Funktionskapselung in Steuergeräten

„Mobilität und Echtzeit“ – Boppard am Rhein, 07.12.2007

Denis Eberhard, I/EF-6

Inhalt

- Ausgangssituation und Motivation
- Begriff "Kapselung"
- Ursachen von Schutzverletzungen
- (Speicher-) Schutzmechanismen
- Zusammenfassung und Ausblick

Denis Eberhard, I/EF-6

Ausgangssituation und Motivation

Wandel der System-Architekturen

Denis Eberhard, I/EF-6

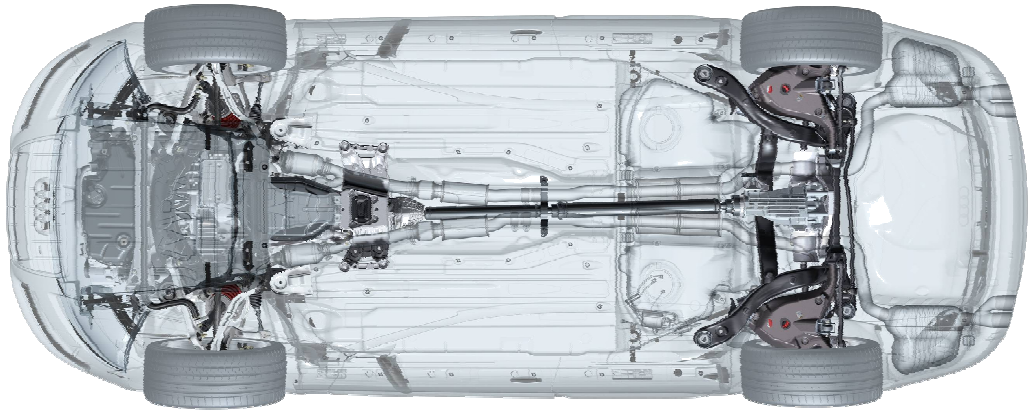
Motivation

- Reduzierung der Steuergeräteanzahl: Einsparung von Gewicht und Platz sowie Verringerung der Fertigungskomplexität
- Erhöhung der Wiederverwendbarkeit und Modularität der Systeme
- Verbesserung der Stabilität durch Einsatz betriebsbewährter Module/Teile

➔ Integration mehrerer Software Module in ein Steuergerät

<u>Kernaspekte:</u>

- Systemsicherheit
- Systemstabilität
- Zuordnung von Haftung



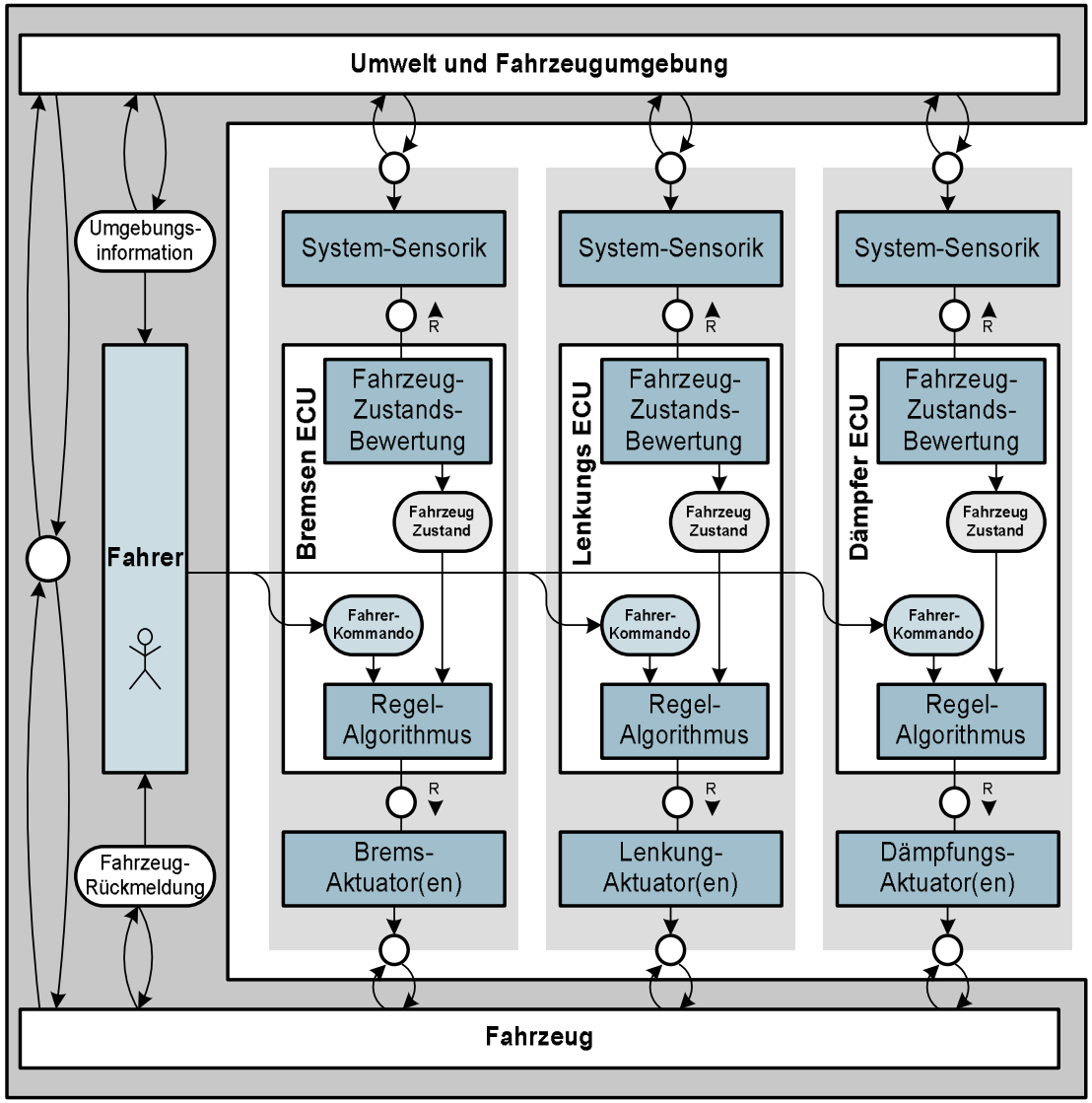
➔ In sicherheitsrelevanten Systemen wie z.B. Fahrwerksystemen ist eine Kapselung von Funktions-Software Modulen untereinander erforderlich.

Systemlandschaft bisher

Fahrzeug-Ebene

Implementierung jeder Funktions-Software auf einem eigenen Trägersystem (Steuergerät)

- ➔ sehr hohe Anzahl an Einzelsystemen
- ➔ Mehrfachkosten in der Implementierung
- ➔ aufwendige Logistik, hohe Fertigungskomplexität



Denis Eberhard, I/EF-6

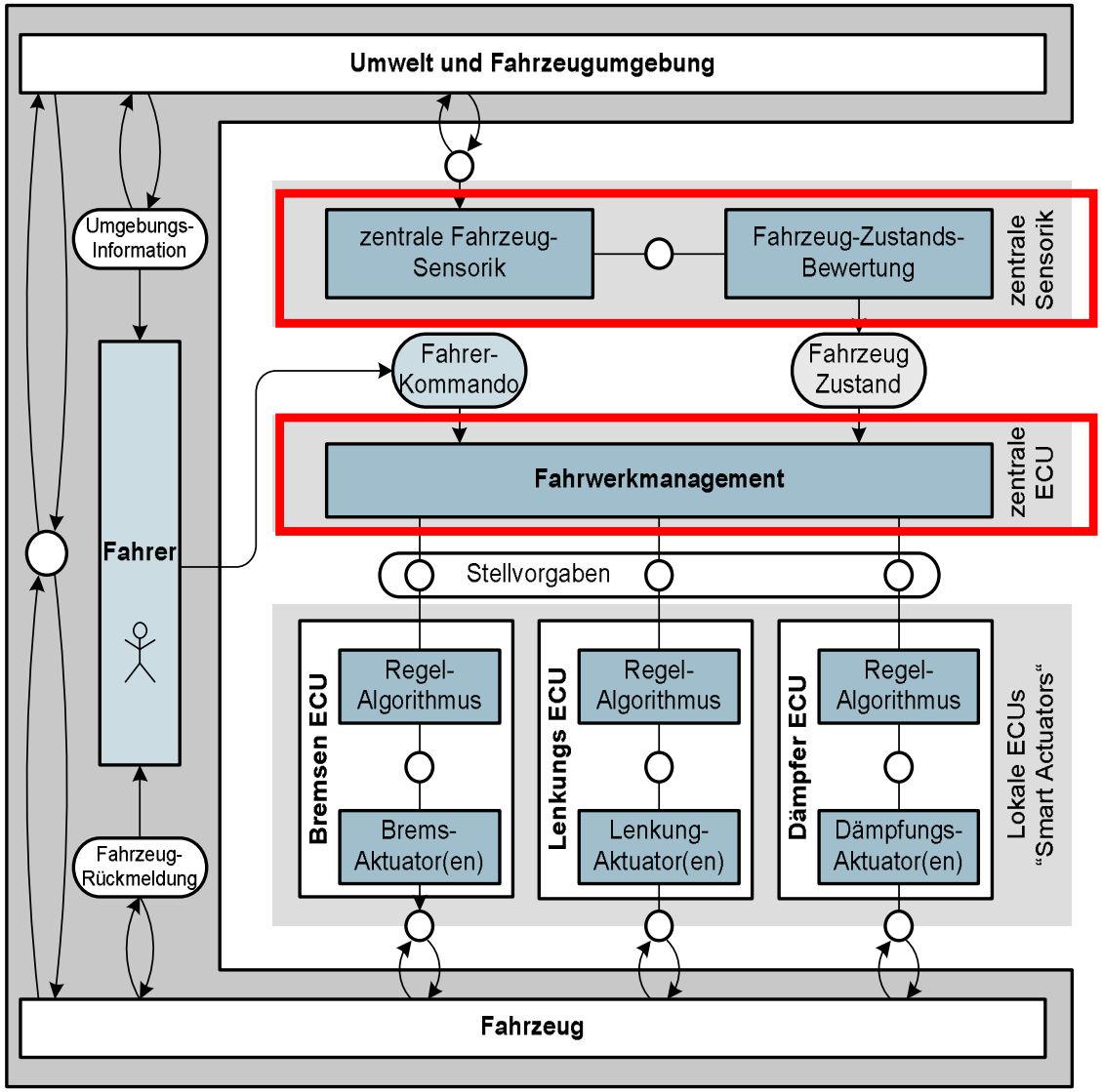
Systemlandschaft zukünftig

Fahrzeug-Ebene

- Modularisierung der Software
- Zentralisierung und Integration von Funktionen

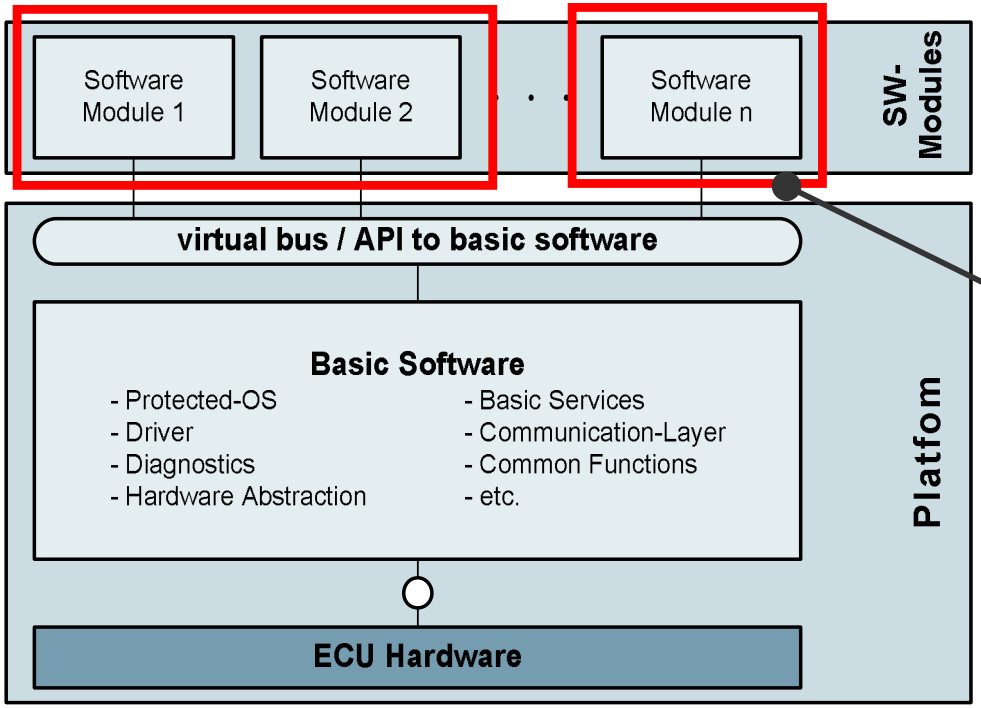
➔ Erschließung neuer Potentiale durch Zentralisierung

➔ Reduzierung der Steuergeräte-Anzahl (Gewicht, Platz, Komplexität)



Dekomposition des Systems

Steuergeräte-Ebene



- **Basis Software**
  Alle allgemeinen und Applikations-abhängigen Teile der Software: OS, Driver Framework, Diagnose Dienste, Kommunikations-Protokolle, Hardware-spezifische Software, etc.: alles außer dem eigentlichen Funktions-Algorithmus
- **Plattform**
  Kombination aus Hardware und der darauf laufenden Basis Software, die alle Funktionen der Hardware unterstützt.
- **Software-Modul**
  Granularer Teil von Software, der eine bestimmte Funktionalität einer Applikation abdeckt.
- **Applikation**
  Algorithmus des Fahrzeug-Systems mit den spezifischen Diagnose- und Fehlerbehandlungs-Routinen, besteht aus einer oder mehreren Software-Modulen

➔ Bei der Integration mehrerer Applikationen auf einer ECU könnten sich diese gegenseitig beeinflussen!

Denis Eberhard, I/EF-6

Begriff "Kapselung"

Bedeutung und Abgrenzung

Denis Eberhard, I/EF-6

Kapselung in der Informatik

Kapselung ist

"[…] eine Technik zur Strukturierung des Zustandsraumes ablaufender Programme. Ziel ist es, durch Bildung von Kapseln mit klar definierten Zugriffsschnittstellen die Daten- und Strukturkonsistenz zu gewährleisten. […]

Kapselung setzt eine Festlegung der Kapselgrenzen und der Schnittstellen an den Kapselgrenzen voraus." *)

*) Poetzsch-Heffter: Vorlesungsunterlagen Fortgeschrittene Aspekte objektorientierter Programmierung. Wintersemester 2002/03

Kapselung in der Automobil-Software

Kapselung bedeutet:

- Festlegung der Kapselgrenzen
- Zuteilung und Abgrenzung aller Ressourcen
- Definition der Zugriffsschnittstellen an den Grenzen
- Wahrung der Daten- und Strukturkonsistenz
- Überwachung und Aufrechterhaltung der Kapselgrenzen zur Laufzeit
- Konsequenzen aus Schutzverletzungen zur Laufzeit

Denis Eberhard, I/EF-6

Ursachen von Schutzverletzungen

Bauzeit und Laufzeit

Denis Eberhard, I/EF-6

Warum ist Kapselung nötig?

Zu verhindern: gegenseitige Korruption und Beeinflussung von verschiedenen Applikationen (Kapseln)

Beeinflussung:
- verursacht durch Implementierungsfehler oder durch äußere Einflüsse
- Ressourcenkonflikte (Zugriff auf fremde Ressourcen, Blockieren von Ressourcen,...)

Sicherheits-Standards:

- IEC 61508
- zukünftige ISO/WD 26262

- Design der Systeme orientiert sich an diesen Standards
- Integration von verschiedenen Applikationen mit unterschiedlicher Sicherheitsrelevanz können nur sinnvoll durch den Einsatz von Kapselung auf einem gemeinsamen Trägersystem integriert werden.

Denis Eberhard, I/EF-6

Kapselung und Sicherheit

Integration mehrerer Software-Module auf einem Trägersystem

Software-Module mit unterschiedlicher Sicherheitsrelevanz

Sicherheitsrelevanz?

Entwicklung nach jeweiligen Sicherheitsanforderungen

Geringere Sicherheitsanforderungen bedeuten erhöhte Restfehler-wahrscheinlichkeit!

Gegenseitige Beeinflussung bei Integration muss ausgeschlossen werden.

Kapselung erforderlich ODER

Entwicklung der gesamten Software-Module nach den höchsten Sicherheitsanforderungen (= teurer)

Denis Eberhard, I/EF-6

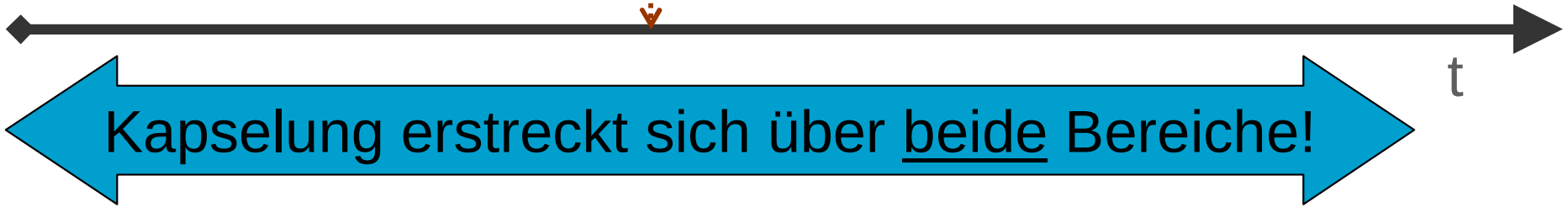
Wann wird "gekapselt" ?

Um Kapselung zu erreichen, müssen einige Schritte zur Bauzeit des Systems erfolgen und andere Mechanismen zur Laufzeit des Systems eingesetzt werden.

Bauzeit:
Konfiguration,
Resourcen-Planung,
Partitionierung,
…

Inbetrieb-nahme

Laufzeit:
Überwachung der Ressourcen und von Zugriffsrechten,
Aktivierung des Speicherschutzes, …



Denis Eberhard, I/EF-6

Verletzung der Kapselgrenzen (Beispiele)

Bauzeit:

- fehlerhaft implementierte Schnittstelle
- Programmierfehler (Pointer, Stack-/Variablenüberlauf, etc)
- Konfigurationsfehler / Ressourcenplanungs-Fehler

Laufzeit:

- Schutzverletzungen durch Software-Module mit niedriger Sicherheitsrelevanz (dadurch weniger Testtiefe bei Entwicklung)
- Nutzung unerlaubter Systemdienste
- Hardware-Fehler (z.B. stuck-at Fehler, EMV Einflüsse,…)

Denis Eberhard, I/EF-6

(Speicher-) Schutzmechanismen

Einsatz in automotive Steuergeräten

Denis Eberhard, I/EF-6

Grundlage von Kapselung

Resourcen:

- Speicher
- CPU Zeit
- Dienste
- Kommunikation
- Interrupts (Anzahl/Zeit)

- zur **Bauzeit** vom System-Integrator zu **konfigurieren**
- zur **Laufzeit** von der Basis Software zu **überwachen** (hauptsächlich vom OS)

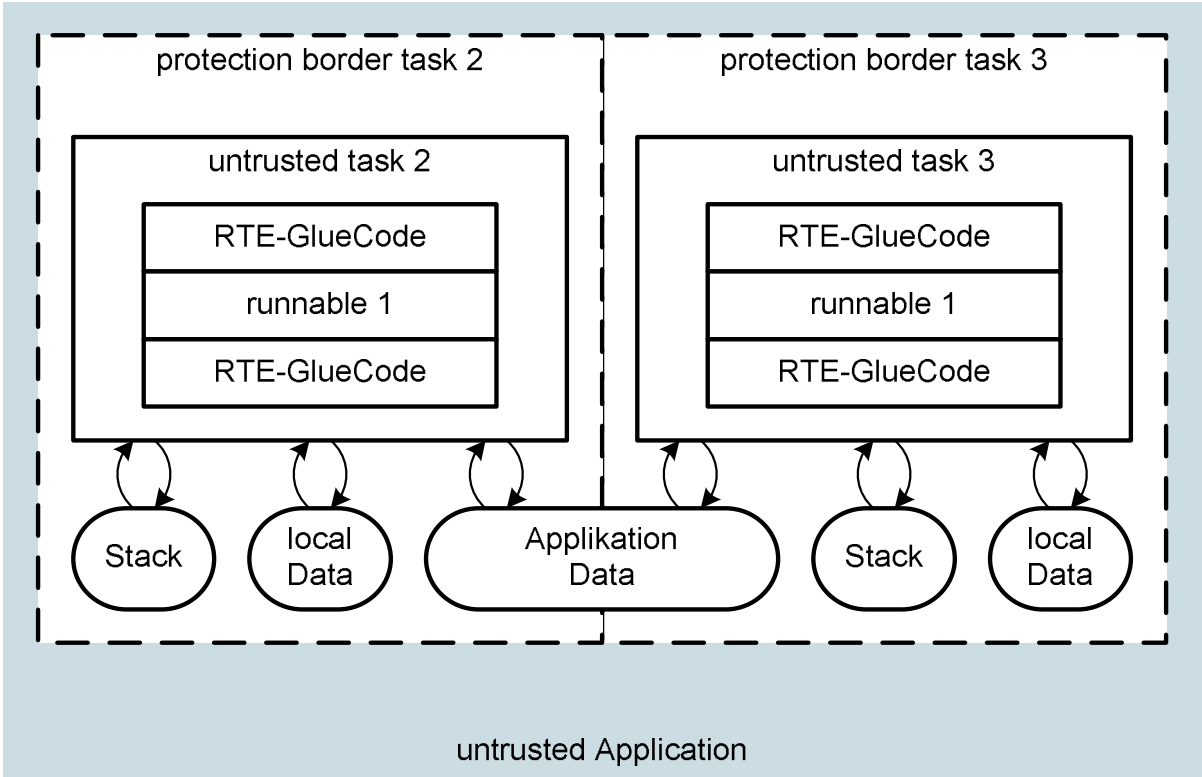
➔ Für jede Ressource sind andere Schutzmechanismen zur Überwachung nötig.

Denis Eberhard, I/EF-6

Speicherschutzbereiche einer Task

Speicherbereiche:

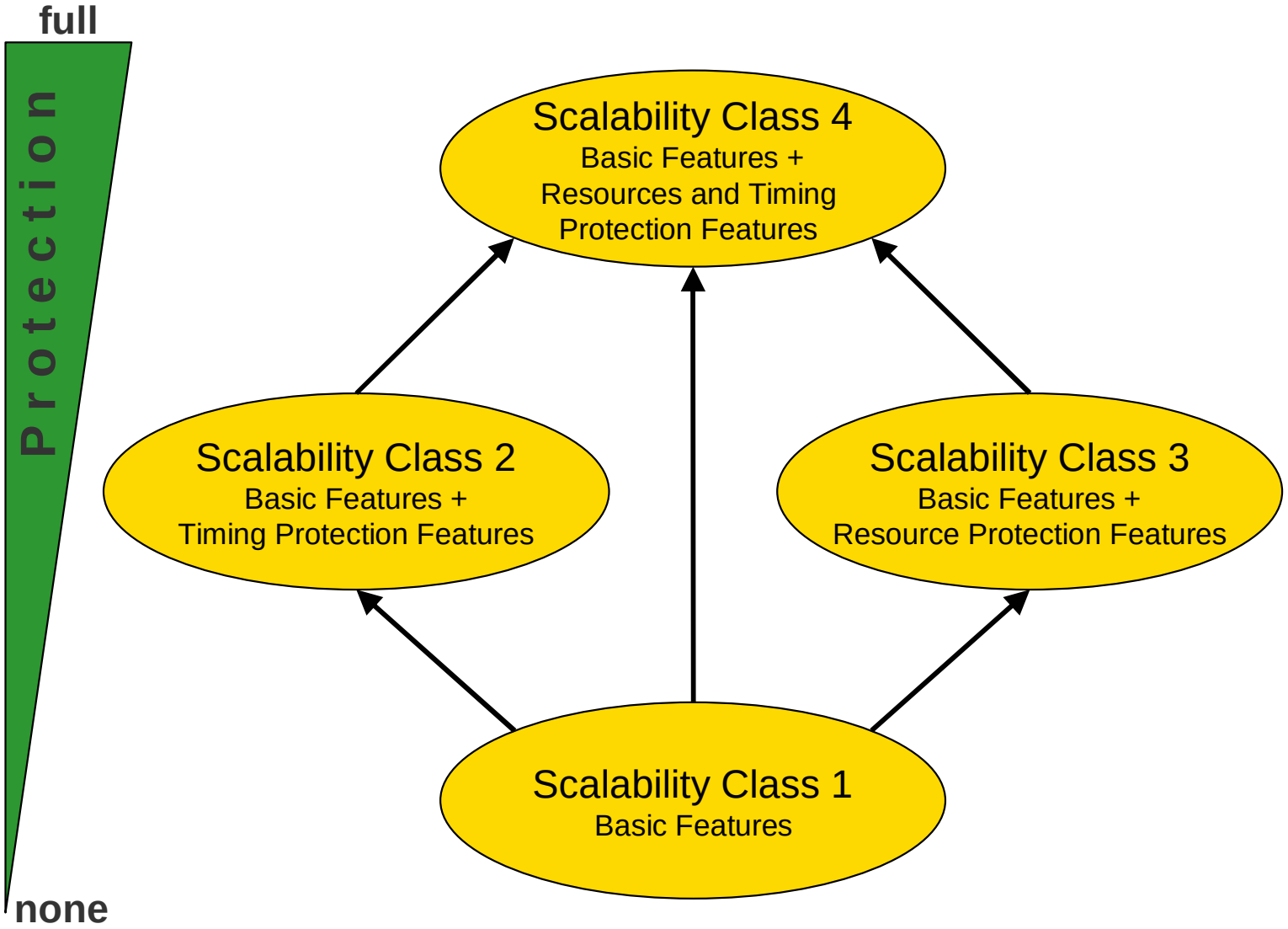
- Lokale Daten
- Applikations-Daten
- Stack
- Restlicher Speicher



➔ Mikrokontroller mit MPU nötig

Verfügbarkeit an Automotive-Mikrokontrollern mit MPU und genügend Speicherschutz-Registersätzen

AUTOSAR OS Scalability Classes



Resource Protection Features:
- Protection Hook
- Memory Protection
- OS-Applications
- Service Protection
- Call Trusted Function

Timing Protection Features:
- Protection Hook
- Timing Protection
- Global Time/ Synchronization Support

Basic Features:
- OSEK OS
- Counter Interface
- Schedule Tables
- Stack Monitoring

Denis Eberhard, I/EF-6

Zusammenfassung und Ausblick

Denis Eberhard, I/EF-6

Zusammenfassung

- Kapselung sicherheitsrelevanter Systeme im Fahrzeug möglich
- verbesserte Systemsicherheit und -stabilität
- reduzierte/begrenzte Anzahl an Steuergeräten
- größere Komplexität bei den Software-Releases
- Reifegrad und Umfang der Kapselungsmechanismen in AUTOSAR muss noch weiter verbessert werden

<u>Schwerpunkte in der Praxis verlagern sich</u>

- funktionsorientierter
- Software-Kapselung und –Schnittstellen werden wichtiger
- mehr Unabhängigkeit von der Hardware
- mehr Konfiguration, weniger manuelle Codierung

Denis Eberhard, I/EF-6

Ausblick

- Kapselung der Module der Basis Software
  ➔ komplett gekapselte Systeme

- Vollständige Unterstützung von Kapselung und Sicherheitsmechanismen in Standard Software Produkten und Architekturen wie z.B. AUTOSAR
  ➔ Je mehr standardisierte Software-Module, desto höher der Reifegrad und desto besser die Wiederverwendbarkeit

- Virtuelle Adressierung mittels MMU
  ➔ erhöhte Modularität und weitere Verbesserung der Speicher-Kapselung

Denis Eberhard, I/EF-6

Vielen Dank!

Denis Eberhard, I/EF-6