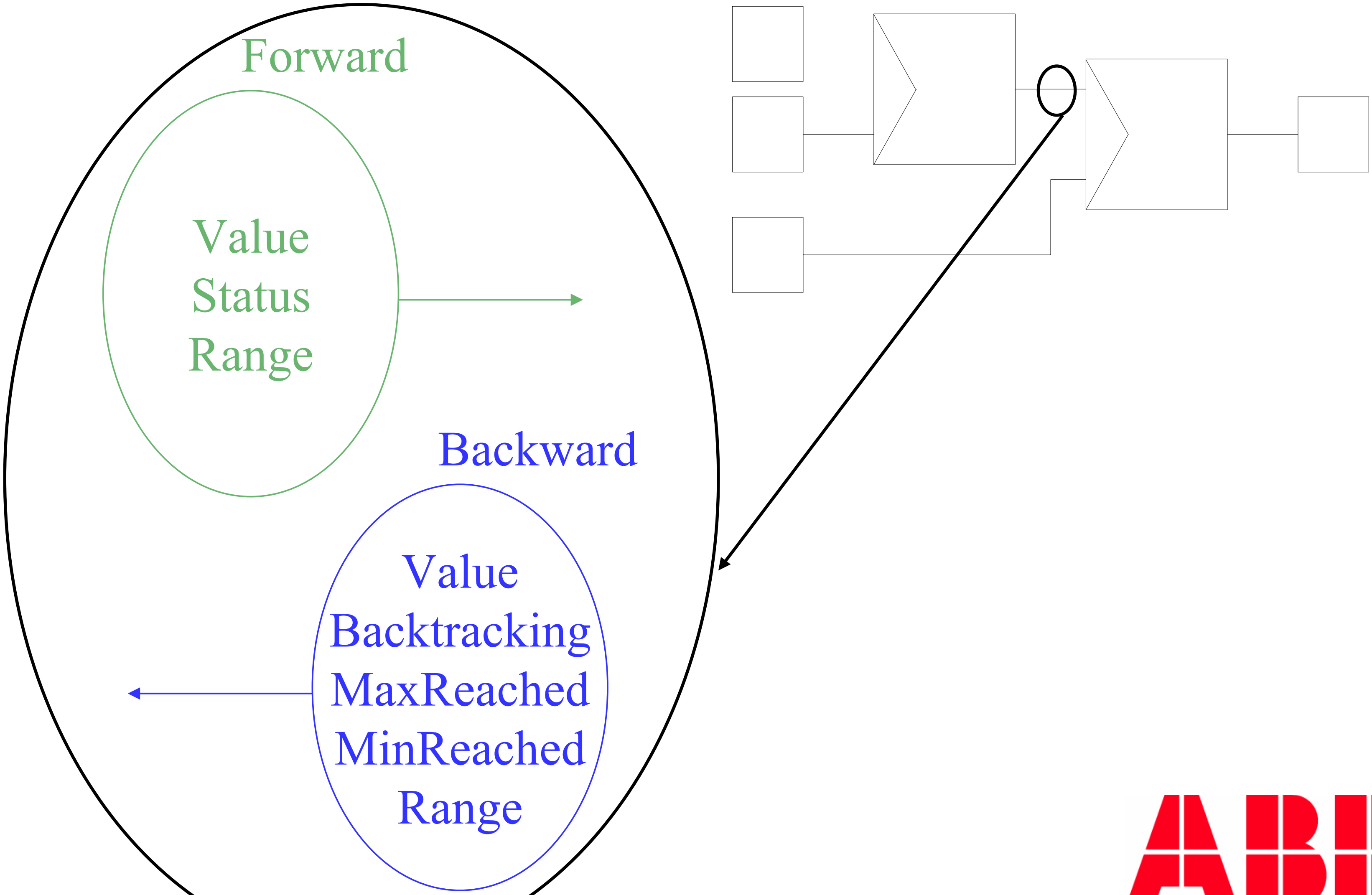
Heat Pump Example



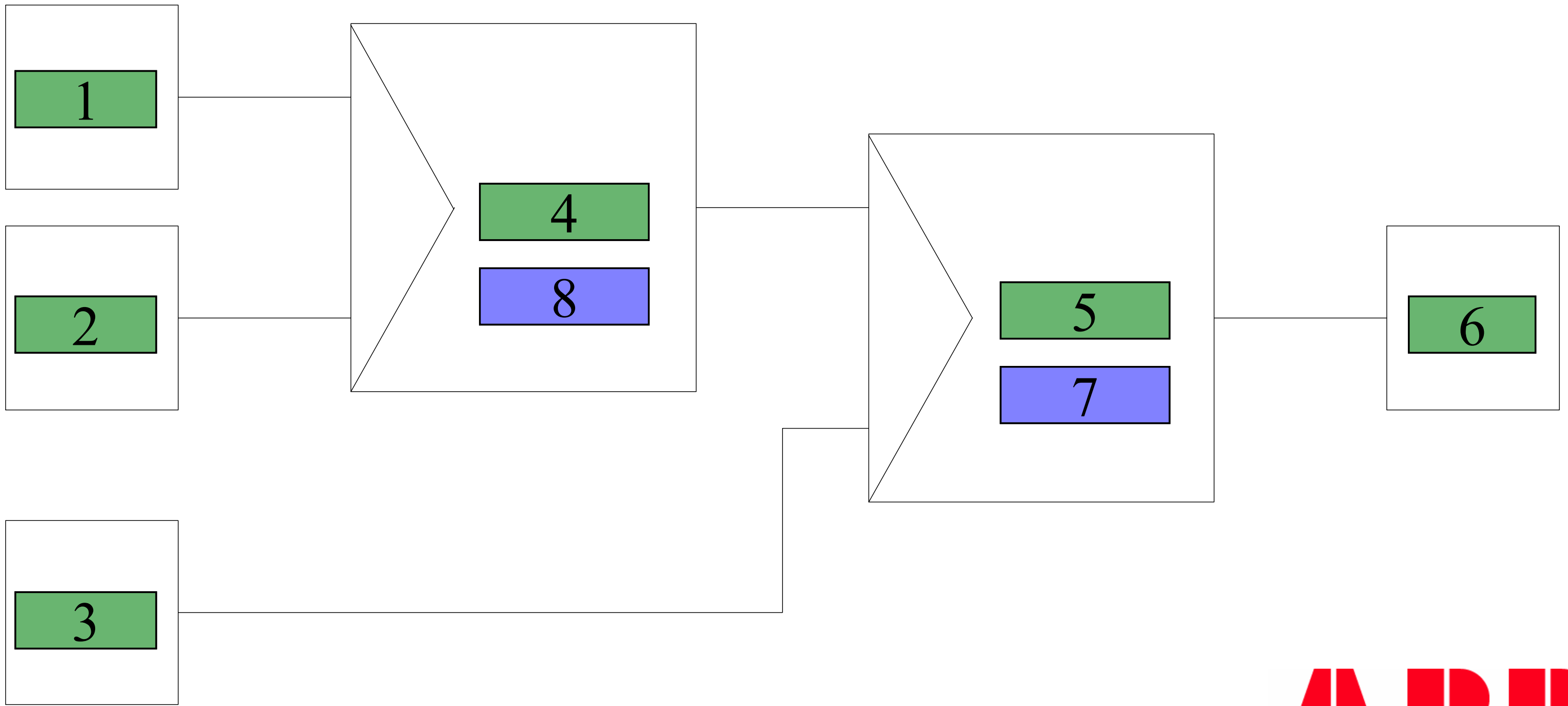


Cascade structure with selector logic

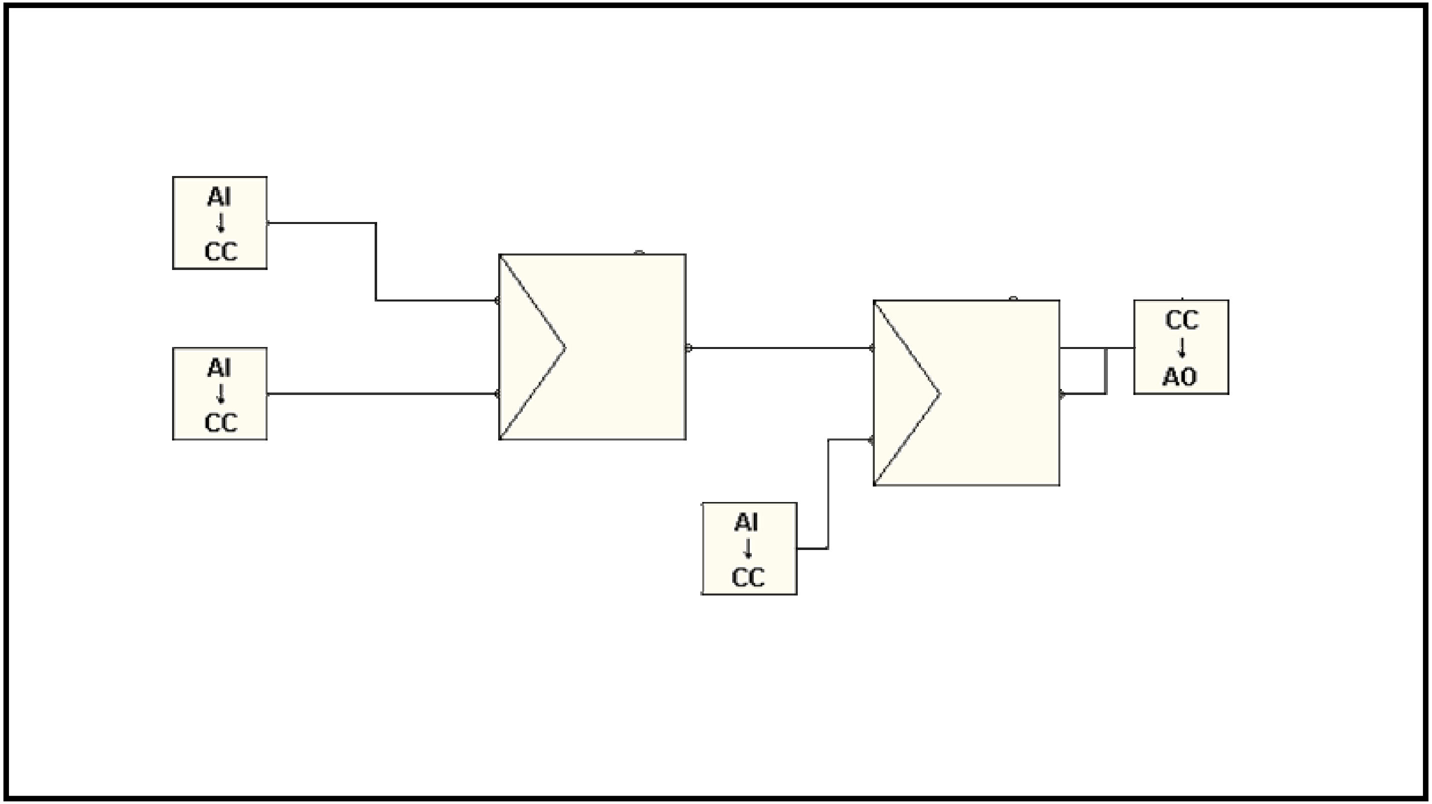
Control Connection



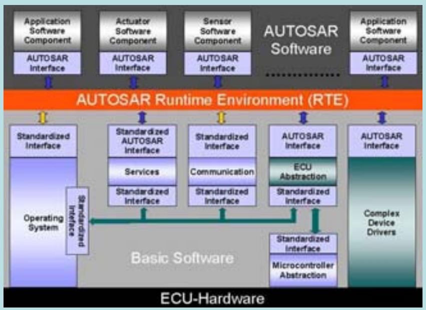
Control Modules - Automatic Sorting of Code

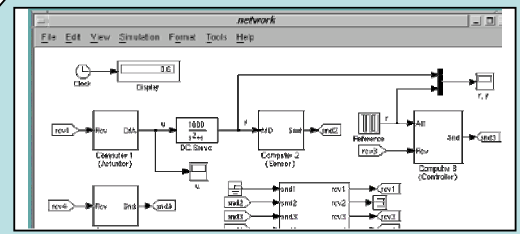


Plug and Play



Automotive Component World





UML2

OMG

Software Community

SysML

MDA

Simulink components

Control Community

Stateflow

Real-Time Workshop

Model-based design

??

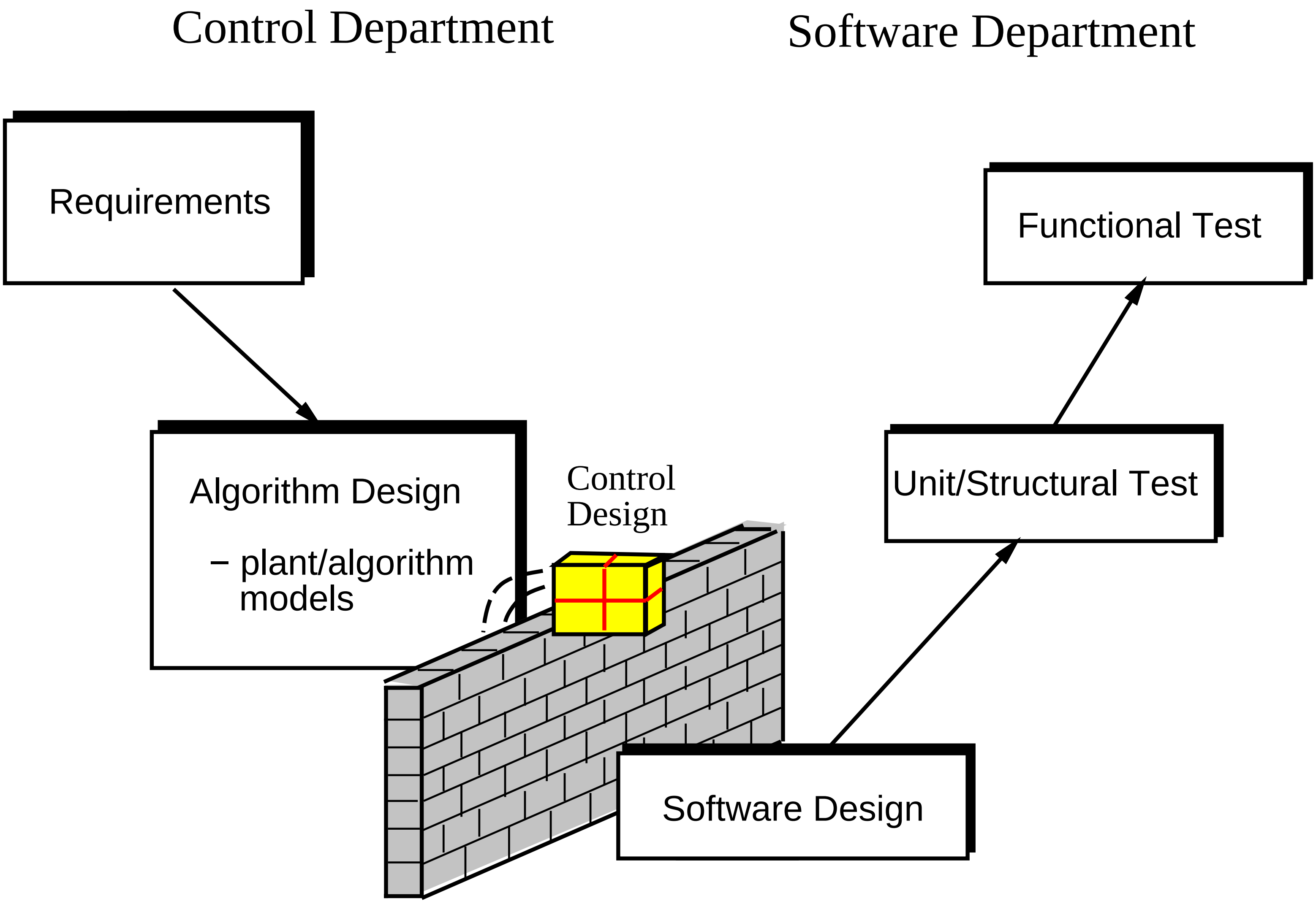
"Autosar"

"Matlab/Simulink"

Beyond Autosar, Innsbruck 23-24 March, 2006

Conclusions

- Time or events is not an easy question to answer
- There can be a tradeoff between temporal determinism and control performance
  - In most cases better with a shorter but varying latency than a longer constant latancy
- Good analysis tools are available
- Reuse and performance puts special requirements on component frameworks for control systems
  - The run-time structure and interfaces must also be standardized


Control Department
Software Department
Requirements
Algorithm Design
– plant/algorithm models
Control Design
Software Design
Unit/Structural Test
Functional Test

End