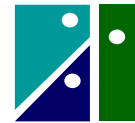


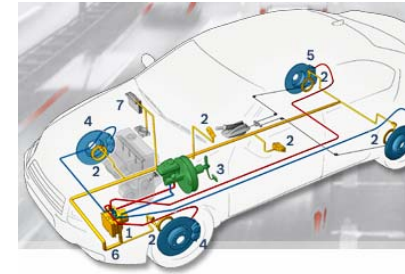
Workshop 2: Robustheit & Safety

Systemsoftwareentwicklung für zeit- und sicherheitskritische Anwendungen im Automobil

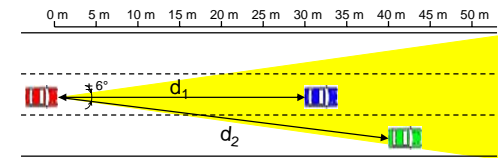
Wolfgang Albrecht
Fachhochschule Koblenz
FB Elektrotechnik und Informationstechnik



Aktive Sicherheit:
z.B. ABS, ESP



Fahrerassistenzsysteme:
z.B. Adaptive Cruise Control (ACC)

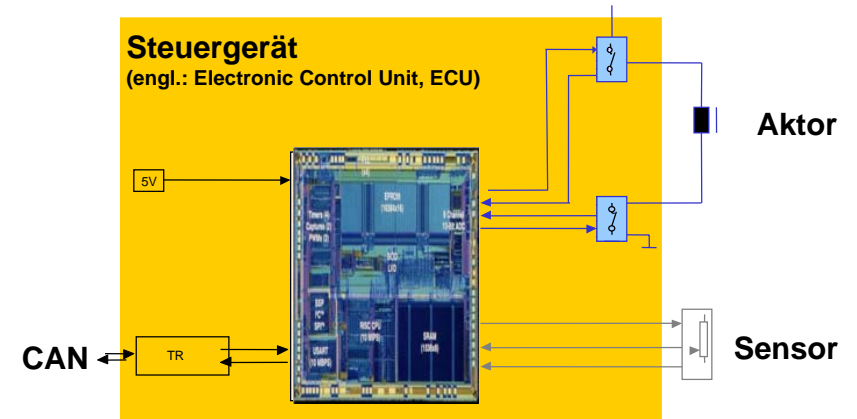


Systemsoftwareentwicklung für zeit- und sicherheitskritische Anwendungen im Automobil

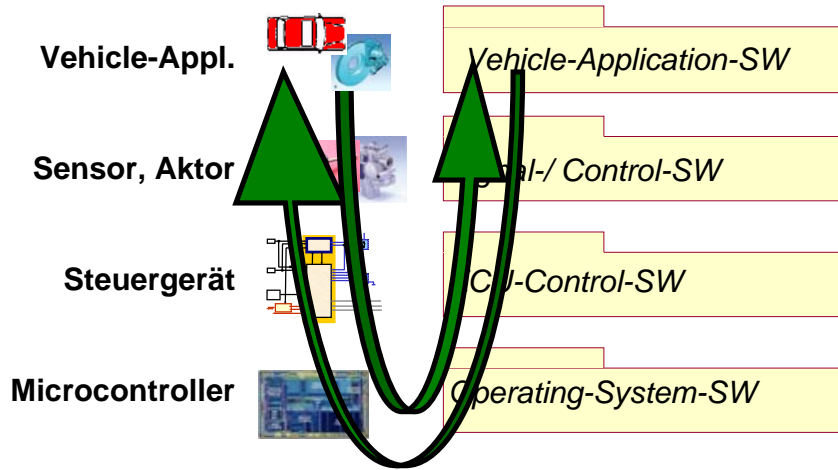
Inhalt:

- SW-Architektur im Anwendungsbereich Automobil
- Robustheit & Safety im Automobil:
Zentrale Begriffe / Klassifizierungen
- Failsafe-Techniken

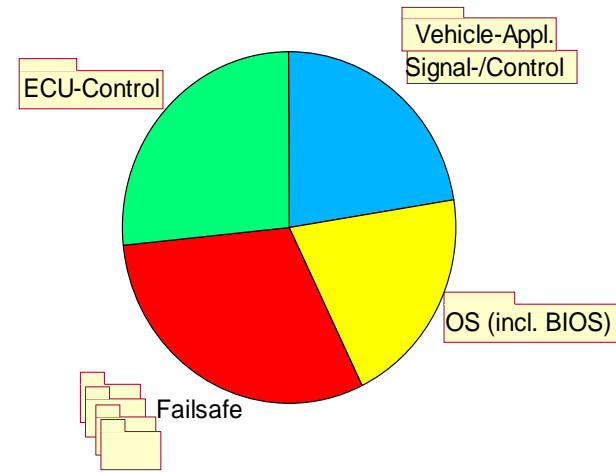
Elektronik: Sensor, Aktor, Steuergerät, uC



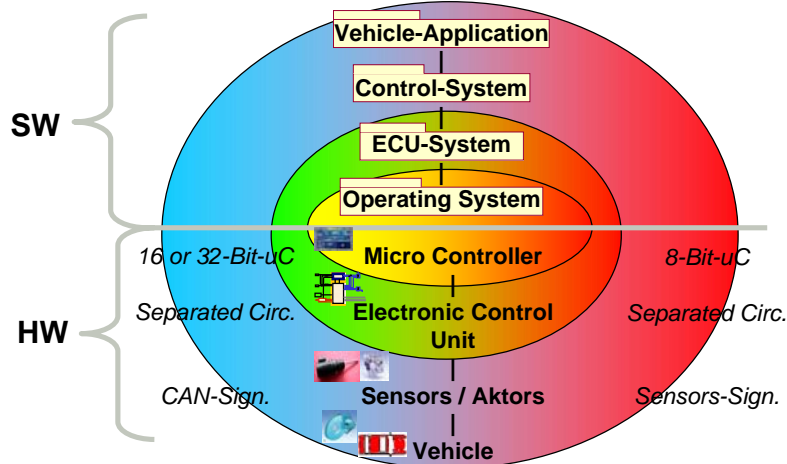
Software-Architektur



Aufteilung des Ressourcen-Verbrauchs (am Beispiel ROM-Bedarf)



Failsafe-Anteile Redundanz



Systemklassifikation (vgl. [Kop97]):

- **fail-safe:**
sicherer Zustand für deaktiviertes System existiert und ist leicht/schnell erreichbar...
- **fail-operational:**
...solch ein Zustand existiert nicht, d.h. Betrieb muss zumindest in Form eines Notlaufs erhalten bleiben

Focus bei **fail-safe** Systemen:

- Fehler muss „nur“ rechtzeitig erkannt werden und
- Melde- und Deaktivierungsmechanismus auslösen

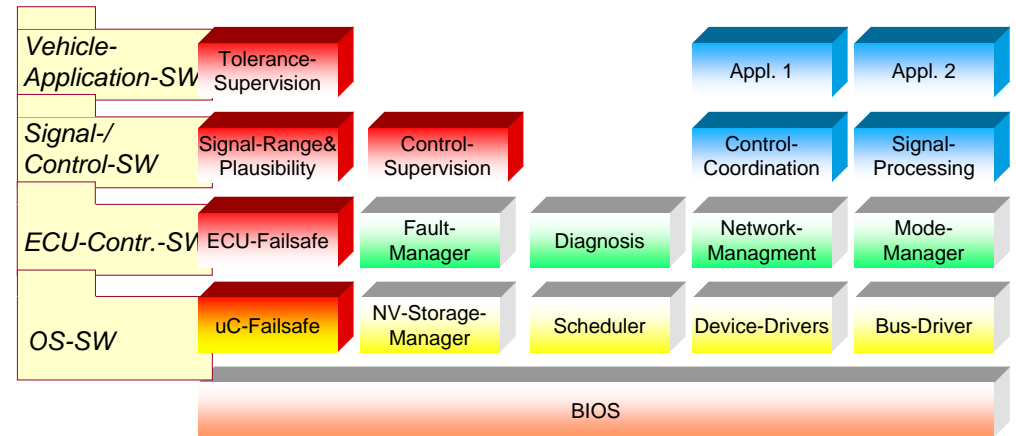
Klassifikation von Fehlfunktionen (vgl. [Mah00])

- 1.) Sicherheitsrelevanz
 - Kein **Einzelfehler** darf sicherheitskritische Systemreaktion auslösen, bzw. zu nicht beherrschbarem Fahrzeugverhalten führen
 - „dual Input“: entweder zweiter, redundanter Sensor, oder über andere Signale plausibilisieren
- 2.) Verfügbarkeitsverlust:
 - Übergang in Notlauf / Rückfallebene
- 3.) Schlafende Fehler:
 - weiterer Fehler kann zu sicherheitskritischen Systemzustand führen
 - Bsp.: Fehler in „Notaus“-Funktion
- 4.) Degradation der Funktionalität ... 5.) Komfort-Beanstandung

Ziel 1: Sicherheit \leftrightarrow Ziel 2: Verfügbarkeit

Software-Architecture

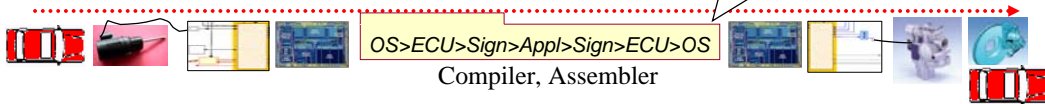
(Example for Electronic Controlled Actuation)



0.) „Strecke“ und Fehlerquellen

Verhinderung von SW-Fehler
→ Focus auf Entwicklungs-Prozesse, z.B. SPICE
Software Process Improvement and Capability dEtermination
(ISO/IEC TR 15504:1998)

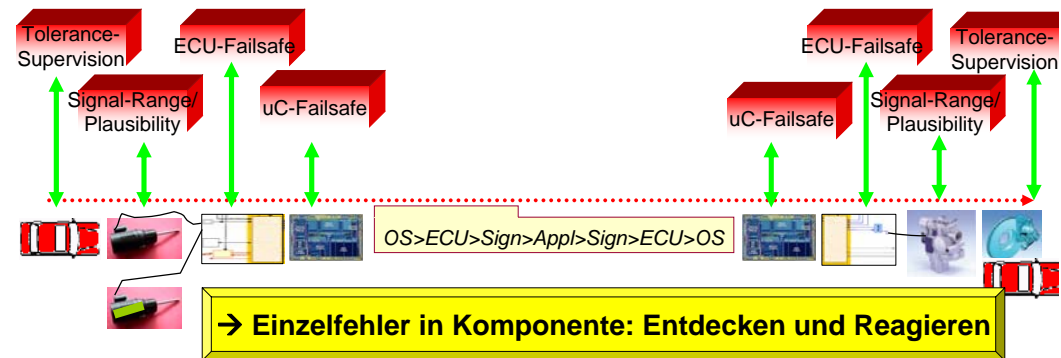
„Strecke“ = Quelle möglicher Fehler



Systematische Fehler: bereits mit Beginn der Lebensdauer einer Komponente, wie beispielsweise einem Rechner bzw. eines Softwareproduktes, in diesem enthalten
→ bisher keine nachweislich gültige Verteilungsfunktion bzgl. Ausfallverhaltens von Softwarefehlern

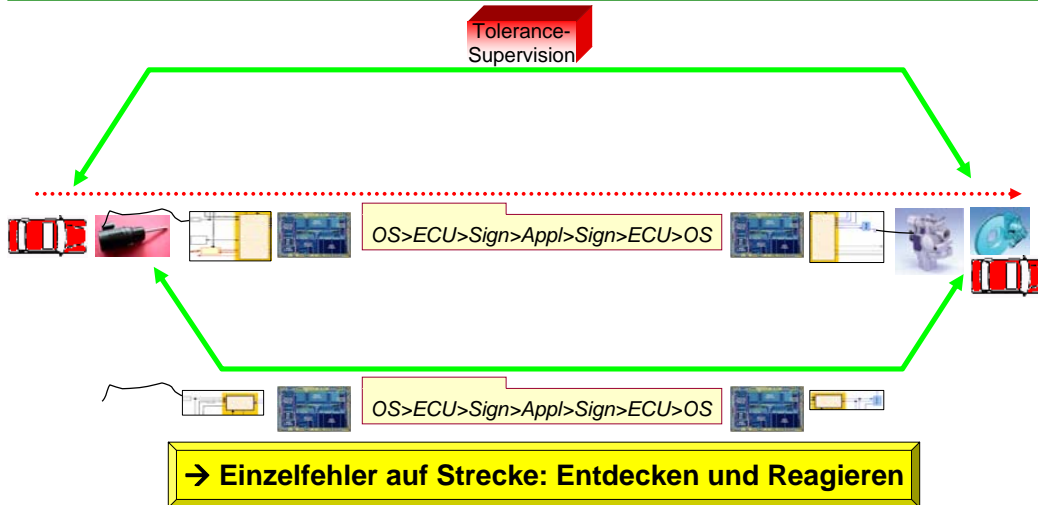
Zufälliger (stochastischer) Fehler: Auftretiszeitpunkte nicht vorhersagbar, jedoch über der Betriebszeit durch Verteilungsfunktion beschreibbar (Bsp.: elektr. Bauteile oder „Bit-kippen“)
→ statistisch unabhängige **Einzelfehler**

1.) Überwachung der Komponenten



Fehler teils nur mittels redundanter Eingabe detektierbar
Dual-input-Prinzip -- am besten **diversitär**

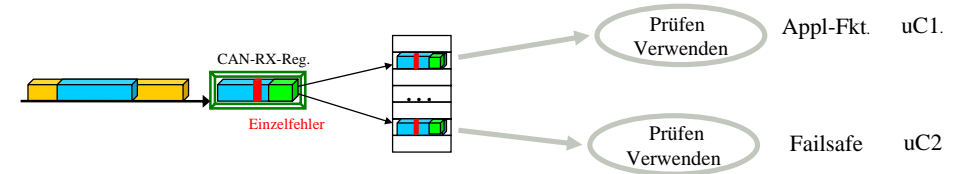
2.) Überwachung der „Strecke“ für Normalfunktion



- 13 -

Beispiel: Absicherung von CAN-Botschaften

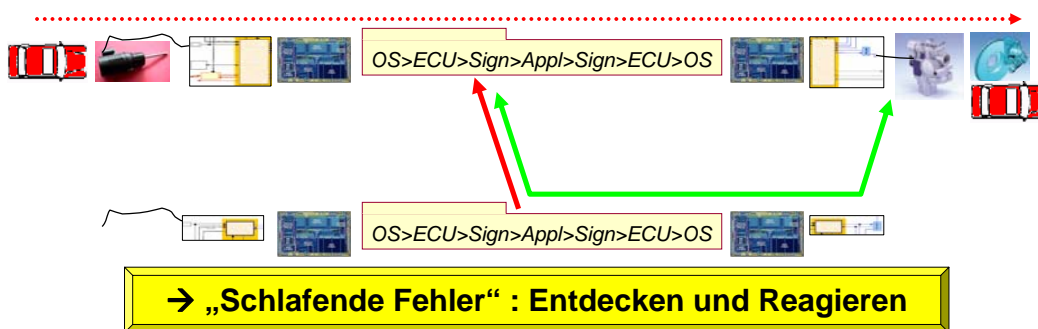
- Voraussetzung: Botschaft kommt von „eigensicherem“ Steuergerät
- Übertragung gilt als sicher (CRC-Checksumme), aber
- entgegen nehmen und weiter reichen!?
 - ggf. immer die selben (alten) Daten weitergereicht
 - Fehler in betr. RAM-Zellen oder fehlerhaftes Kopieren



- Lösung (1Byte der Nutzdaten für)
 - Live-Counter
 - Checksumme

- 14 -

3.) Überwachung der Überwachung für Failsafe-Funktion



- 15 -

Literatur:

- [Alb01] W. Albrecht, V. Braschel, S. Borsch: *TRW-Braking-Systems: Sensors, Software, Systems*, Handout zu Vorträgen an FH-Saarbrücken, 2001
- [Kop97] H. Kopetz: *Design Principles for Distributed Embedded Applications*, Serie Real-Time Systems, Kluwer Academic Pub., 1997
- [Mah00] R. Mahmoud: *Sicherheits- und Verfügbarkeitsanalyse komplexer Kfz-Systeme*, Dissertation Fachbereich Elektrotechnik und Informatik, Universität-Gesamthochschule Siegen, 2000

- 17 -