An experience report on the integration of ECU software using an HSF-enabled real-time kernel

Martijn van den Heuvel – Erik J. Luit – Reinder J. Bril – Johan J. Lukkien – Richard Verhoeven – Mike Holenderski

System Architecture and Networking (SAN)
Department of Mathematics and Computer Science
Eindhoven University of Technology
The Netherlands

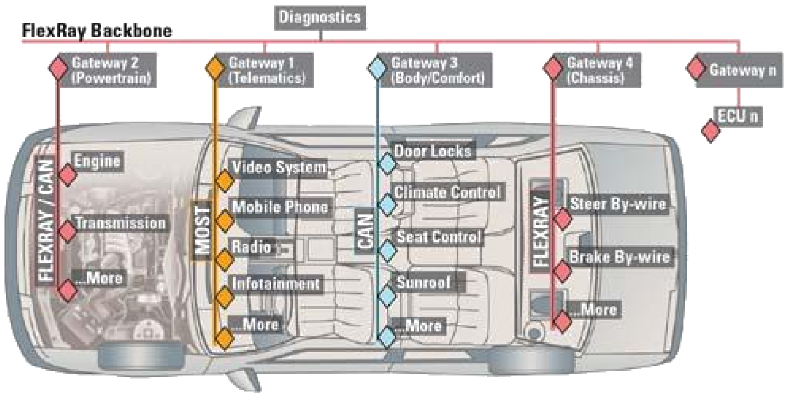
7th July 2015

Technische Universiteit **Eindhoven** University of Technology

SAN

HTAS AUTOMOTIVE INNOVATION

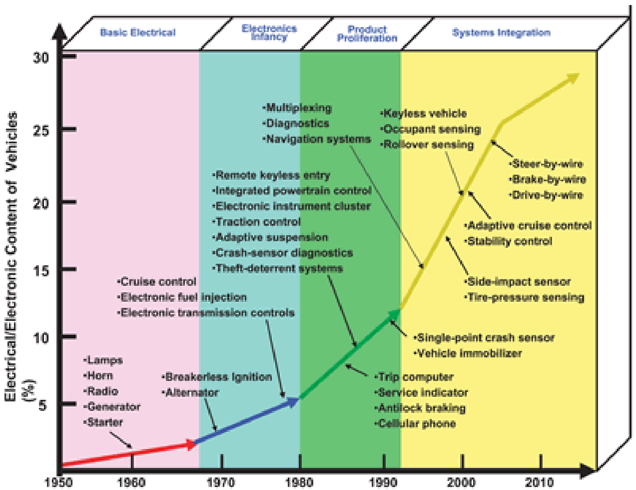
An automotive example



Why this growth of electronic parts?

HTAS AUTOMOTIVE INNOVATION

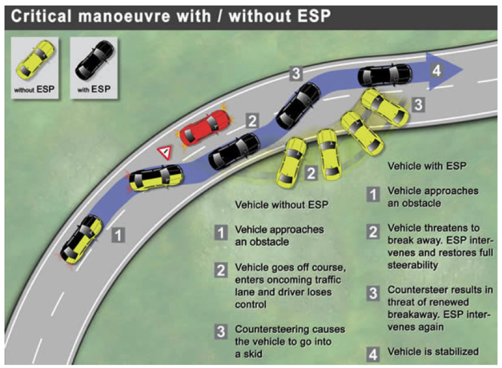
An automotive example: electronic stability program (ESP)

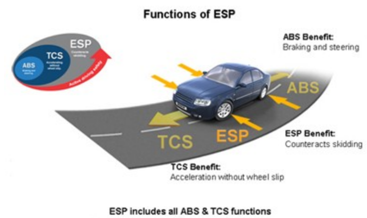


Chip Design Magazine (Jan. 2005)

- Increasing number of applications;
- Extensive networking between them.

An automotive example: electronic stability program (ESP)

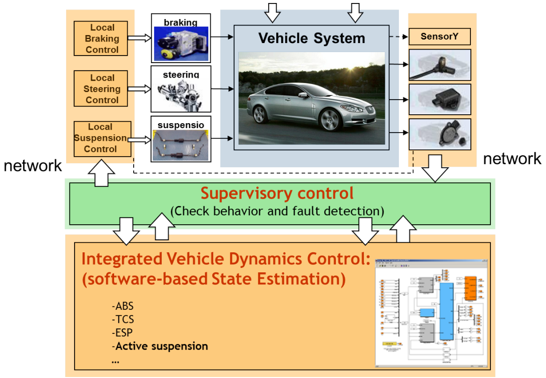




- Increasing number of applications, for example: ABS, TCS, ESP;
- Extensive networking between them.

An automotive example: IVDC advances beyond ESP ...

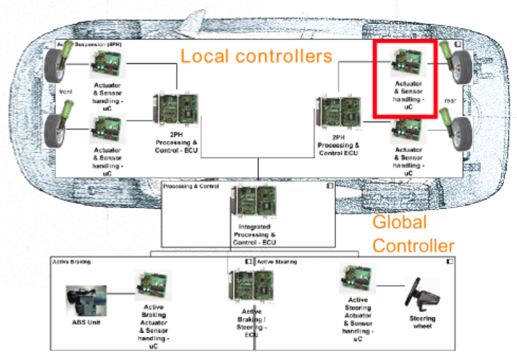
Adding suspension control and software-based vehicle state estimation:

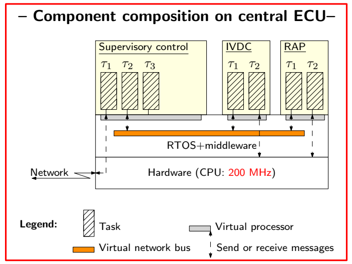


B. Bonsen, R. Mansvelders, and E. Vermeer.
Integrated vehicle dynamics control using state dependent riccati equations.
In *AVEC*, Aug. 2010.

An automotive example: deployment of IVDC advances

- 4X Local controllers for steering, braking, suspension;
- Front and rear IVDC;
- 1X Global IVDC state estimation and supervisory control.





(Semi-)independent developed components by various partners!

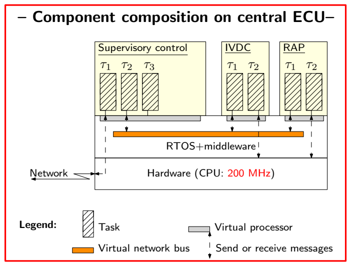
Demo: IVDC with active suspension

Experimental setup:

- 4X Local controllers for active suspension;
- 1X Global IVDC and supervisory control.

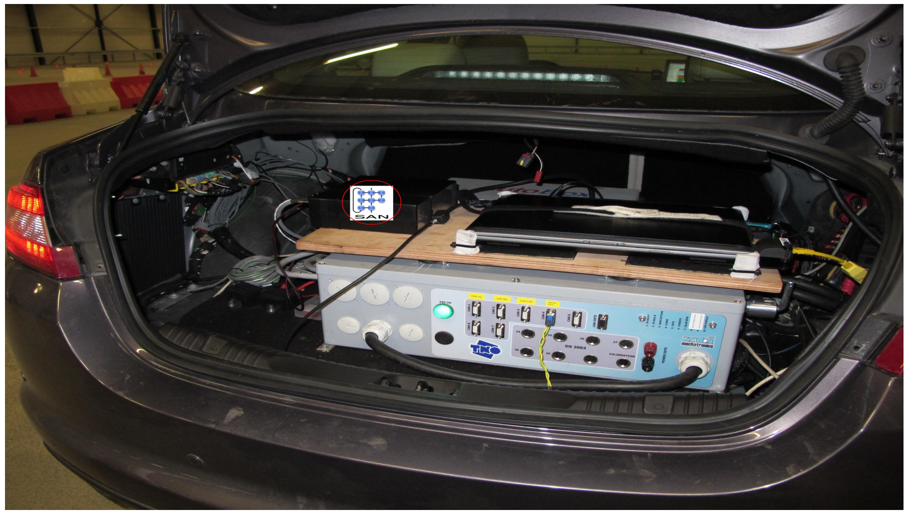
Our contribution:

- Virtualization techniques applied to a local control node:

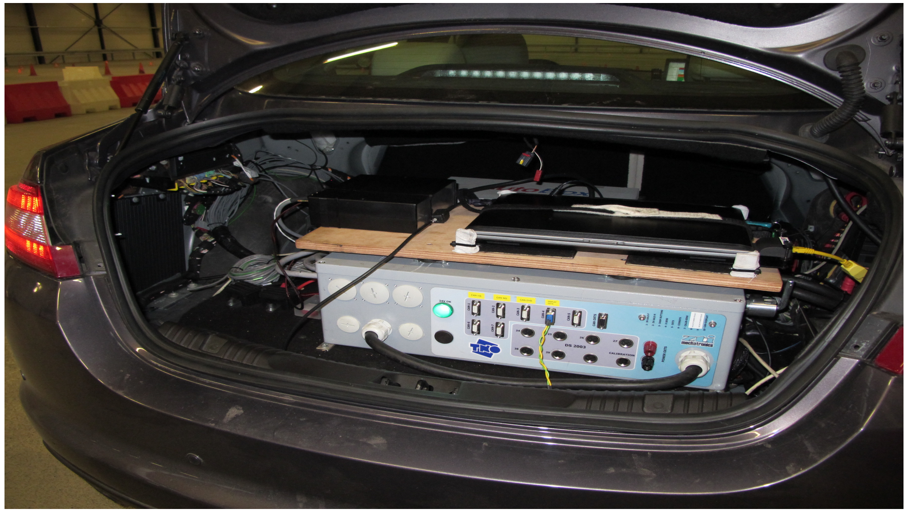


- Communication between independently developed components.

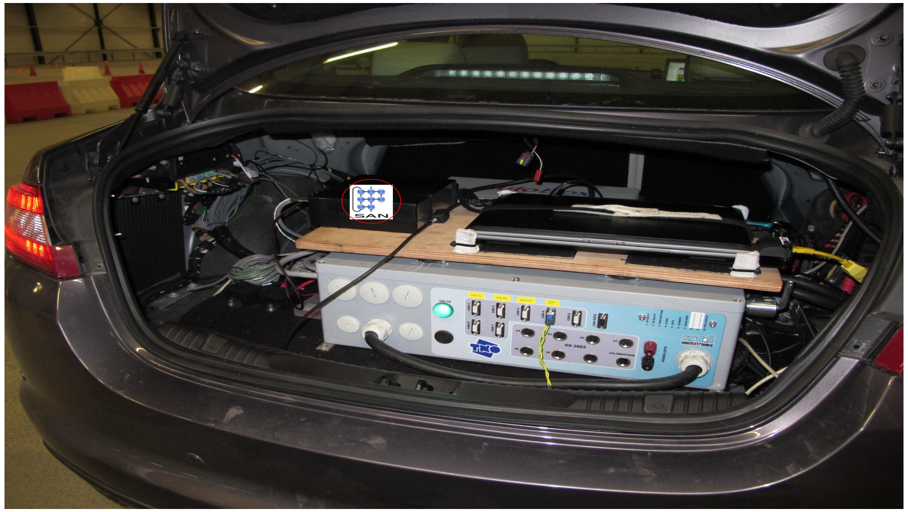
Demo: active suspension on



Demo: active suspension off



Demo: active suspension on

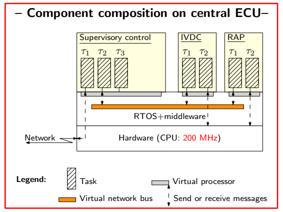


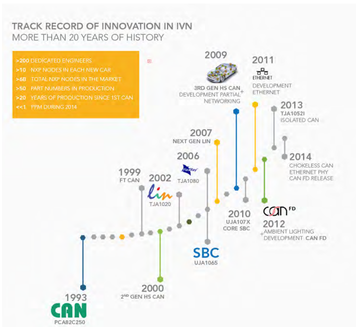
Resource sharing across components: a closer look

Tasks may request:

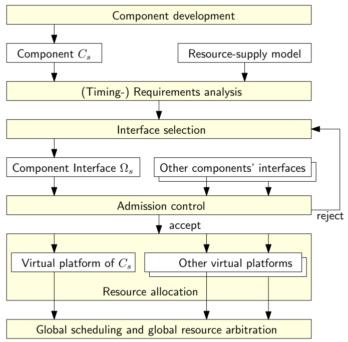
1. Access to shared memory:
   - shared buffers;
2. Operating-system services:
   - processor scheduling;
   - device drivers.
3. Network services:
   - send/receive messages;
   - abstraction of fieldbus technology:

   FlexRay vs. CAN



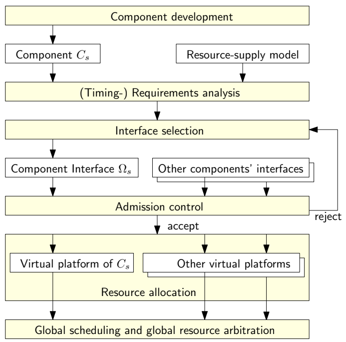


Challenges for resource sharing across components



1. Independent development of components:
   - RAP, Suspension and Supervisor;
2. Integration of components:
   - Communication abstraction;
   - Servers as a virtual processor;
3. Scheduling components and containment of temporal faults.

Challenges for resource sharing across components



1. **Independent development of components:**
   - RAP, Suspension and Supervisor;
2. Integration of components:
   - Communication abstraction;
   - Servers as a virtual processor;
3. Scheduling components and containment of temporal faults.

Independent development: Run-Away Process

Greedy for processor cycles:

On received-message "start"

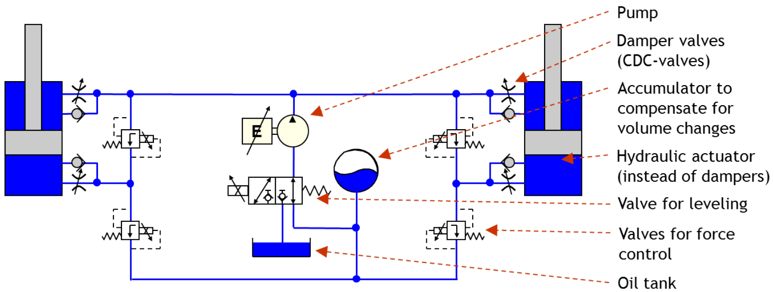
```
repeat
    ; {Do nothing}
until received-message "stop"
```

- Can be started/stopped via network messages;
- Highest priority application;

Purpose: Demonstrate temporal isolation

Independent development: Suspension control

Hardware for 1 axle:



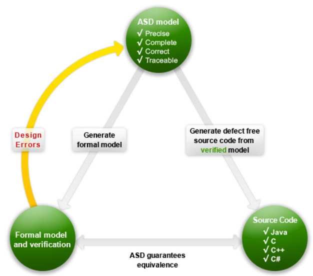
Software per wheel (2 tasks for force control):

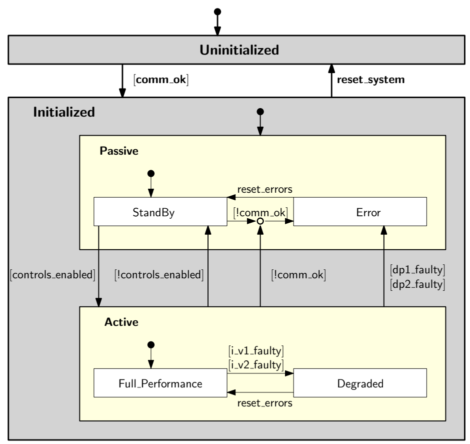
- Current control of valves (400 Hz);
- Pressure control of valves (100 Hz).

Independent development: Supervisory control

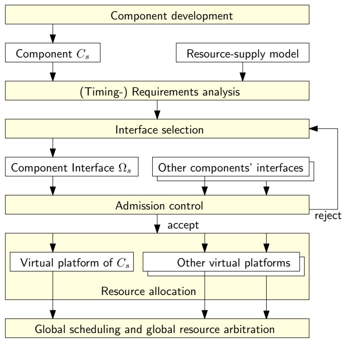
Handle state changes:

- Fault model:
  - message loss;
  - range check sensor values.
- Formal verification:
  - deadlock avoidance;
  - completeness of actions.
- Code generation:
  - MISRA C compliant;





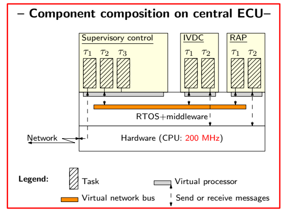
Challenges for resource sharing across components

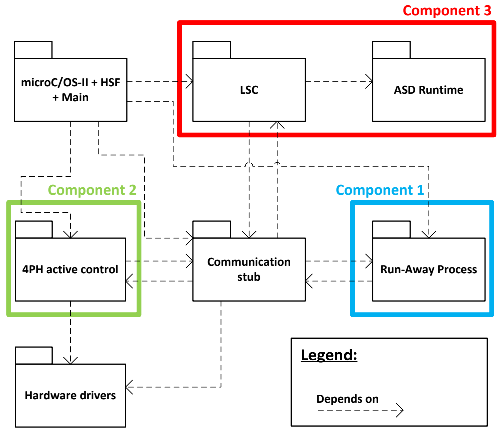


1. Independent development of components:
   - RAP, Suspension and Supervisor;
2. **Integration of components:**
   - Communication abstraction;
   - Servers as a virtual processor;
3. Scheduling components and containment of temporal faults.

Integration of components: overview

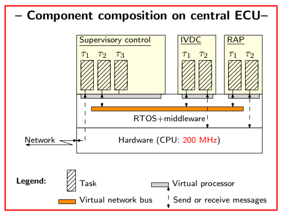
Independently developed components, their modules and interfaces:

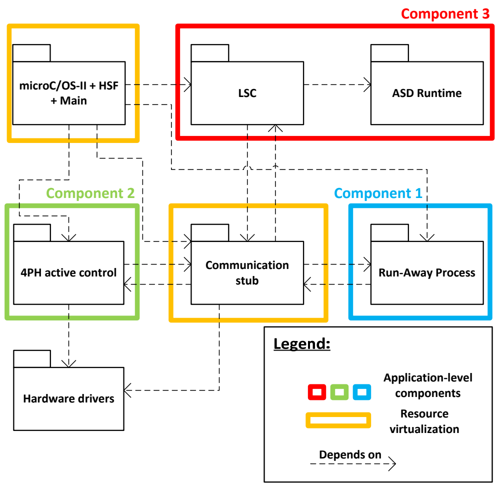




Integration of components: overview

Dependencies between software modules and components:



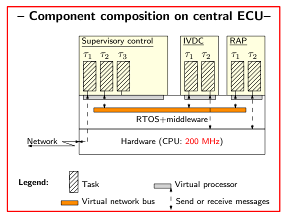


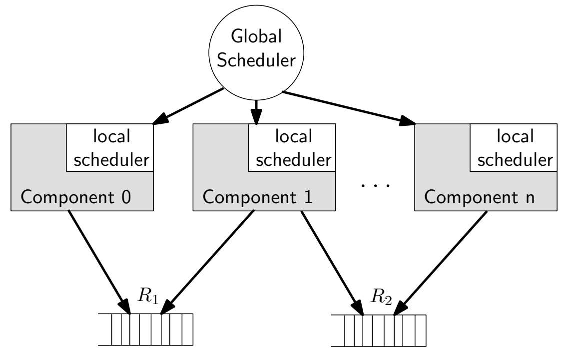
Virtualization of:

1. the processor (HSF);
2. the network (communication stub).

Integration of components on HSF-enabled kernel

Hierarchy of processor schedulers:

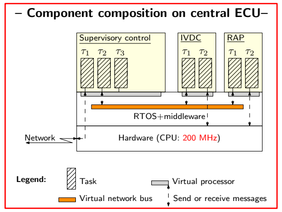




- component: *server*, *set of tasks* and *local (task) scheduler*
- server or virtual processor: a CPU budget allocated each *period*

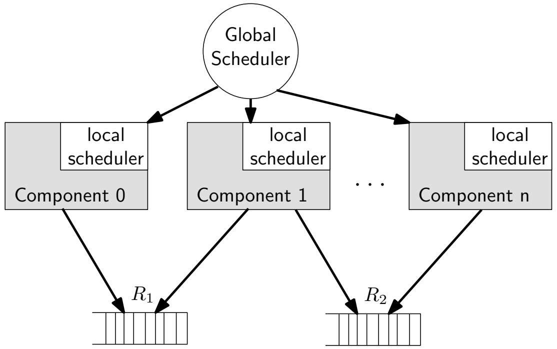
Tasks, located in arbitrary components, may communicate

Components, servers and tasks



4 servers:

- In descending priority:
  RAP, suspension, supervisor, idle



System tasks:

1. initialization task;
2. idle task.

Application tasks:

1. Run-Away Process;
2. 2x suspension control (400 Hz and 100 Hz);
3. Supervisor (100 Hz).

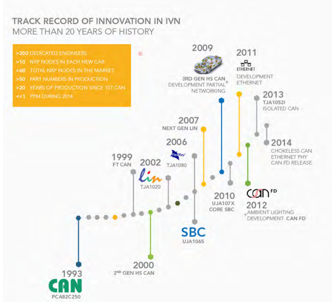
Integration of components: communication abstraction

Time- vs. event-triggered communication:

- Suspension control loops assume timed activation;
- Supervisory control assumes event triggers.

Network technology:

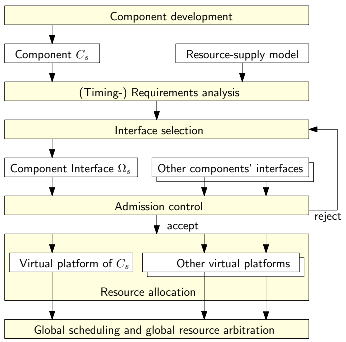
- CAN: event-triggered.
- FlexRay (static segment): time triggered.



Communication abstraction:

- Make all events timed.
- Assumption: application states depend on last event only.

Challenges for resource sharing across components



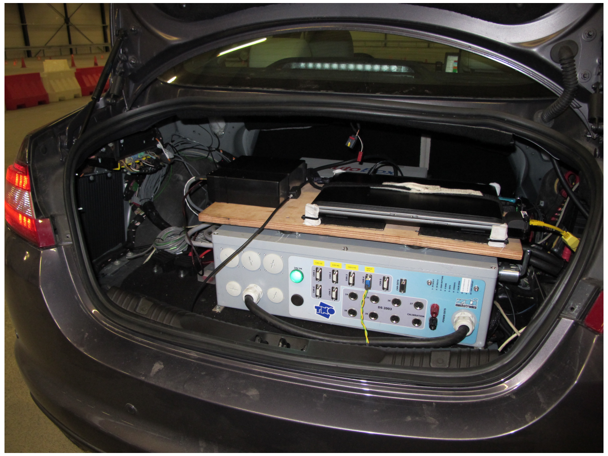
1. Independent development of components:
   - RAP, Suspension and Supervisor;
2. Integration of components:
   - Communication abstraction;
   - Servers as a virtual processor;
3. Scheduling components and containment of temporal faults.

Current and future challenges:

- **Mixed criticality:**
  1. deal with uncertain timing specs of tasks;
  2. graceful degrade functions by enabling/disabling optional ones.

- **Industrial standards:** timing augmented descriptions of components.

Questions?



Let's pass the remote . . .

HTAS AUTOMOTIVE INNOVATION