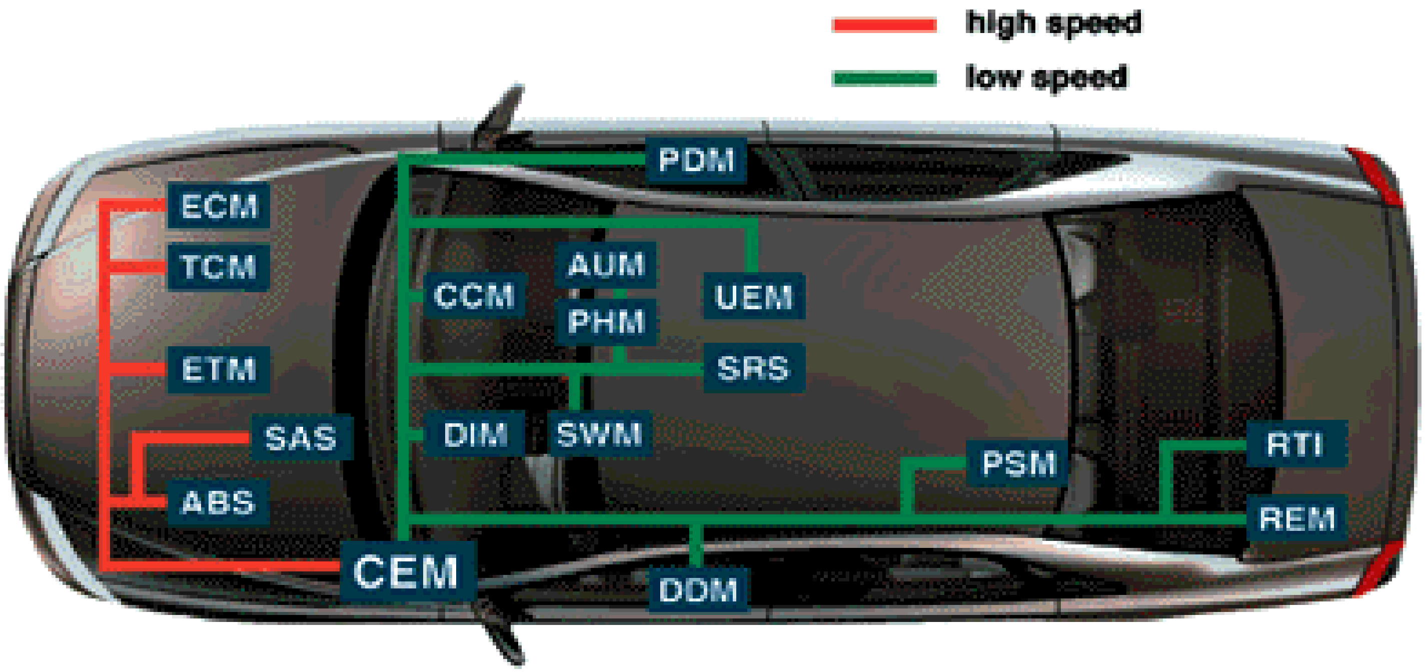
Dependable Automotive Electronics

Electronic systems: work areas

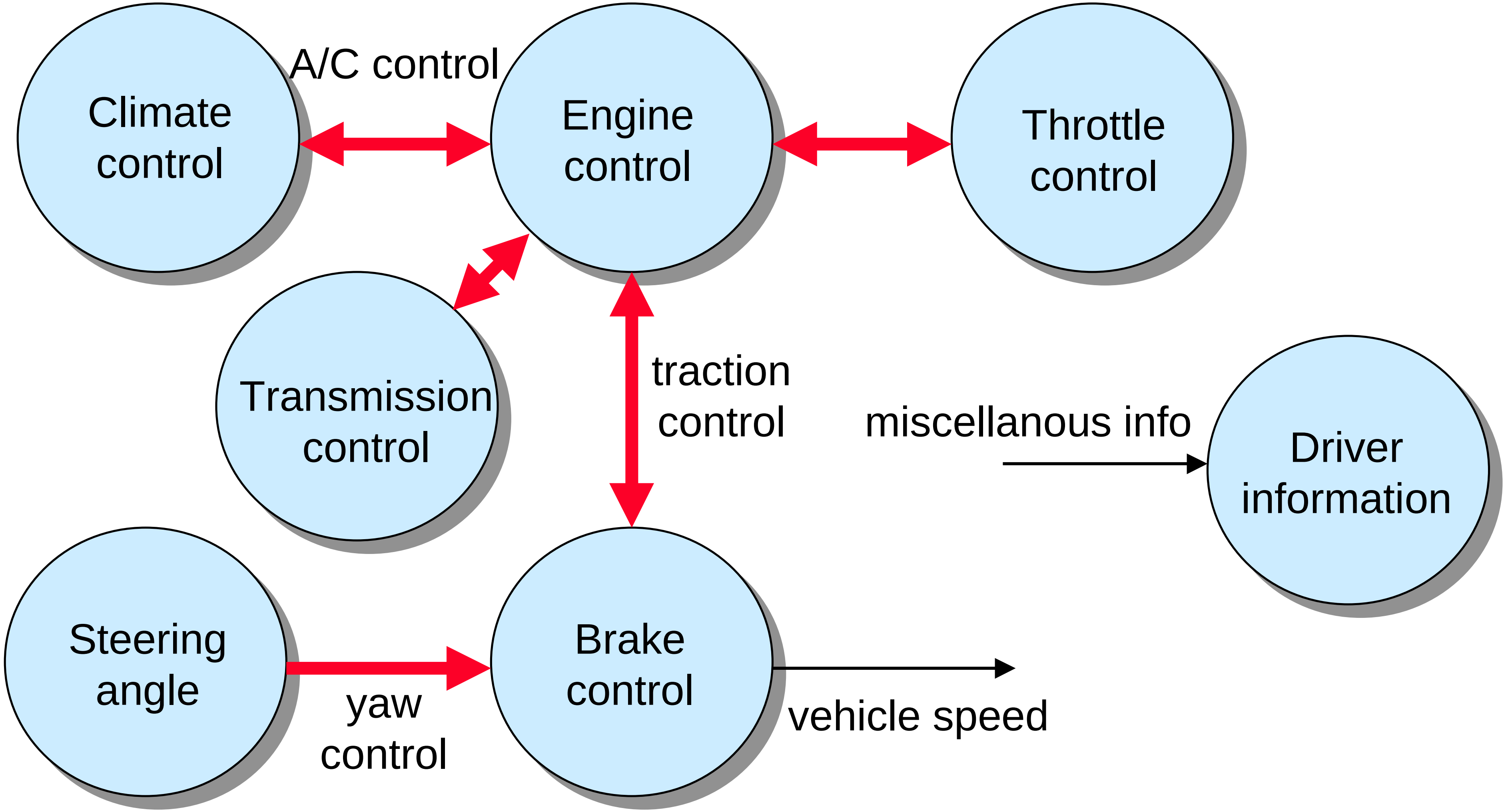
- **Systems engineering**
  e.g. requirements, specification, modelling, ...
- **Dependable computer systems**
  e.g. fault tolerance, fail-safe systems, ...
- **Data communication**
  e.g. fiber optics, bluetooth, ...
- **Distributed systems**
  e.g. timing, allocation of functionality, ...
- **Electronics**
  e.g. sensors/actuators, prototypes, ...

Current situation and trends in automotive electronics

S80 Electrical System



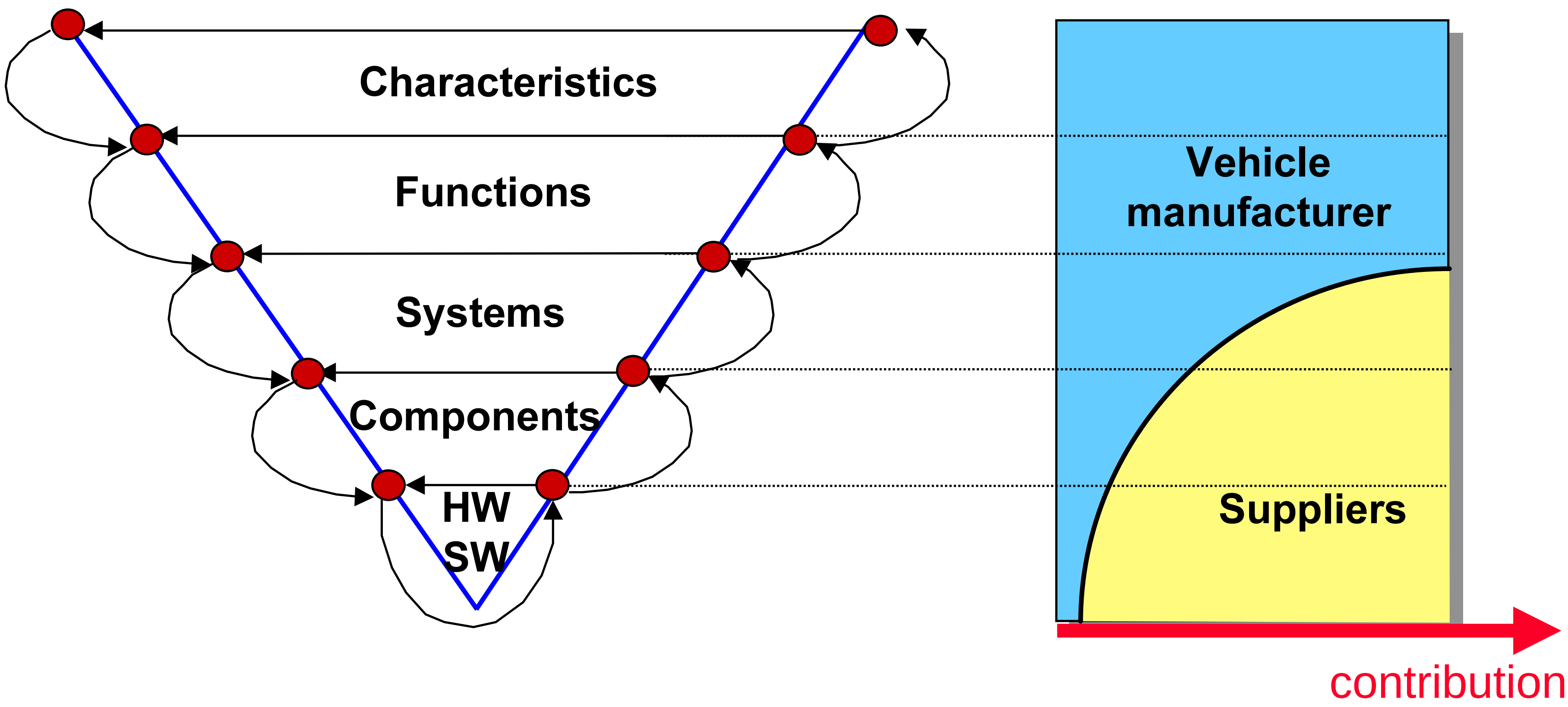
Some examples of today's distributed functions



Why by-wire systems?

- more advanced functionality (“no” physical limitations)
- facilitates distributed functions
- hydraulic systems are environmentally unfriendly
- cost, weight and space reduction
- MMI design can focus on safety and ergonomy

Development Process



Automotive dependability requirements

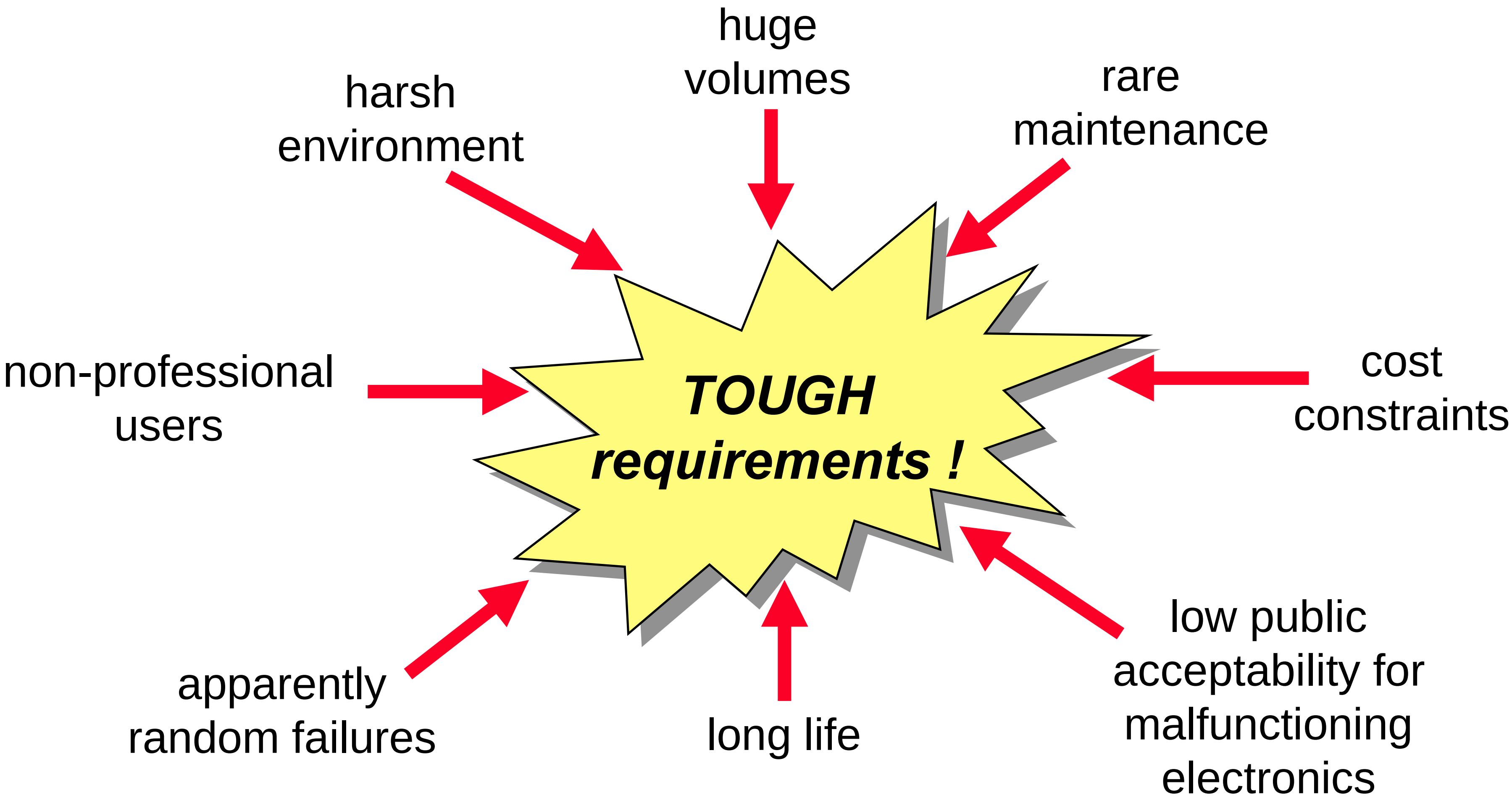
Background: Some statistics

• VCC annual production	~ 400 000 cars
• Life length per car	~ 15 years
• Driving time/car/year	~ 500 hours

→ In one year, Volvo cars accumulate around 3 000 000 000 driving hours

Conclusion: Safety-critical systems have to be extremely dependable

Dependability requirements considerations



Fault-tolerant and fail-safe systems

different applications

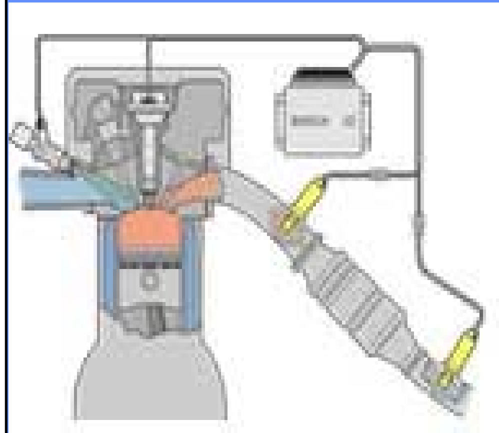
↓

different requirements

↓

different solutions

Engine control



selection of a degraded mode when an error is detected

Anti-lock braking

system switch-off when an error is detected

Safety-Related Requirements

Error detection requirement:

- The system shall be able to detect fault X

Error response requirement:

- If fault X is present, the system shall at least provide the functionality**...**
- If fault X is present, the system shall switch itself off ("fail-stop")
- No single fault shall cause safety-critical malfunctioning

Development process requirement:

- The software shall be developed according to**…**

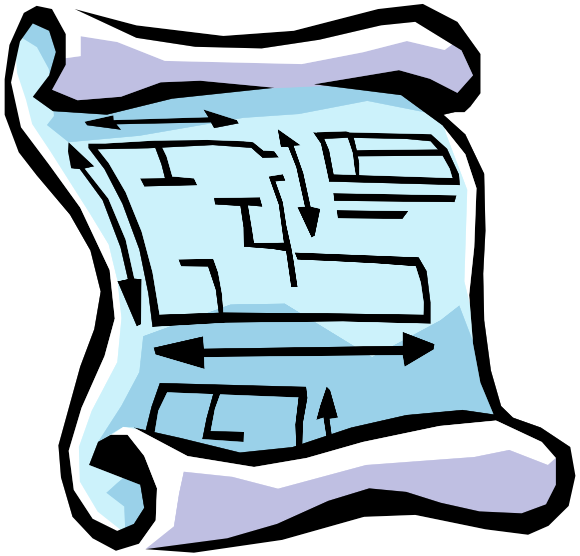
Design-specific requirements:

- The design shall be such that **…** (specific design details)

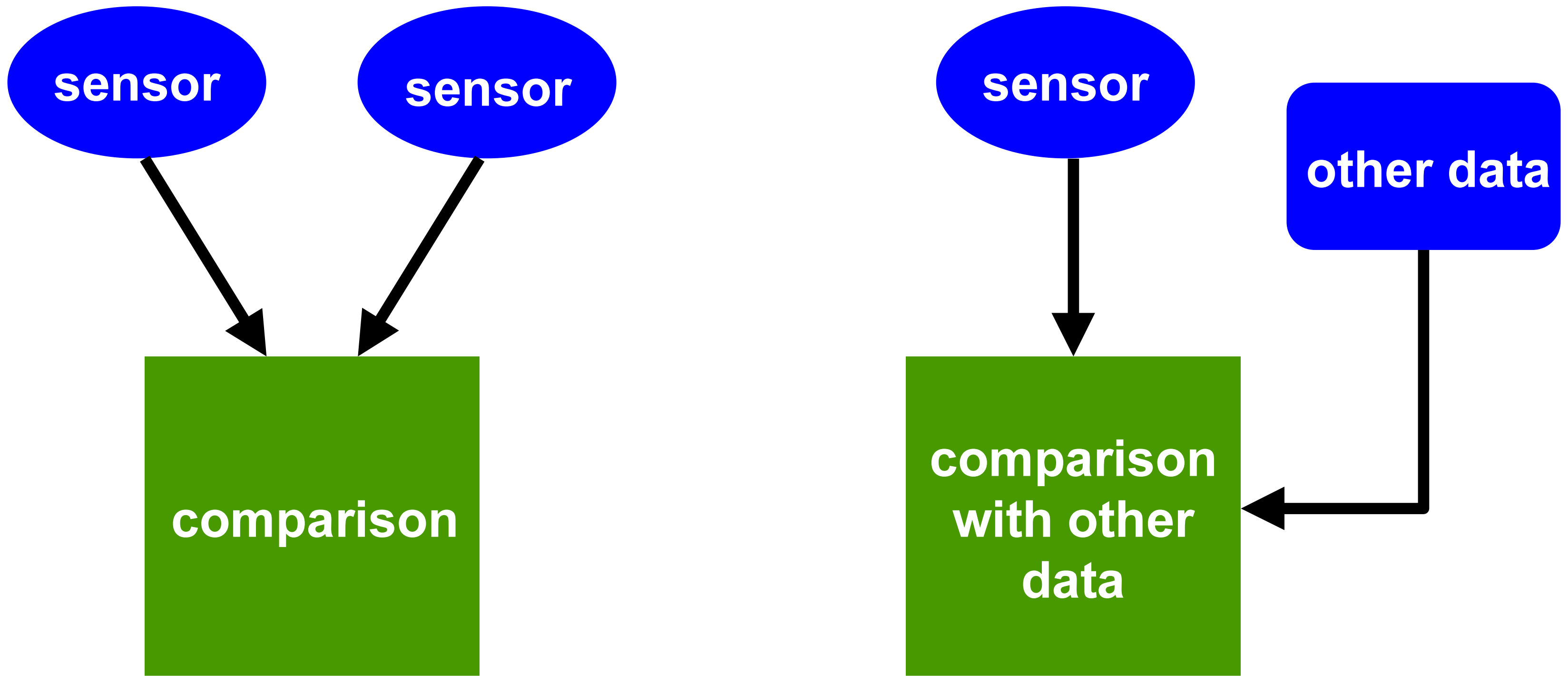
Quantitative requirements on safety and reliability:

- The probability of malfunction Y during the time interval (0, T) shall be less than**…**

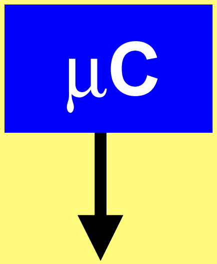
Some design issues



Alternative mechanisms for detection of sensor errors

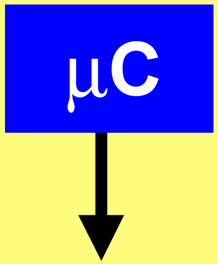


Detection of “computer errors”



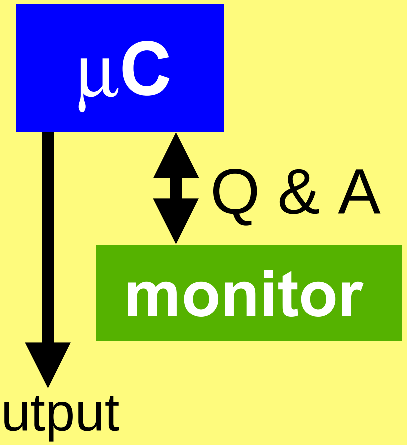
direct control of e.g. electrical motor

Requires that the output is monitored for correctness by redundant HW



encoded output (e.g. communication)

Sufficient to check that the output format is correct?



Sufficient that the monitor checks the answers?

Computer redundancy

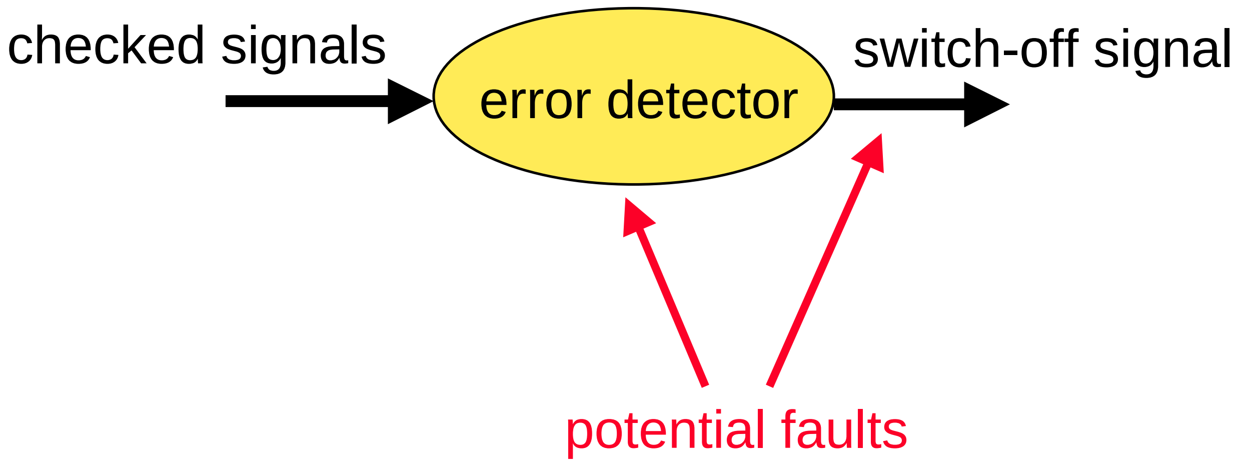
	≈100% redundancy	< 100% redundancy
single HW	double execution N-version programs	checks: - memory - program flow - exceptions application assertions
multiple HW	dual redundancy - SW replication - SW diversity TMR	monitoring: - end-to-end - Questions/Answers - watchdog timer

Detection of latent faults

A single fault shall never lead to safety-critical malfunctioning

This is a nice requirement if multiple faults are avoided

→ Detection and handling of latent faults is essential



Verification and Validation

Overview of V & V techniques

FMEA (Failure Mode and Effects Analysis)

- Qualitative checklist

FTA (Fault Tree Analysis)

- Focus on critical faults; Qualitative or quantitative analysis

Markov models (state-space models)

- (Usually) quantitative analysis

Software inspection (development process and code)

Formal verification of software

- Prove that design has certain properties

Test and simulation, with/without fault injection

Testing of an onbord system

Hardware-in-the-loop

Real system

Controllable, repeatable,
non-dangerous

Simulation

Complete vehicle

Rapid control prototyping

Real car

Possible before the
system exists

Summary

- Increased amount of distributed functions
- Increased amount of safety-critical electronic systems
- Automotive dependability requirements are extremely strong
- Strategy for interaction with suppliers is becoming increasingly important
- Key techniques:
  - Specification methods, e.g. UML, Statecharts, Matlab/Simulink
  - Formal methods
  - Fault-tolerant and fail-safe design methods
  - Dependability analysis (FMEA, Fault trees, etc)
  - Hardware-In-the-Loop Simulation
  - Fault injection