TECHNISCHE UNIVERSITÄT DRESDEN

Vorlesung
Automotive Software Engineering
Teil 7 Normen und Standards
AUTOSAR 1 Grundlagen

Sommersemester 2015

Prof. Dr. rer. nat. Bernhard Hohlfeld
Bernhard.Hohlfeld@mailbox.tu-dresden.de
Technische Universität Dresden, Fakultät Informatik
Honorarprofessur Automotive Software Engineering

TECHNISCHE UNIVERSITÄT DRESDEN

Inhaltsverzeichnis

Teil I Grundlagen

2 Softwarearchitektur in der Fahrzeugentwicklung

Allgemeine Informationen
Nicht AUTOSAR-spezifisch

2.1 Softwarearchitektur oder Systemarchitektur?

- AUTOSAR „Automotive Open System Architecture“
  - Verspricht eine „offene Systemarchitektur“
  - Beschreibung einer Systemarchitektur nach Automotive SPICE (www.automotivespice.org)
    - Festlegung der beteiligten Elemente aus Hardware und Software.
  - Automotive Produktentwicklung
    - Systemarchitektur beschreibt das ganzheitliche Zusammenspiel
      - Mechanik
      - Hydraulik
      - Pneumatik
      - Elektrik
      - Elektronik
        - Hardware
        - Software
      - Umwelt und Benutzer
        - Sensoren
        - Aktuatoren
        - Sollwertgeber
- AUTOSAR
  - Realisierung von Funktionen auf Steuergeräten durch Software
  - Architekturvorgaben für Realisierung durch Software.

2.1 Softwarearchitektur oder Systemarchitektur?

- Automotivebereich
  - Vielfältige Hardwarestandards
  - Beispiel: Bussysteme
    - Ein Bus als Kommunikationsmedium ist ohne die Daten sinnlos, die über ihn transportiert werden sollen.
    - Diese Daten werden von Software erzeugt und verarbeitet
    - Die oberen Schichten der Bussysteme werden mit Software realisiert.
    - Nächste Folie: Überblick über relevante Bussysteme im KFZ
    - Vertiefung
      - Teil 5 der Vorlesung
      - Vector e-learning Plattform

Klassifikation von Bussystemen mit Ergänzungen

Klasse	Übertragungsraten	Anwendung	Vertreter
Klasse A	Geringe Datenraten (bis 10 kBit/s)	Vernetzung von Aktoren und Sensoren	LIN, PSI5
Klasse B	Mittlere Datenraten (bis 125 kBit/s)	Komplexe Mechanismen zur Fehlerbehandlung, Vernetzung von Steuergeräten im Komfortbereich	Lowspeed-CAN
Klasse C	Hohe Datenraten (bis 1 MBit/s)	Echtzeitanforderungen, Vernetzung von Steuergeräten im Antriebs- und Fahrwerksbereich	Highspeed-CAN
	bis zu 15 MBit/s	Wie Klasse C, zusätzlich - Mehr ECUs / CAN - Schnellere Kommunikation über lange CAN	CAN FD (Flexible Data Rate)
Klasse C+	Sehr hohe Datenraten (bis 10 MBit/s)	Echtzeitanforderungen, (Sicherheitsanforderungen,) Vernetzung von Steuergeräten im Antriebs- und Fahrwerksbereich	FlexRay
Klasse D	Sehr hohe Datenraten (ab 10 MBit/s)	Vernetzung von Steuergeräten im Telematik- und Multimediabereich, MOST verliert an Bedeutung	MOST, Ethernet

Quelle: BOSCH: Kraftfahrtechnisches Taschenbuch, Vieweg+Teubner, 28. Auflage, 2014.

Wichtige Begriffe

- Systeme
  - Das Systems Engineering versteht unter einem System eine Kombination von Elementen aus
    - Mechanik
    - Hydraulik
    - Pneumatik
    - Elektrik
    - Elektronik
      - Hardware
      - Software
    - Umwelt und Benutzer
      - Sensoren
      - Aktuatoren
      - Sollwertgeber
  - Erweiterung auf Systeme von Systemen

Wichtige Begriffe

- Allgemeiner Systembegriff
  - Allgemein versteht man unter einem System eine Menge von Teilen mit einer Beziehung zueinander.
  - Ein Softwaresystem besteht aus einer Menge von Softwareelementen:
    - Funktionen
    - Module
    - Datentypen
    - Variablen
    - …
- Struktur beschreibt
  - den Aufbau des Systems aus seinen Teilen
  - das Zusammenwirken der Teile untereinander und mit der Umwelt

Wichtige Begriffe

- Komplexität
  - wörtlich Zusammengesetztheit
  - Jedes System ist damit automatisch komplex, da es per Definition aus Teilen zusammengesetzt ist.
- Komplizierte Systeme
  - Komplex nicht gleich kompliziert
  - Kompliziert wird ein System, wenn es schwer zu verstehen ist.
  - Das kann unterschiedliche Gründe haben.

Wodurch Systeme kompliziert werden

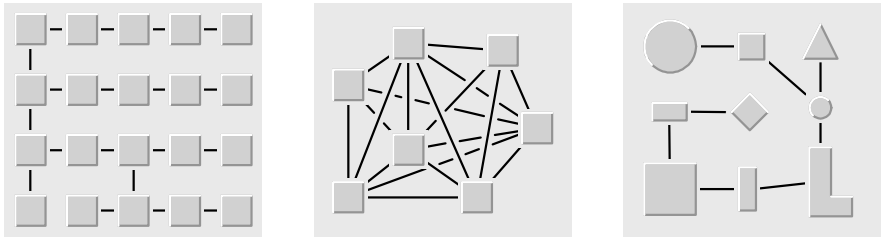


Abb. 2–1
Wodurch Systeme kompliziert werden

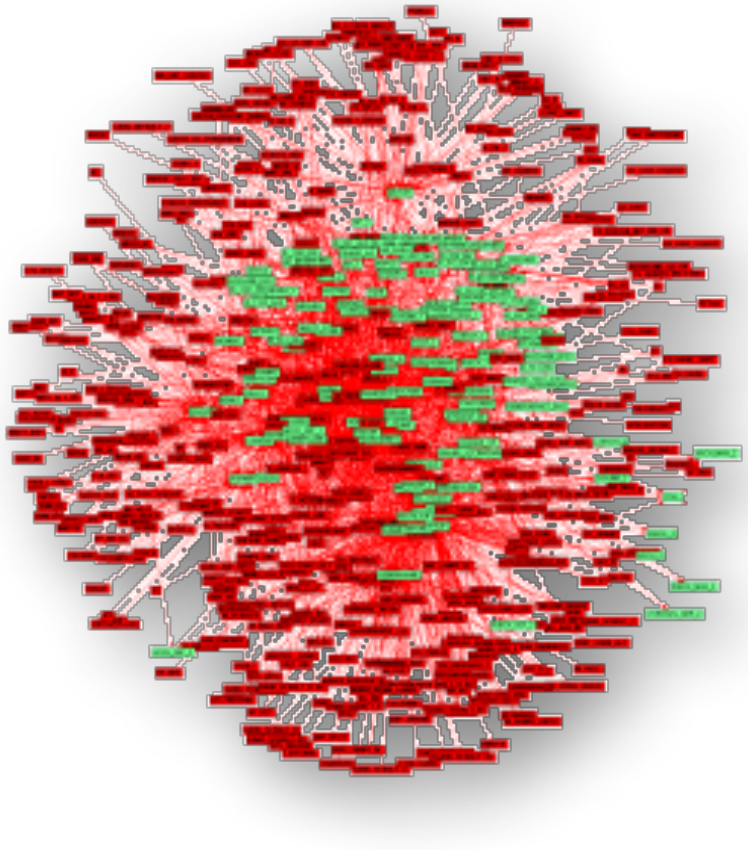
a) Das System ist sehr groß, es besteht aus vielen Teilen.
b) Das System ist stark vernetzt, es gibt viele Beziehungen zwischen den einzelnen Teilen.
c) Das System ist heterogen, es besteht aus vielen verschiedenen Arten von Elementen.

- In der Praxis a), b) und c), also sehr grosse, heterogene und stark vernetzte Systeme
- Beispiele siehe nächste beide Folien

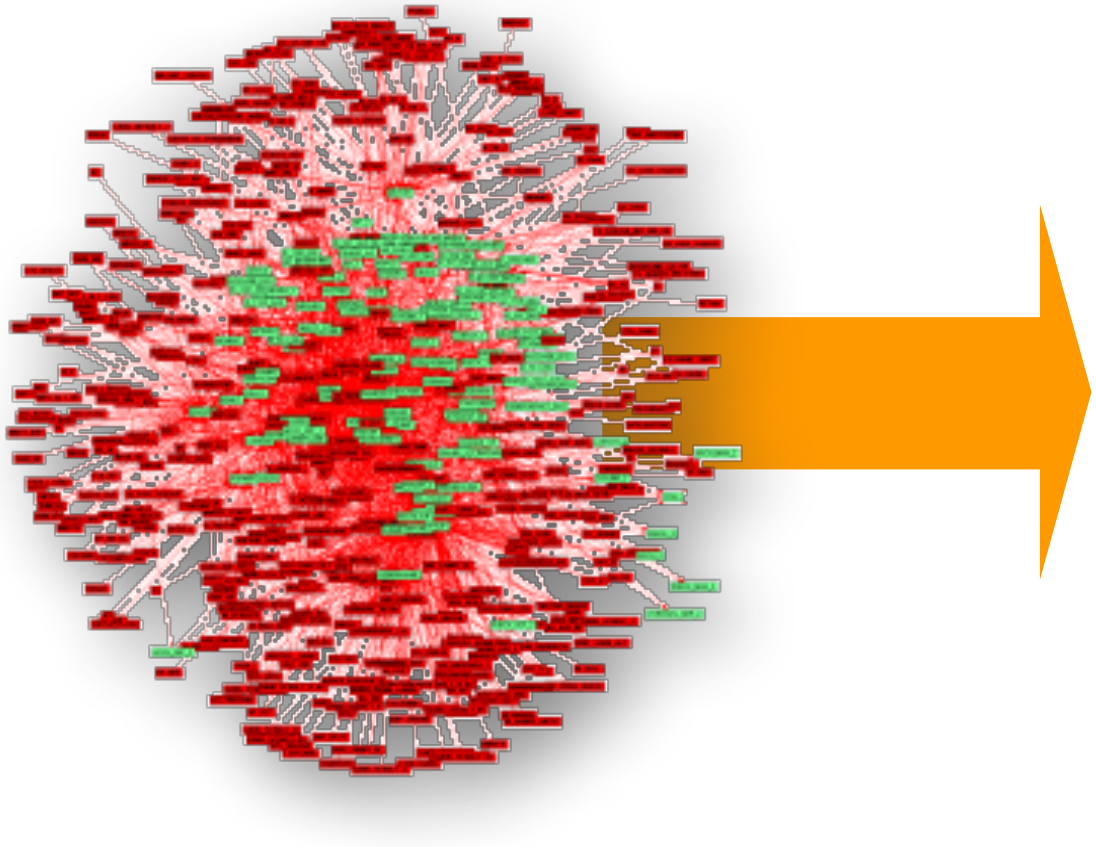
- Architektur: Beherrschung der Komplexität / Kompliziertheit

Codeanalyse und Reengineering

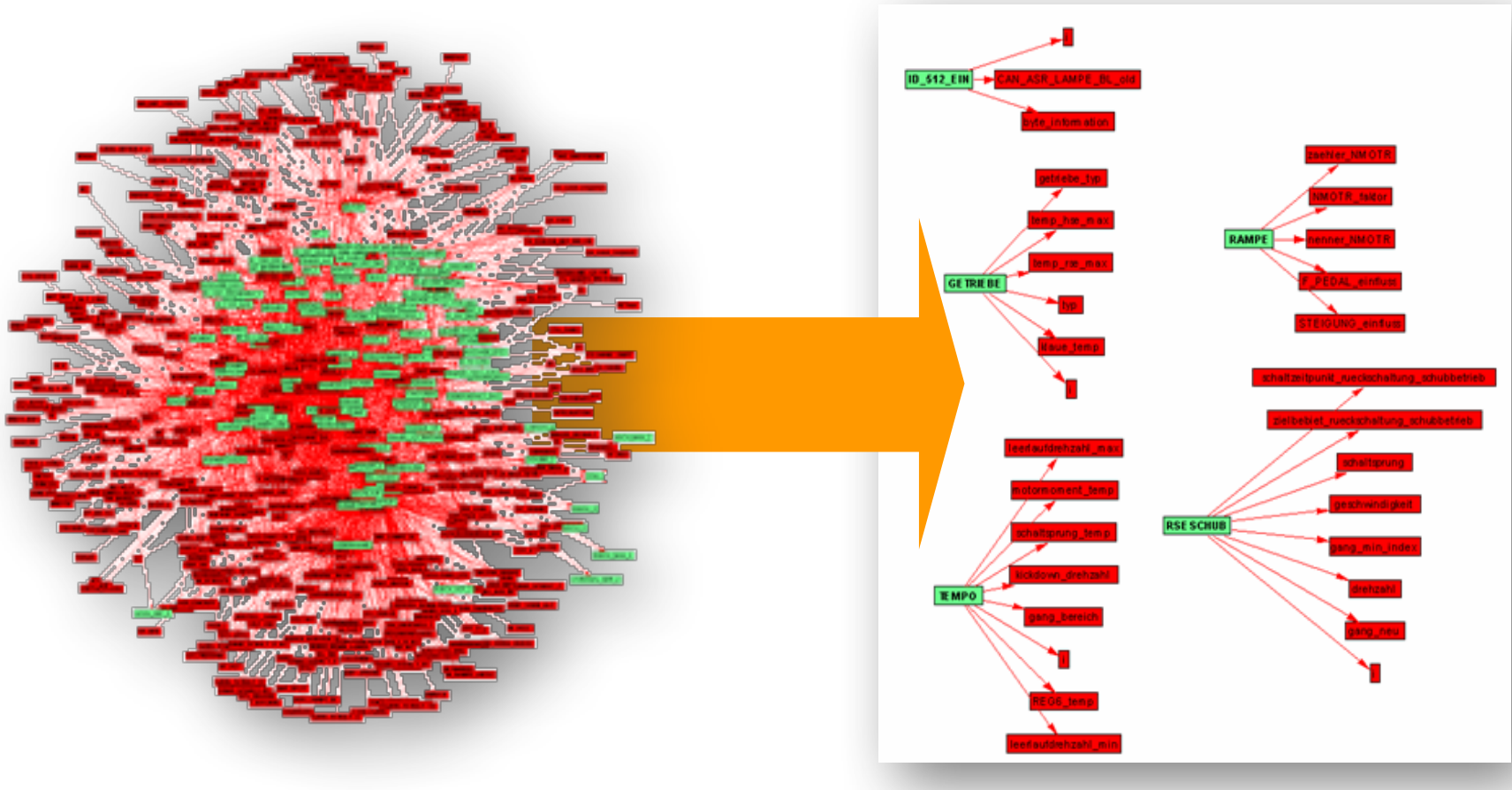
Codeanalyse und Reengineering



Codeanalyse und Reengineering

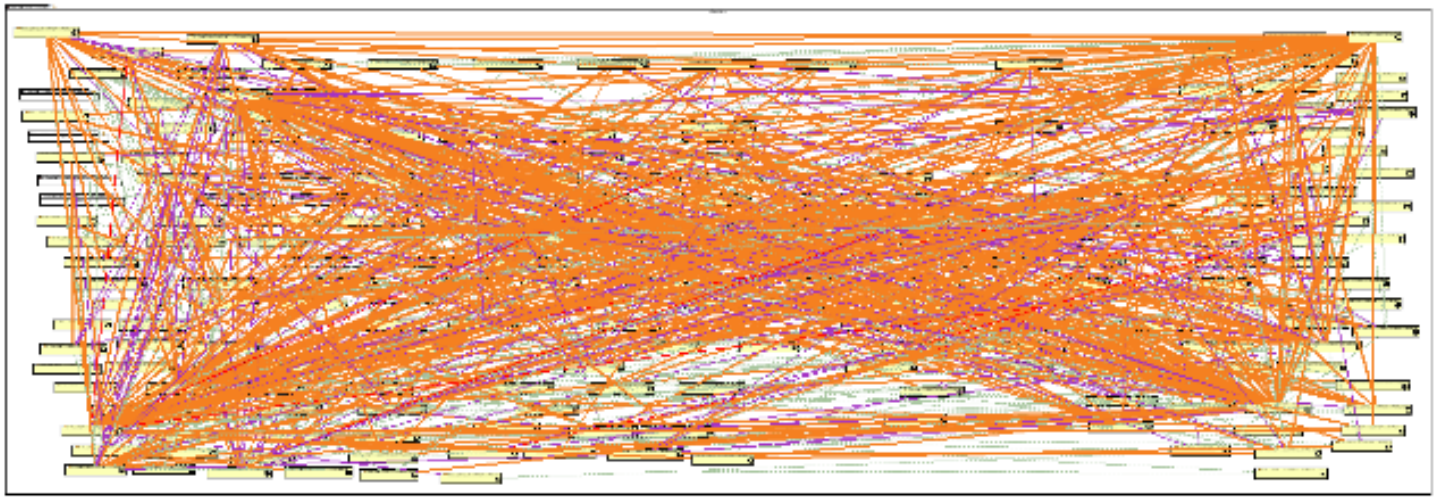


Codeanalyse und Reengineering



Structure of grown software systems

Dependencies – Visualization of the Structure



No subsystems, no predefined structure

7828 relations between 208 files (4355 relations between 121 entities*)

Study from John Deere with Fraunhofer IESE March, 2009

2.2 Was ist überhaupt Architektur?

- Softwarearchitektur
  - Keine genormte bzw. einheitlich akzeptierte Definition.
  - Vergleiche hinken
    - Gebäudearchitektur
    - Kathedralenbau
    - Festungsbau
  - Software Engineering Institute (SEI) der Carnegie Mellon University (CMU)
    - Sammlung von Definitionen für Softwarearchitektur
      - bis Anfang 2009 weit über 200 Definitionen
      - http://www.sei.cmu.edu/ architecture/ definitions.html
  - IEEE Std 1471
    - „Architekturbeschreibung in Software-intensiven Systemen“
    - „Recommended Practice“, keine Norm.
  - ISO/IEC 42010
    - Erweiterung des IEEE- Standards 1471 für Systemarchitekturbeschreibungen.

2.2 Was ist überhaupt Architektur?

- In diesem Lehrgang gem. Buch Kindel/Friedrich
  - Definition: Softwarearchitektur
    Softwarearchitektur ist die oberste Strukturebene eines großen Softwaresystems.
  - Nach SEI moderne Definition der Softwarearchitektur

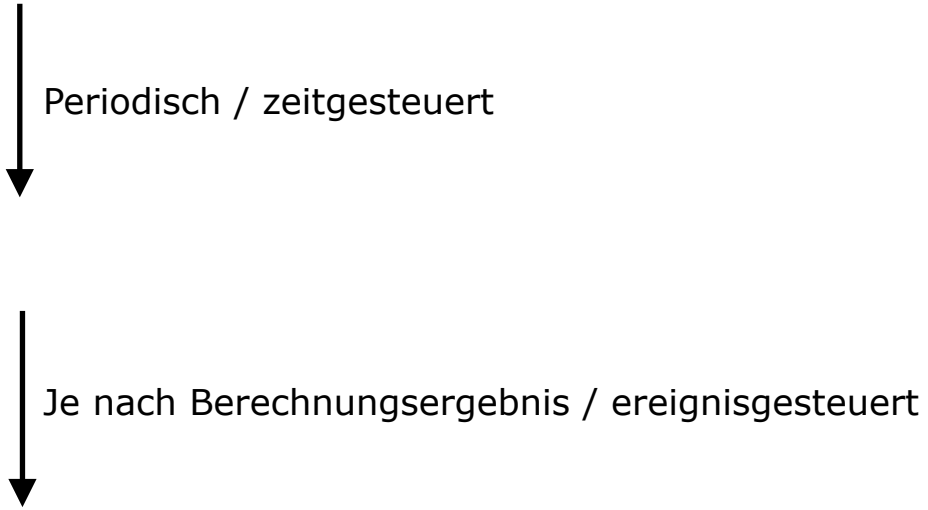
- Architektur hat etwas mit Struktur zu tun.
- Auf einer Strukturebene bilden die Elemente dieser Struktur ein Netzwerk.
- Struktur heißt nicht nur statischer Aufbau, sondern auch dynamisches Verhalten (wer ruft was auf?).
- Jedes Element einer Strukturebene kann für sich auch wieder eine Architektur haben.
- In dieser Hierarchie interessiert den Architekten zunächst nur die obere Ebene. (Der IEEE-Standard 1471 nennt das die „fundamentale Organisation des Systems“.)
- Triviale Algorithmen und Datentypen haben keine Architektur. Sie bilden den Abschluss der Hierarchie, sind also gewissermaßen Architekturatome.
- Oder umgekehrt: Architektur entsteht auf einer abstrakten Ebene oberhalb von Algorithmen und integralen Datentypen.

2.2 Was ist überhaupt Architektur?

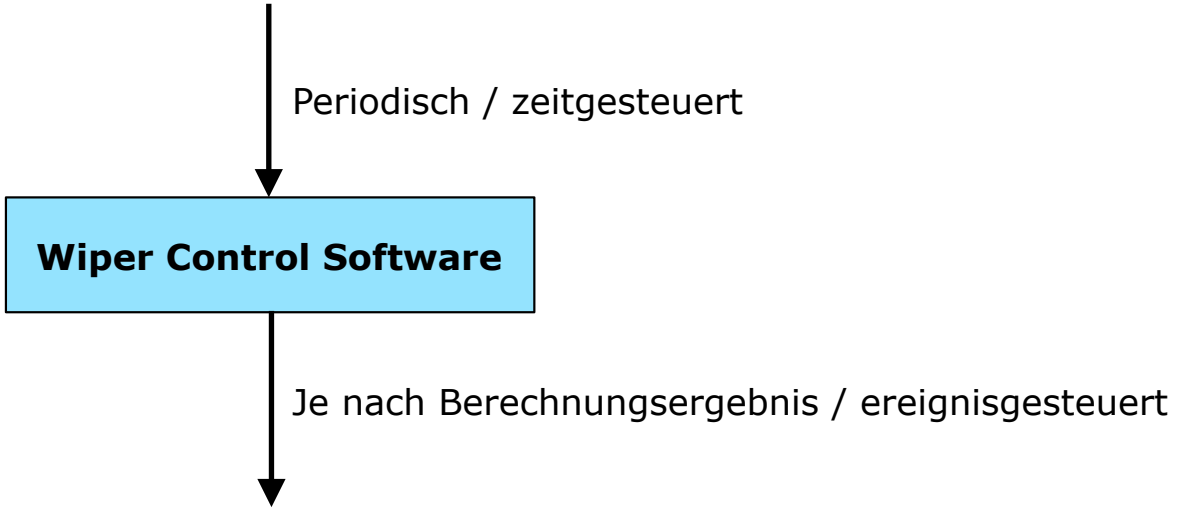
- In diesem Lehrgang gem. Buch Kindel/Friedrich
  - Definition: Softwarearchitektur
    Softwarearchitektur ist die oberste Strukturebene eines großen Softwaresystems.
  - Nach SEI moderne Definition der Softwarearchitektur

- Architektur hat etwas mit Struktur zu tun.
- Auf einer Strukturebene bilden die Elemente dieser Struktur ein Netzwerk.
- Struktur heißt nicht nur statischer Aufbau, sondern auch dynamisches Verhalten (wer ruft was auf?). Wann? Warum?
- Jedes Element einer Strukturebene kann für sich auch wieder eine Architektur haben.
- In dieser Hierarchie interessiert den Architekten zunächst nur die obere Ebene. (Der IEEE-Standard 1471 nennt das die „fundamentale Organisation des Systems“.)
- Triviale Algorithmen und Datentypen haben keine Architektur. Sie bilden den Abschluss der Hierarchie, sind also gewissermaßen Architekturatome.
- Oder umgekehrt: Architektur entsteht auf einer abstrakten Ebene oberhalb von Algorithmen und integralen Datentypen.

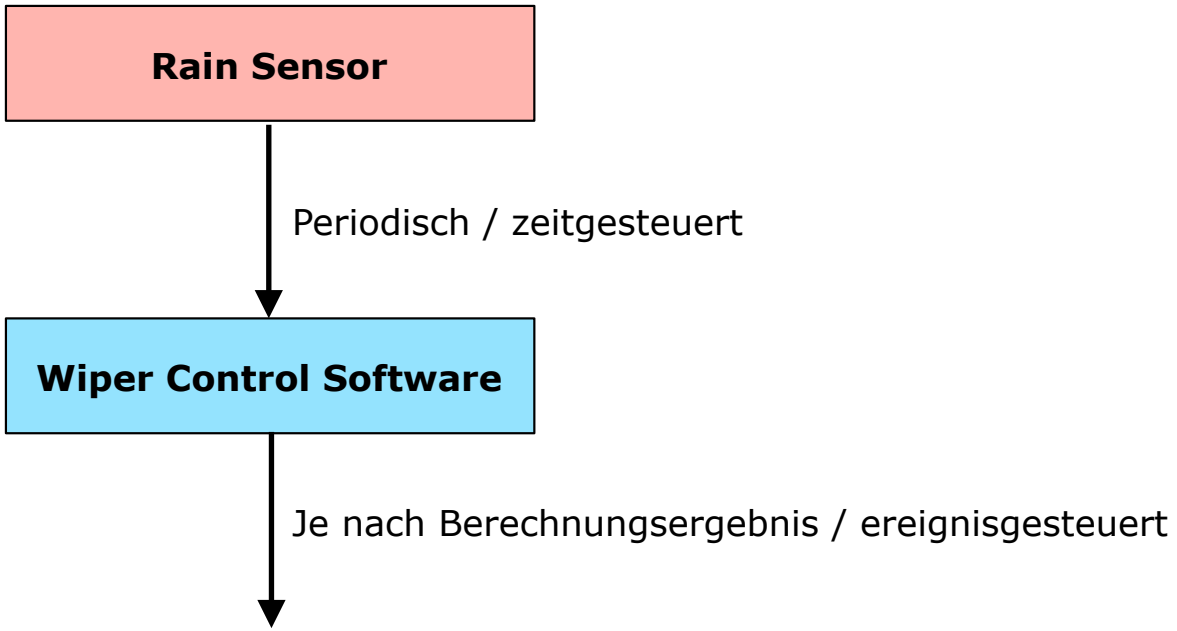
Wann? Warum?


Periodisch / zeitgesteuert
Je nach Berechnungsergebnis / ereignisgesteuert

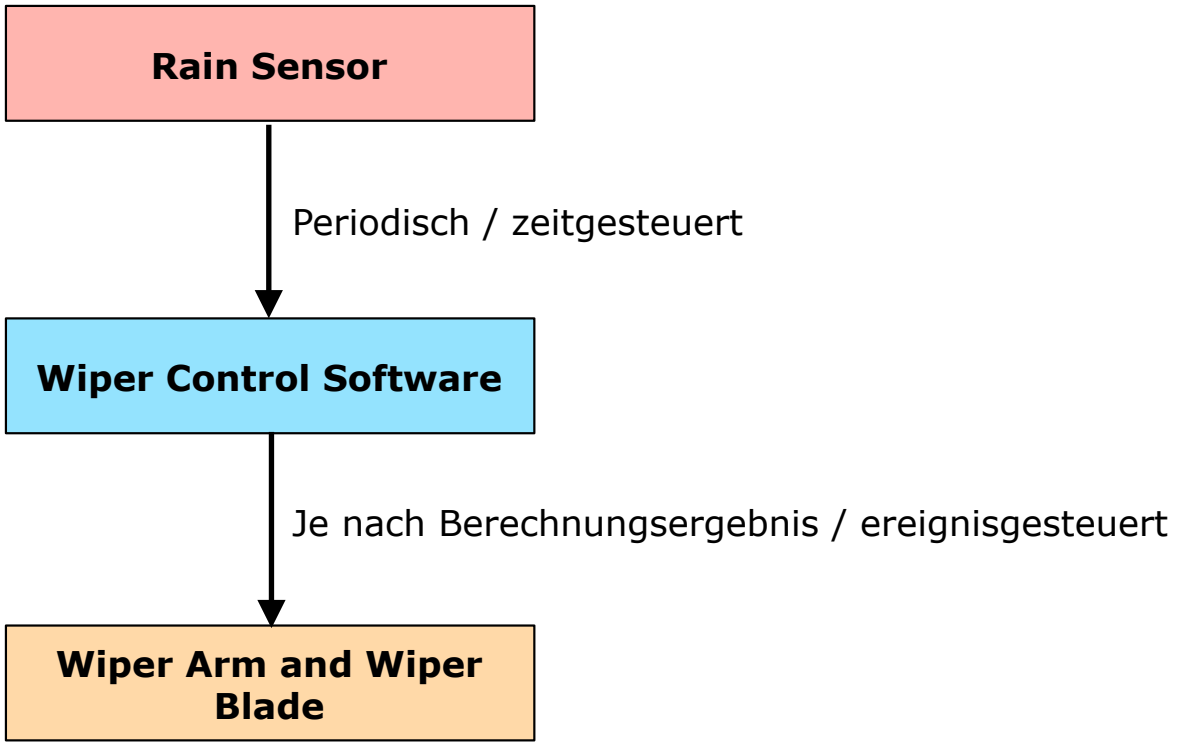
Wann? Warum?


Periodisch / zeitgesteuert
Wiper Control Software
Je nach Berechnungsergebnis / ereignisgesteuert

Wann? Warum?


Rain Sensor
Periodisch / zeitgesteuert
Wiper Control Software
Je nach Berechnungsergebnis / ereignisgesteuert

Wann? Warum?


Rain Sensor
Periodisch / zeitgesteuert
Wiper Control Software
Je nach Berechnungsergebnis / ereignisgesteuert
Wiper Arm and Wiper Blade

Example of a simple CPS

- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS

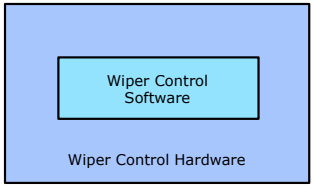
- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS

Wiper Control Software

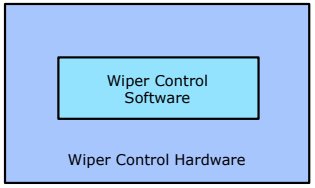
- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

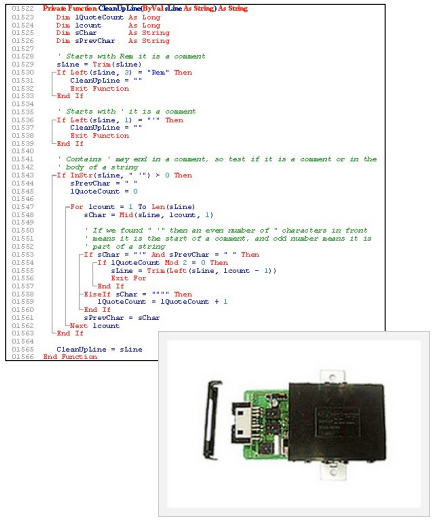
Example of a simple CPS



- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

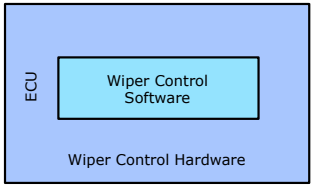
Example of a simple CPS

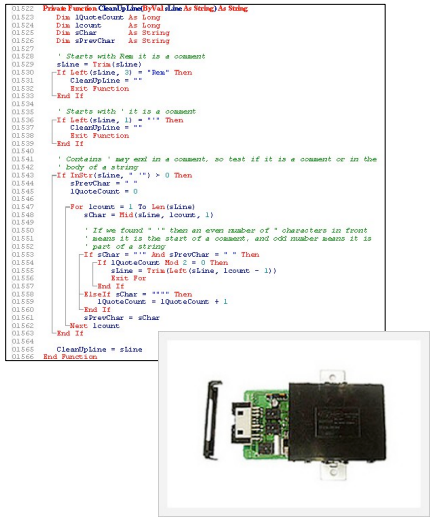




- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

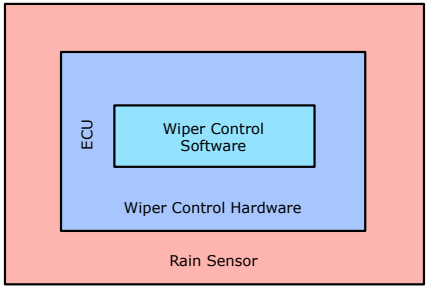
Example of a simple CPS

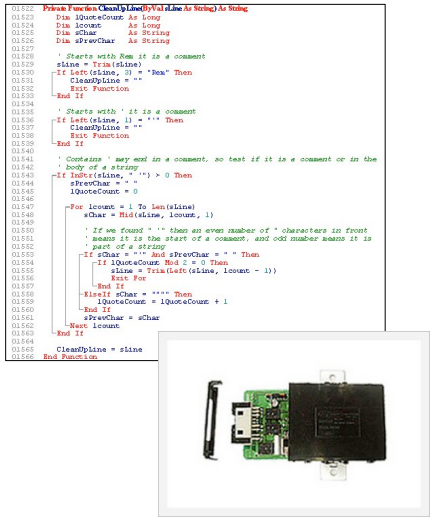




- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

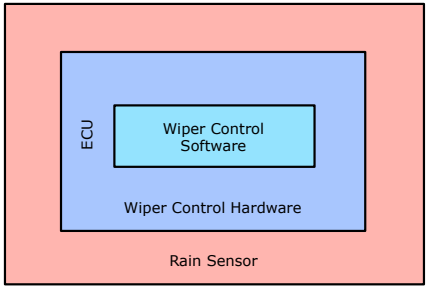
Example of a simple CPS

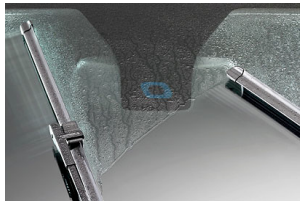




- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

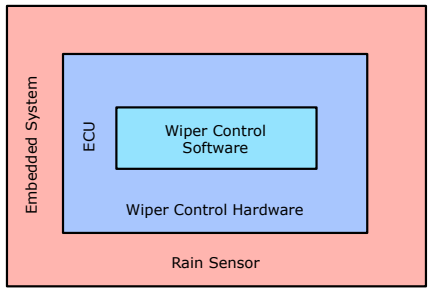
Example of a simple CPS





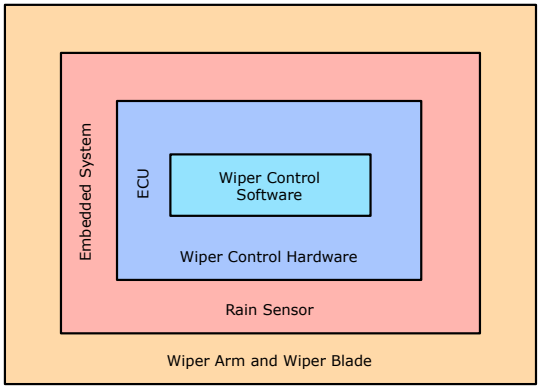
- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

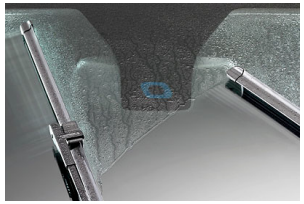
Example of a simple CPS



- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

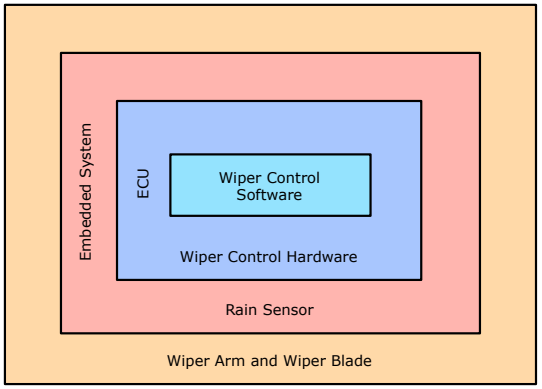
Example of a simple CPS

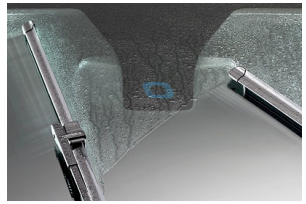




- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

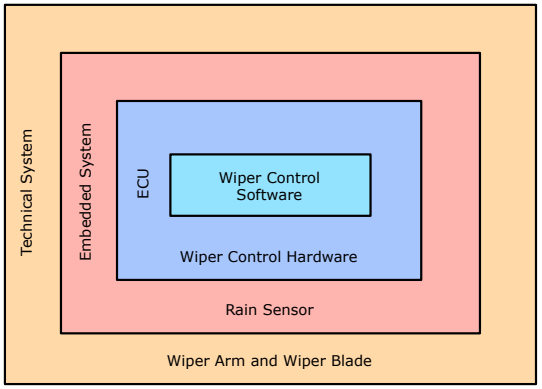
Example of a simple CPS

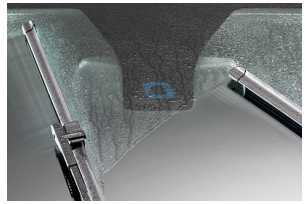




- Based on „Onion ring like structure of CPS (modified)“ Source: Manfred Broy

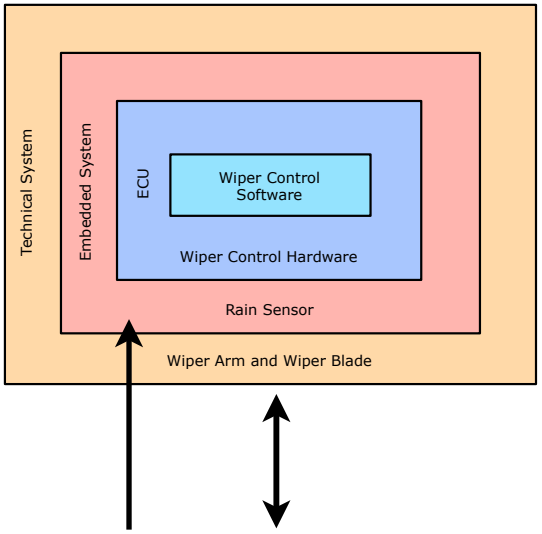
Example of a simple CPS



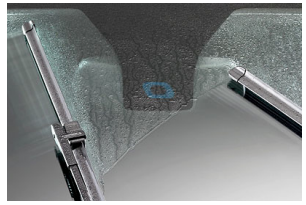


- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS

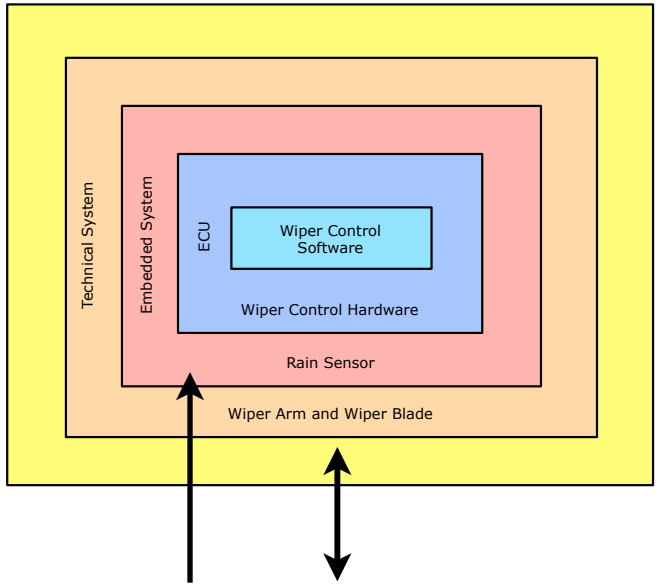


Interaction with physical environment

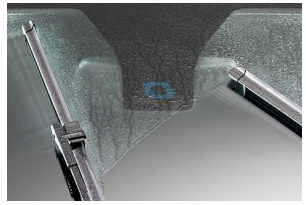


- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS

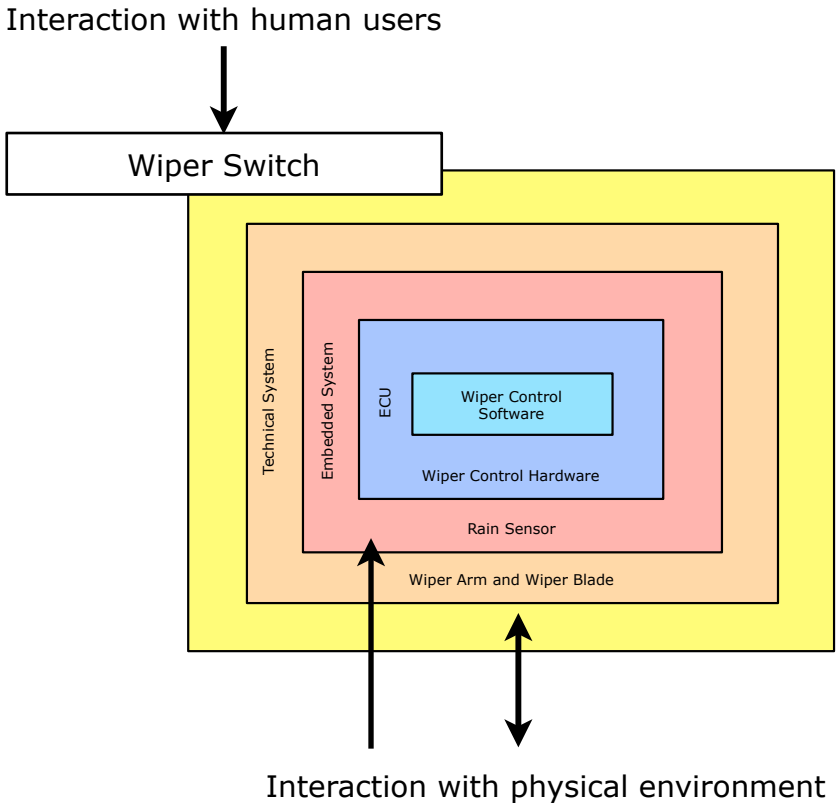


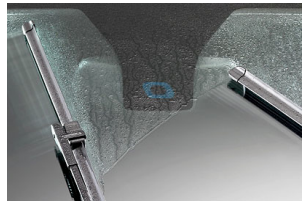
Interaction with physical environment



- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

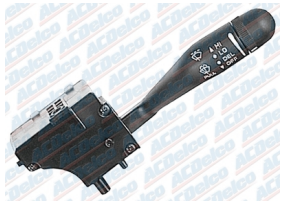
Example of a simple CPS

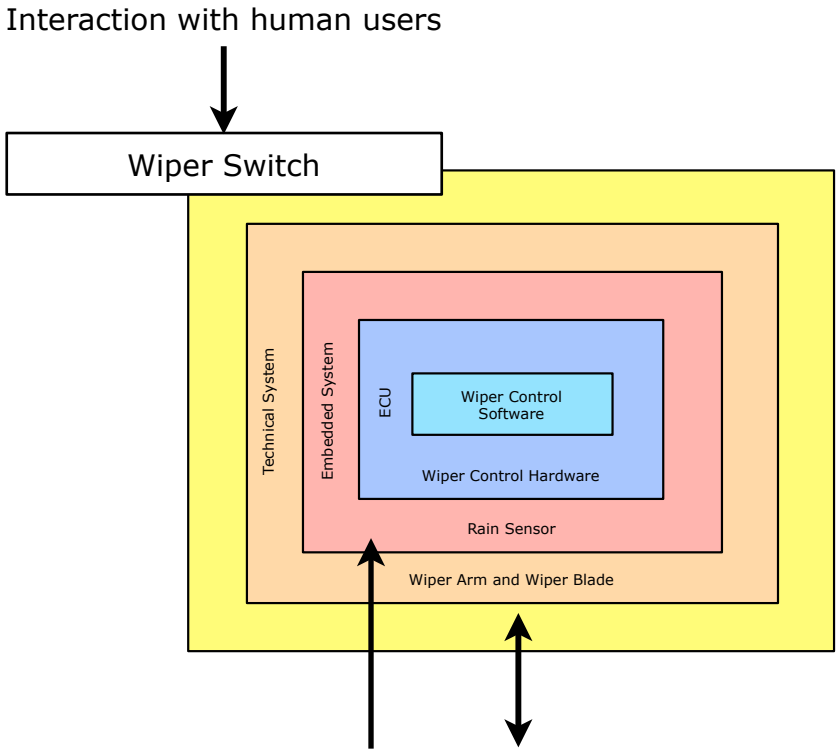


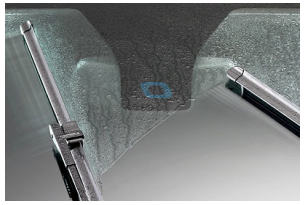


- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS

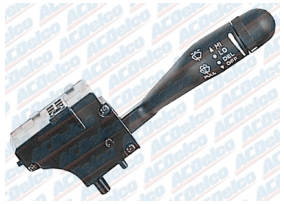


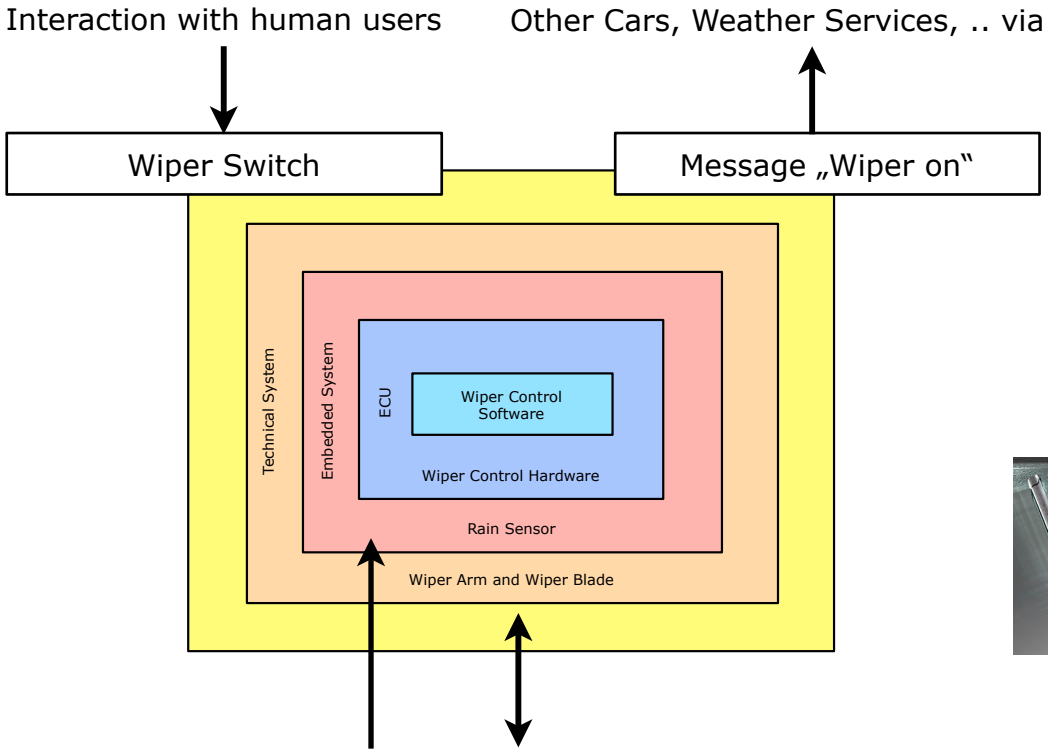


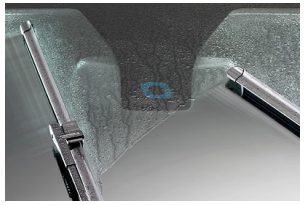


- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS

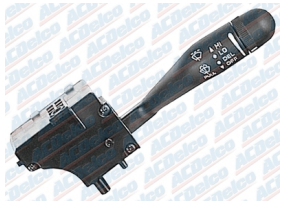


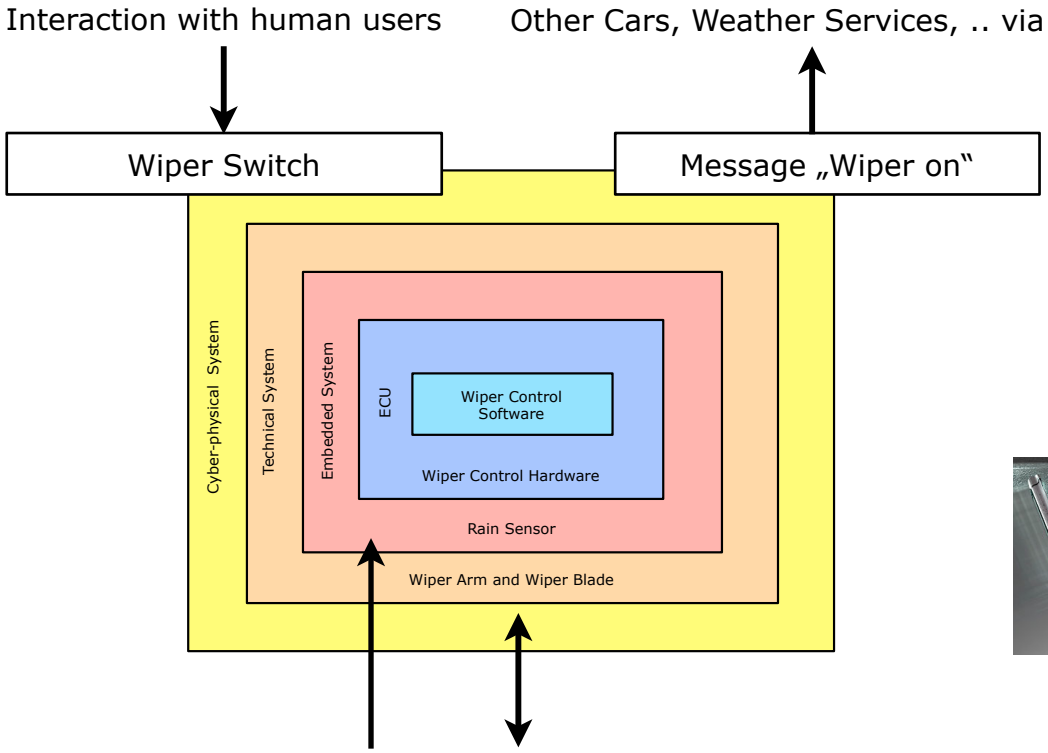


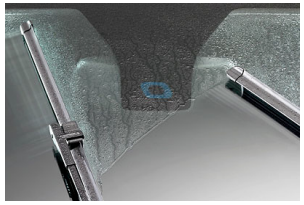


- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Example of a simple CPS







- Based on „Onion ring like structure of CPS (modified)“
  Source: Manfred Broy

Darstellungen von Struktur

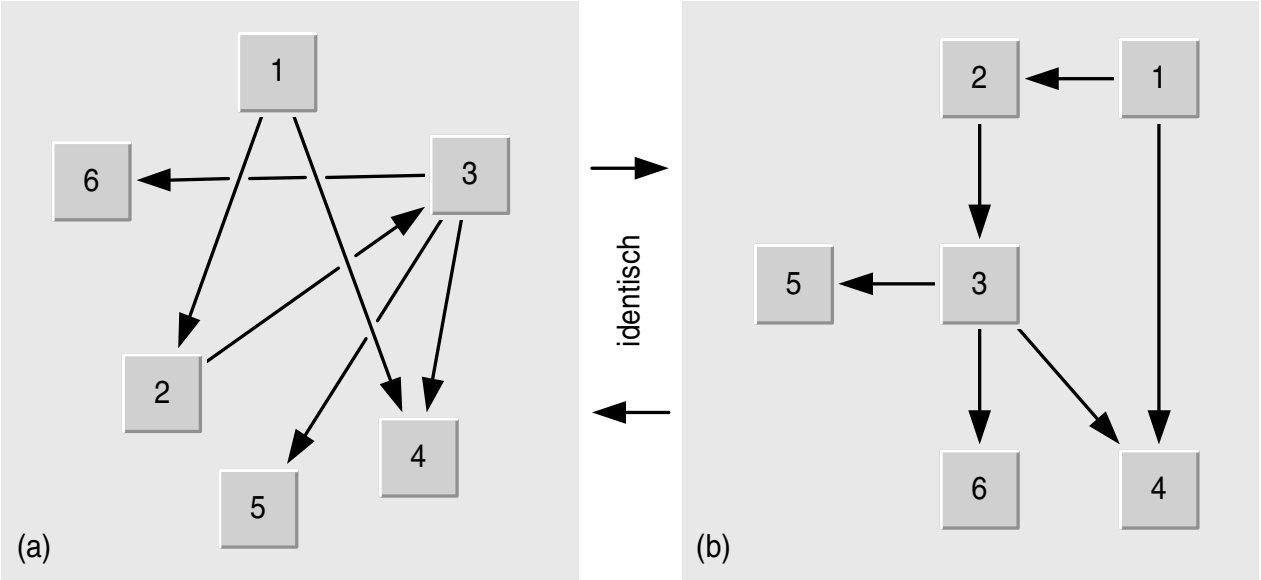


Abb. 2–2

Zwei unterschiedliche Darstellungen der gleichen Struktur

2.3 Architektur in der Softwaretechnik

- Softwareentwicklungsprozess
  - Architektur ist das Bindeglied zwischen Anforderungen und ausführbaren Code
- Systemanalyse
  - Das durch die Kundenanforderungen beschriebene Softwaresystem wird in seine Bestandteile zerlegt (griech. analysiert).
  - Anschließend werden die Beziehungen dieser Bestandteile zueinander formal beschrieben.
  - Softwarearchitektur ist also keine Tätigkeit, sondern vielmehr ein Arbeitsergebnis oder ein Output der Systemanalyse.
  - Die Softwareanforderungen sind der zugehörige Input.
- Über Architektur und Design sind Anforderungen und Code miteinander verkettet.
  - Die Architektur liefert zunächst eine geeignete Partitionierung der Anforderungen in Architekturelemente
  - Das Design formuliert anschließend Prinzipien für die Realisierung in implementierbaren Modulen (siehe auch Tabelle 2–1).

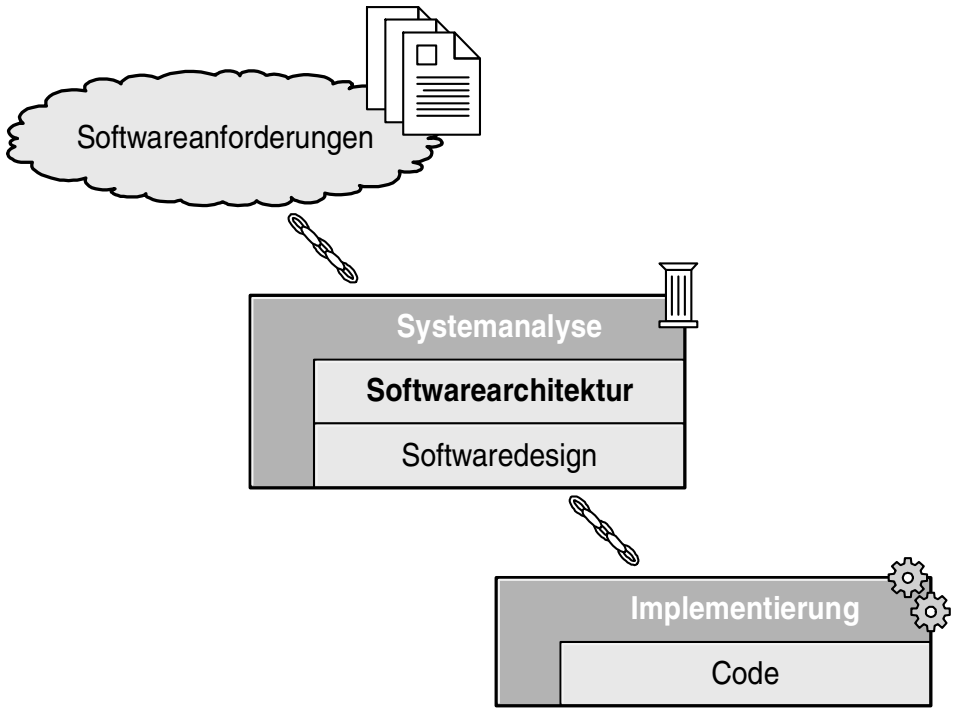
2.3 Architektur in der Softwaretechnik

- Softwaretechnik verbindet zwei Welten, in denen völlig unterschiedliche Sprachen gesprochen werden.
  - Anforderungen
    - Natürliche Sprache
    - Formal
      - XML
      - Logik
  - Code / Implementierung
    - Programmiersprache
      - Direkt
      - Modellierung z. B. ML/SL und Code-Generierung

Architektur und Design

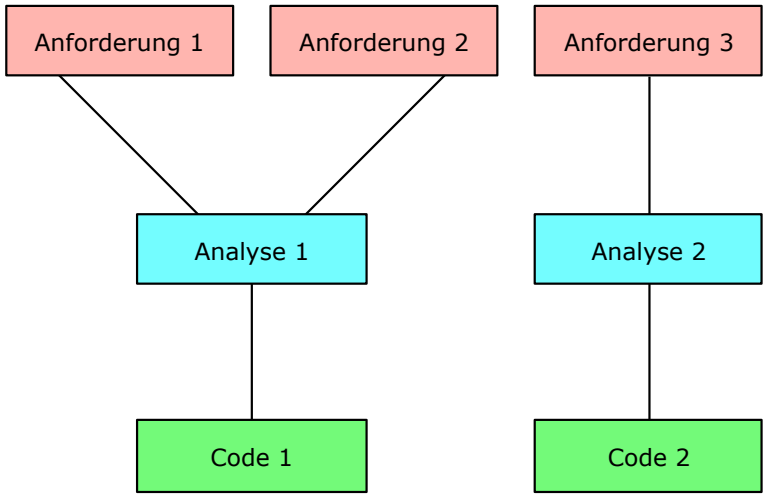
Abb. 2–3

Architektur und Design verketten Anforderungen und Code



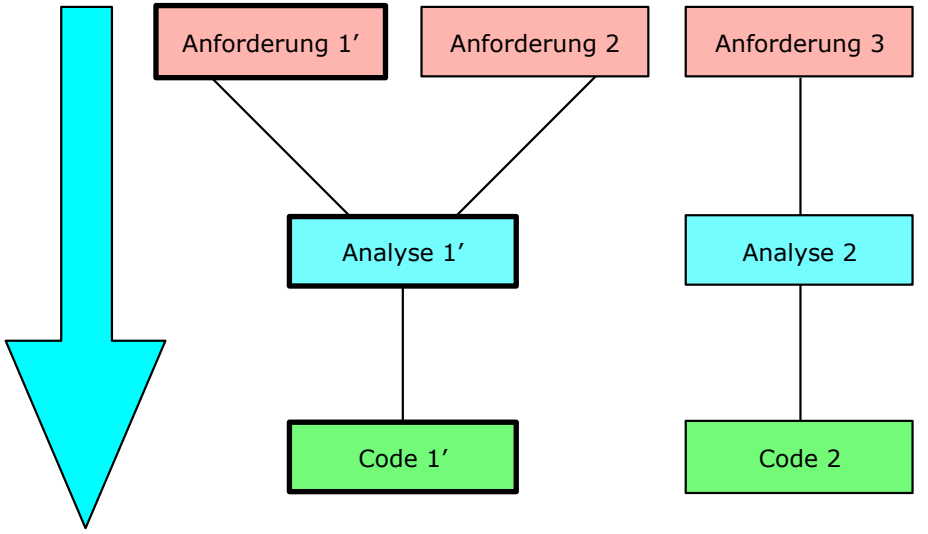
Anforderungen - Analyse - Code

- Jedes Codeelement hat eine Legitimation durch ein Element der Analyse.
- Jedes Element der Analyse existiert, weil es ein oder mehrere Anforderungen erfüllt.
- Damit sind für jedes Codeelement die zugehörigen Anforderungen bekannt.
- Anforderungsänderung
  - Welche Codeelemente sind betroffen?
  - Ausgehend vom Codeelement
    - Wechselwirkung zu anderen Anforderungen,die das Codeelement ebenfalls erfüllen muss (Impact-Analyse).



Anforderungen - Analyse - Code

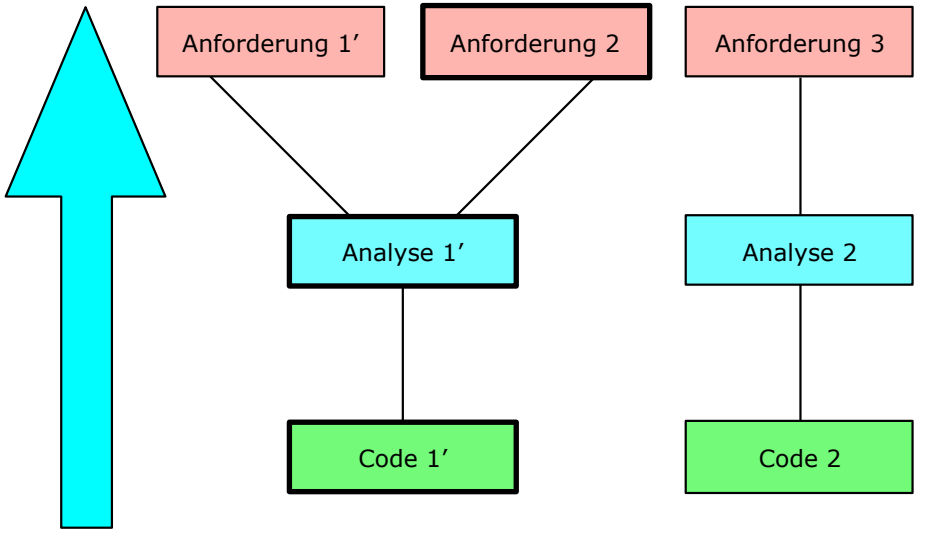
- Jedes Codeelement hat eine Legitimation durch ein Element der Analyse.
- Jedes Element der Analyse existiert, weil es ein oder mehrere Anforderungen erfüllt.
- Damit sind für jedes Codeelement die zugehörigen Anforderungen bekannt.
- **Anforderungsänderung**
  - Welche Codeelemente sind betroffen?
  - Ausgehend vom Codeelement
    - Wechselwirkung zu anderen Anforderungen,die das Codeelement ebenfalls erfüllen muss (Impact-Analyse).



Anpassung Code 1' an Anforderung 1'

Anforderungen - Analyse - Code

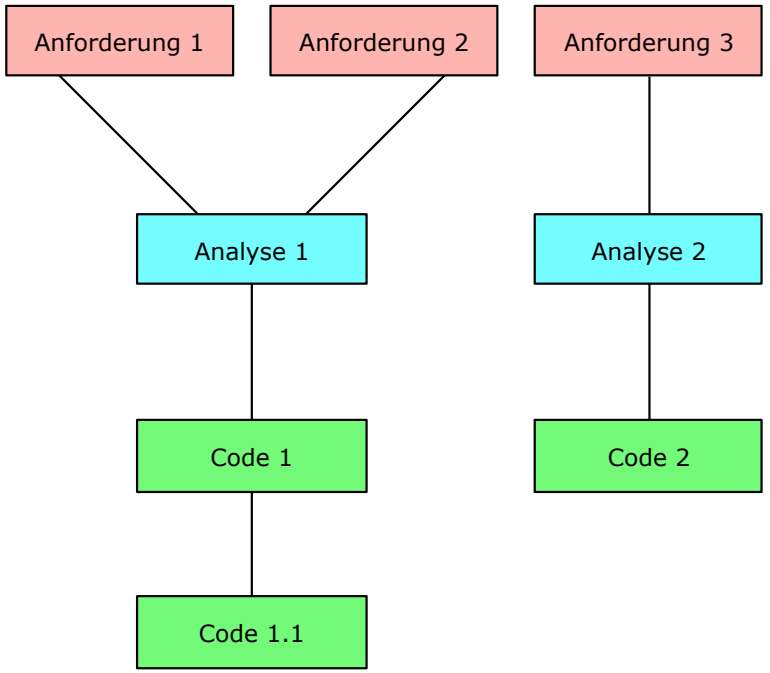
- Jedes Codeelement hat eine Legitimation durch ein Element der Analyse.
- Jedes Element der Analyse existiert, weil es ein oder mehrere Anforderungen erfüllt.
- Damit sind für jedes Codeelement die zugehörigen Anforderungen bekannt.
- Anforderungsänderung
  - Welche Codeelemente sind betroffen?
  - Ausgehend vom Codeelement
    - Wechselwirkung zu anderen Anforderungen,die das Codeelement ebenfalls erfüllen muss (**Impact-Analyse**).



Erfüllt Code 1' noch Anforderung 2?

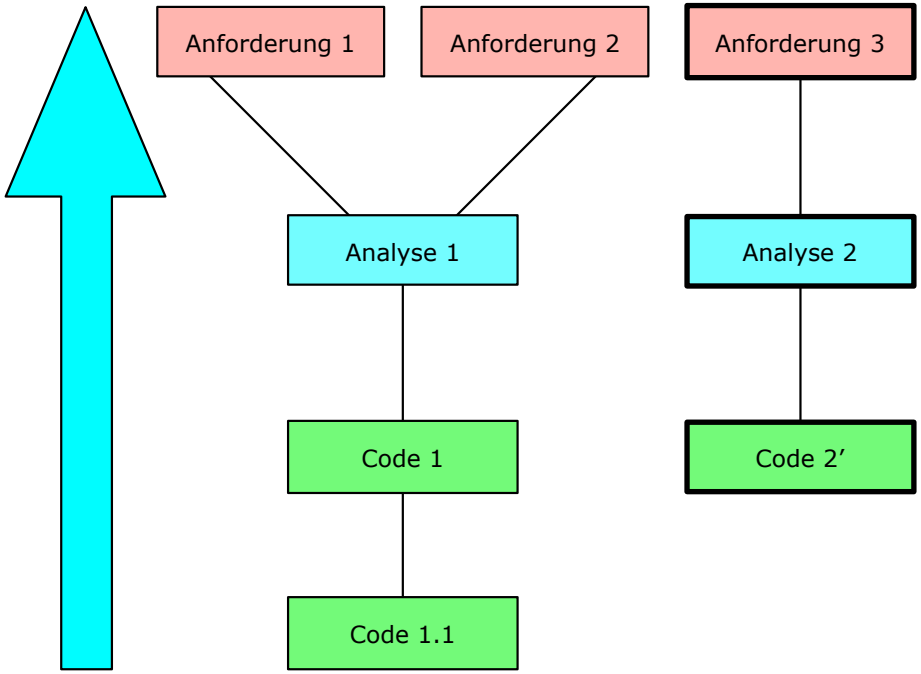
Anforderungen - Analyse - Code

- Alle aktuellen Prozessmodelle:
  - Forderung der Nachverfolgbarkeit von der Architektur zum Code und auch wieder zurück (bidirektionale Traceability).
  - Voraussetzung: Die Architektur bleibt bei Änderungen vollständig und konsistent
    - Keine unkontrollierten Änderungen am Code oder gar an der Modulstruktur.



Anforderungen - Analyse - Code

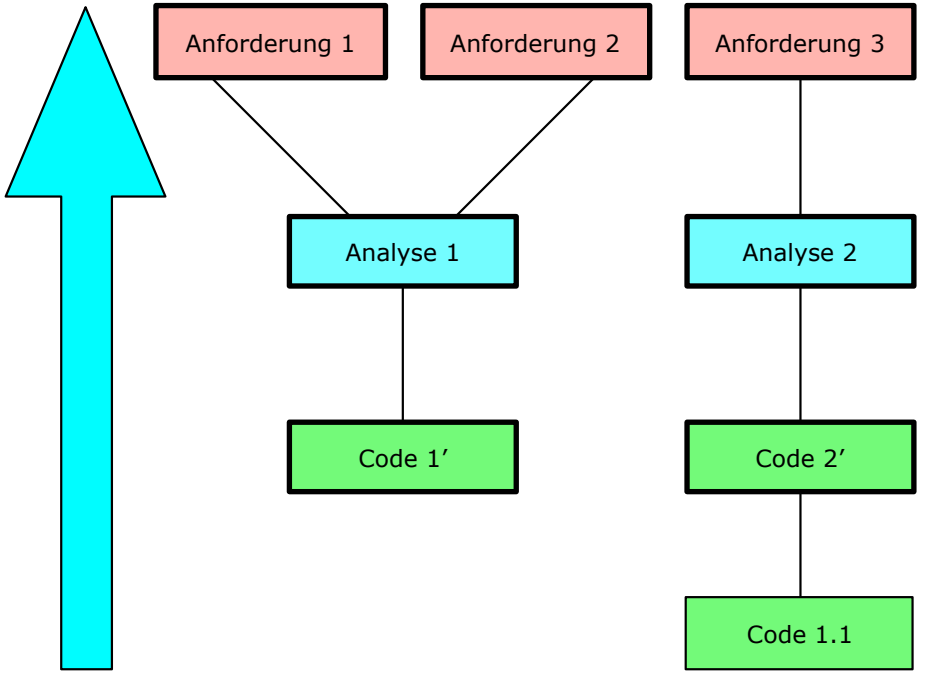
- Alle aktuellen Prozessmodelle:
  - Forderung der Nachverfolgbarkeit von der Architektur zum Code und auch wieder zurück (bidirektionale Traceability).
  - Voraussetzung: Die Architektur bleibt bei Änderungen vollständig und konsistent
    - **Keine unkontrollierten Änderungen am Code** oder gar an der Modulstruktur.



Erfüllt Code 2’ noch Anforderung 3?

Anforderungen - Analyse - Code

- Alle aktuellen Prozessmodelle:
  - Forderung der Nachverfolgbarkeit von der Architektur zum Code und auch wieder zurück (bidirektionale Traceability).
  - Voraussetzung: Die Architektur bleibt bei Änderungen vollständig und konsistent
    - **Keine unkontrollierten Änderungen** am Code oder gar **an der Modulstruktur.**



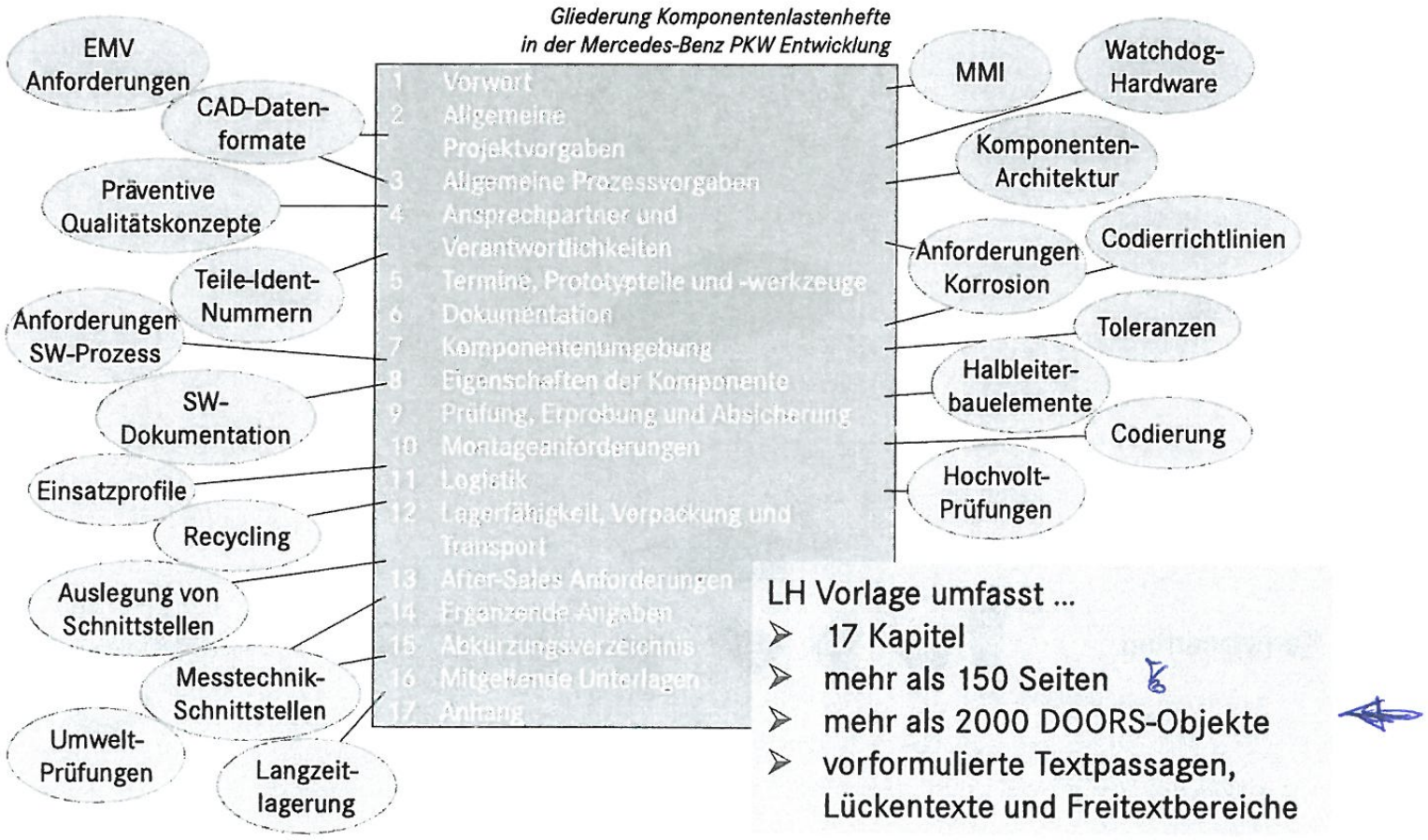
Erfüllt Code 1' noch Anforderung 1 und Anforderung 2?
Erfüllt Code 2' noch Anforderung 3?

Wie sieht es in der Praxis aus?

- Quelle:
  Durchgängiges Anforderungsmanagement vom Fahrzeug bis zur Komponente,
  Dr. Matthias Recknagel, Daimler AG,
  Ringvorlesung: Forum Software und Automatisierung,
  IAS, Universität Stuttgart, 9. Januar 2014

REM

Komponentenlastenhefte – Aufbau und Inhalt



12 Requirements Engineering für Automotive Systeme / 30.1.2013 Mercedes-Benz

Standards Recycl's
gesetze, Materialien, Kennzeichnungspflicht
z.B. für Technik 5 Jahre 25 Tiere

Requirements Engineering History at Mercedes-Benz: A Guided Tour

REM

Zahlen, Daten, Fakten
Lastenhefte @ Mercedes-Benz

Reifen

Headunit

Umfang

Seitenzahl

60 — 600 — 2.000

Anzahl der DOORS-Objekte

1.000 — 50.000

Anzahl referenzierter Dokumente

30 — 300

40.000 Seiten relevant worst case

Häufigkeit

Häufigkeit Lastenhefterstellung

Vollzeit-Spezifikateur — Ein Lastenheft in drei Jahren

Änderungsrate und -häufigkeit

Eine Version mit mit kleineren Änderungen — > 10 Versionen , viele kleinere Änderungen

Lastenheft-Vorlage

- 17 Kapitel
- > 150 Seiten
- > 2000 DOORS Objekte
- Viele Standard-Formulierungen

1 Vorwort
2 Allgemeine Projektvorgaben
3 Allgemeine Prozessvorgaben
4 Ansprechpartner und Verantwortlichkeiten
5 Termine, Prototypteile und -werkzeuge
6 Dokumentation
7 Komponentenumgebung
8 Eigenschaften der Komponente
9 Prüfung, Erprobung und Absicherung
10 Montageanforderungen
11 Logistik
12 Lagerfähigkeit, Verpackung und Transport
13 After-Sales Anforderungen
14 Ergänzende Angaben
15 Abkürzungsverzeichnis
16 Mitgeltende Unterlagen
17 Anhang

Spezifikationsinhalte

- Überwiegend natürlichsprachlich (German, English)
- Tabellen und Zeichnungen nach Bedarf
- Formale Spezifikation bei
  - ausführbaren Modellen (Matlab/Simulink)
  - CAD Zeichnungen

13 Improving Requirements Engineering Processes | Frank Houdek | PROFES, 14.6.2012

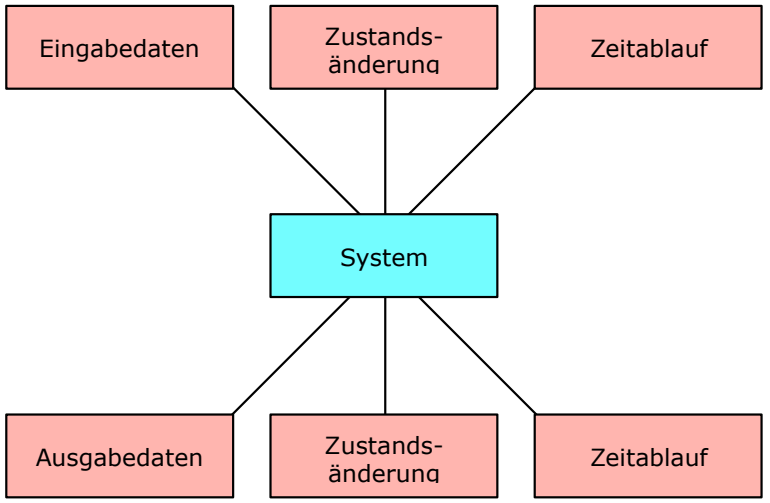
Mercedes-Benz

2.3.1 Anforderungstypen

- Drei Arten von Anforderungen
  - Funktionale Anforderungen
  - Entwurfsanforderungen
  - Rahmenanforderungen
- Entwurfsanforderungen und Rahmenanforderungen werden oft als nichtfunktionale Anforderungen bezeichnet.
- Beispiel
  - Nichtfunktionale Entwurfsanforderung:
    „Das System muss an neue XYZ-Controller anpassbar sein«
  - Realisierung: Spezifikation von Treibern und Schnittstellen
    Umsetzung der nichtfunktionalen Entwurfsanforderung in funktionale Anforderungen
- Allgemein: Nichtfunktionale Anforderung = Sammlung funktionaler Anforderungen
  - noch nicht funktionale Anforderung
  - unterspezifizierte funktionale Anforderung

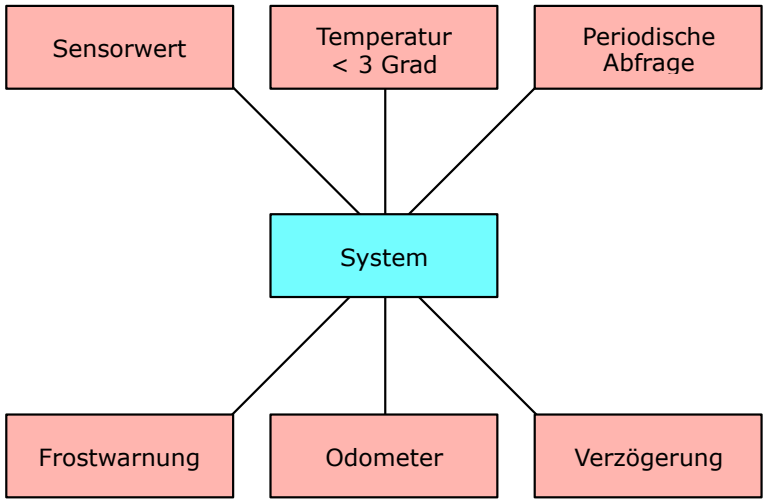
Funktionale Anforderungen

- Eine funktionale Anforderung beschreibt detailliert, was das System bei einem konkreten Ereignis leisten soll. Mögliche Ereignisse sind eine Eingabe, eine Zustandsänderung oder auch ein Zeitablauf. Das Ergebnis kann eine Ausgabe oder andere Aktion sein. Gute funktionale Anforderungen beschreiben, was vom System erwartet wird, und geben noch keine Lösung vor. Die funktionalen Anforderungen spezifizieren:
  - die Eignung für den Einsatzzweck
  - Interoperabilität (Zusammenarbeit mit anderen Systemen)
  - Sicherheit: Security (Datensicherheit)



Funktionale Anforderungen

- Eine funktionale Anforderung beschreibt detailliert, was das System bei einem konkreten Ereignis leisten soll. Mögliche Ereignisse sind eine Eingabe, eine Zustandsänderung oder auch ein Zeitablauf. Das Ergebnis kann eine Ausgabe oder andere Aktion sein. Gute funktionale Anforderungen beschreiben, was vom System erwartet wird, und geben noch keine Lösung vor. Die funktionalen Anforderungen spezifizieren:
  - die Eignung für den Einsatzzweck
  - Interoperabilität (Zusammenarbeit mit anderen Systemen)
  - Sicherheit: Security (Datensicherheit)



Entwurfsanforderungen

- Die Entwurfsanforderungen spezifizieren:
  - Effizienz (Timing, Ressourcenausnutzung)
  - Wartbarkeit (Änderbarkeit, Zerlegbarkeit, Testbarkeit, Stabilität)
  - Portabilität (Anpassbarkeit, Installierbarkeit, Austauschbarkeit, Koexistenz mit anderen Modulen)
  - Sicherheit: Safety (Betriebssicherheit)
  - Ergonomie (Übersichtlichkeit, Verständlichkeit, Bedienbarkeit)
- Einfluss auf Architektur
  - Zerlegbarkeit
  - Austauschbarkeit
  - Zusätzliche Schnittstellen und Module

Rahmenanforderungen

- Die Rahmenanforderungen spezifizieren:
  - Fertigungs- und Integrationsanforderungen
  - Anforderungen an die Umgebungsbedingungen beim Betrieb
  - einzuhaltende Vorschriften, Normen und Gesetze.
- Diese Anforderungen betreffen oft den anzuwendenden Entwicklungsprozess und die von ihm festgelegten Tätigkeiten sowie anzuwendende Normen und Standards.
- Auf das beobachtbare Verhalten des fertigen Systems („Funktionale Eigenschaften“) haben sie keinen direkten Einfluss.

2.3.2 Abgrenzung zur Systemarchitektur

- Systemarchitektur: Verteilung der Anforderungen an
  - Mechanik
  - Hardware
  - Software
  - …
- Entgegen dem ersten Eindruck: Nicht immer eindeutig
- Beispiel Qualitätsanforderung
  „Das System muss einen Ruhestrom von weniger als 20mA aufweisen“
  - Zunächst zweifelsfrei Anforderung an Elektrik
  - Wenn der geforderte Ruhestrom aber nur durch einen Sleep-Modus der CPU erreicht werden kann, ist auch Softwareunterstützung gefordert. Der Sleep-Modus muss mit Eintreten der Ruhebedingung aktiviert werden und – was häufig noch anspruchsvoller ist – nach Ende der Ruhebedingung auch wieder fehlerfrei verlassen werden.

2.3.3 Konzepte über alles

- Zunächst lesenswerte Bonmots zu Grobkonzept, Feinkonzept etc.
- Die Systemanalyse liefert detaillierte Informationen, die eine große Gruppe von Stakeholdern über die gesamte Projektlaufzeit nutzen kann. Zu diesen Stakeholdern zählen insbesondere:
  - Auftraggeber,
  - Architekten,
  - Implementierer,
  - Build-Manager,
  - Integratoren,
  - Tester,
  - Administratoren, die das System installieren, und
  - Wartungspersonal.

Reifegradmodelle SPICE und CMMI

- SPICE (vgl. [MHDZ07]) und CMMI (vgl. [CKS07]) beschreiben die Systemanalyse zweiteilig.
  - SPICE: Begriffe Architecture und Design
  - CMMI: Preliminary Design und Detailed Design
  - Unterschiedliche Begriffe aber identisches Ziel (vgl. Tab. 2–1)
- Die Systemanalyse liefert durch schrittweise Verfeinerung der Architekturkomponenten ein Bild des Gesamtsystems, das immer noch alle ursprünglichen Kundenanforderungen enthält. Diese sind nun aber für die Implementierung überschaubar gruppiert und angeordnet. So wird ein kompliziertes System verständlich; egal ob es nun umfangreich, stark vernetzt oder besonders heterogen ist.
- Kriterien die Partitionierung eines Systems in Komponenten
  - Erfahrung,
  - Vorgaben aus einem entsprechenden Modell oder
  - die Einführung von Abstraktion.
- Falls keine Erfahrungswerte vorliegen und keine Referenzarchitektur aus ähnlichen Projekten existiert, muss zum Mittel der Abstraktion gegriffen werden. (vgl. 2.4)

Reifegradmodelle SPICE und CMMI - Tabelle 2-1

SPICE	CMMI	Ziel
Architecture	Preliminary Design	Beschreibt: • Die Partitionierung des Systems in Komponenten • Die Schnittstellen der Komponenten untereinander • Die verschiedenen Systemzustände • Die Schnittstellen des Systems nach außen Jede Anforderung wird anschließend einer oder mehreren Komponenten zugeordnet. Hier sind auch noch Entwurfsanforderungen zugelassen.
Design	Detailed Design	Beschreibt: • Aus welchen Modulen die Komponenten bestehen • Welche Algorithmen in welchen Modulen verwendet werden • Die konkrete hardwarenahe Umsetzung der Datenstrukturen Alle Entwurfsanforderungen sind in funktionale Anforderungen transformiert und den Modulen zugeordnet.

2.4 Abstraktion erzeugen

- Umgangssprache
  - „Das ist mir zu abstrakt“ meint oft „Ich verstehe es nicht“
- Systems Engineering, Software Engineering
  - Abstraktion ist ein Werkzeug zur Strukturierung komplizierter Systeme

2.4.1 Umgang mit Abstraktion

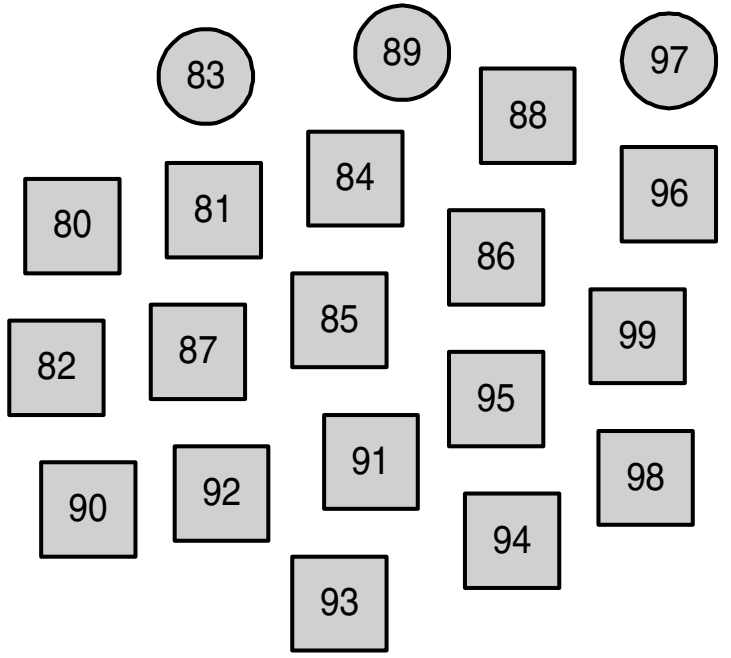
- Abstraktion hat das Ziel, zu konkreten Objekten einen neuen allgemeinen Oberbegriff zu bilden.
- Dazu werden die konkreten Objekte auf wesentliche Gemeinsamkeiten untersucht.
- Diese Gemeinsamkeiten werden dann beim Oberbegriff herausgestellt.
- Unwesentliche Details werden dadurch versteckt.
- Verstecken heißt nicht weggelassen!
- Hinweis für Architekten
  - Das Weglassen von Details führt lediglich zu einer Simplifizierung und nicht zu einer Abstraktion. Durch Simplifizierung entsteht selten eine brauchbare Lösung.

Abstraktion - Ein Beispiel (1)

- Runde Objekte
- Eckige Objekte

Abb. 2–4

Beispiel: Eine Sammlung von Objekten (Zahlen von 80 – 99)



Abstraktion - Ein Beispiel (2)

- Weglassen der runde Objekte
  - Weglassen von Anforderungen!
- Scheinbare Ordnung der eckigen Objekte

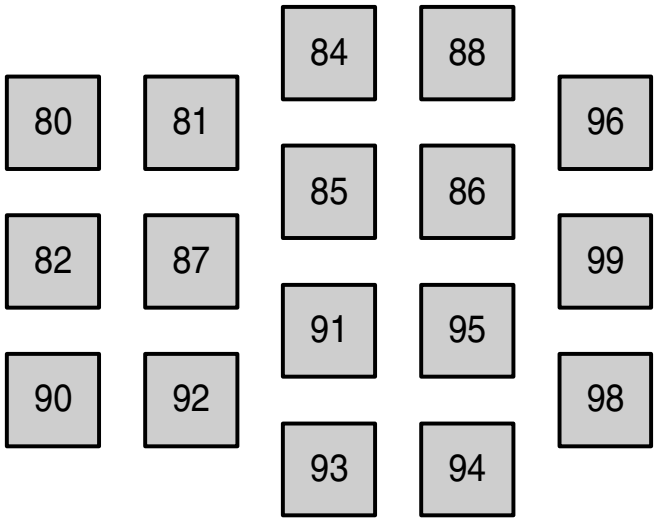
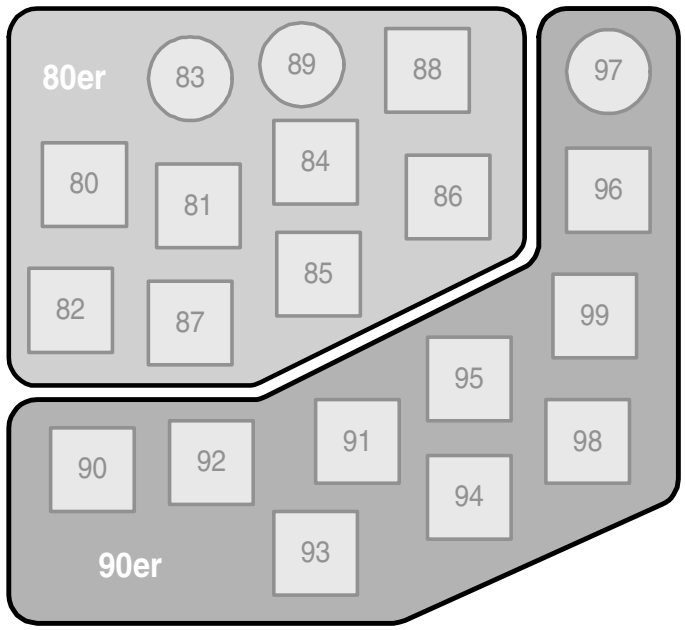


Abb. 2–5

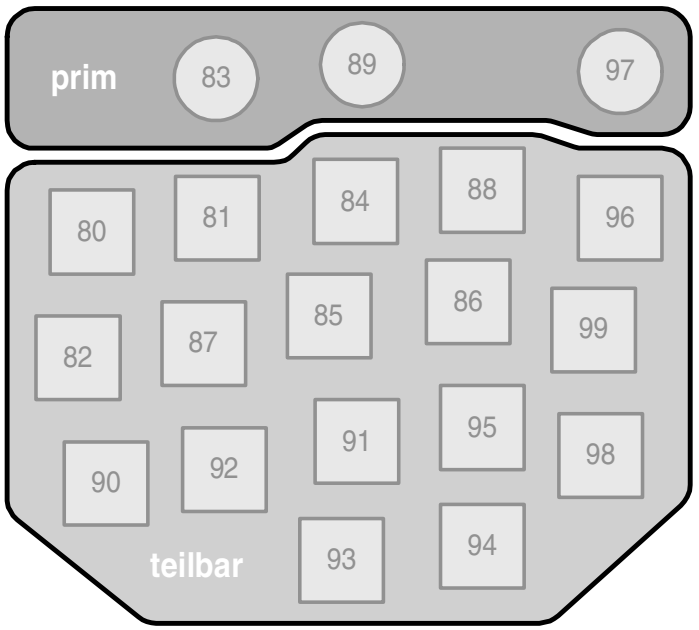
Simplifizierung ist keine Abstraktion

Abstraktion - Ein Beispiel (3)

- Jeweils Abstraktion von 20 auf 2 Elemente (vorher 20 auf 17)



(a) Abstraktion nach Zehnern



(b) Abstraktion nach Teilbarkeit

Abb. 2–6
Zwei korrekte Abstraktionen

2.4.2 Domänenspezifische Datentypen

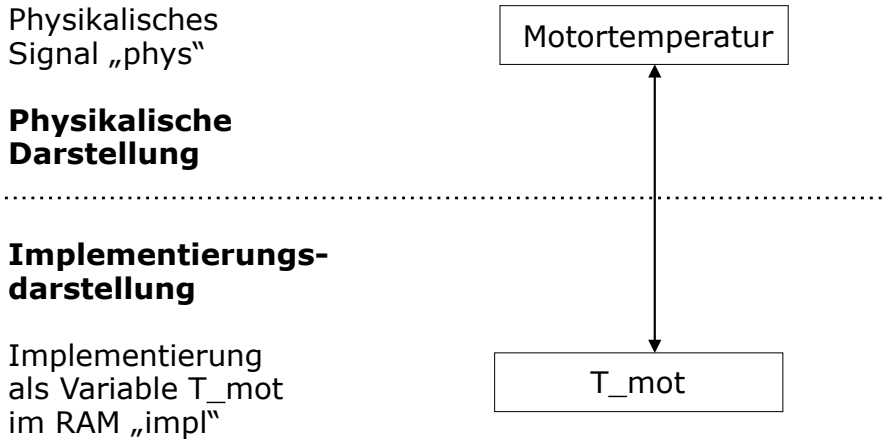
- Wichtiges Abstraktionsmittel bei der Datenmodellierung
- Verzicht auf INT8, INT16, INT32 und ähnliche Hilfstypen. Diese Typplatzhalter spezifizieren keine Datentypen, sondern lediglich Speicherplatz.
- Datentypen abstrahieren vom Typ Zahl und verwenden Begriffe aus dem Kontext der Anwendungsdomäne. Beispiele sind:
  - COLOR
  - VELOCITY
  - SUN_INTENSITY.
  - Klarheit durch neue Information im Code
- Nächster Schritt: Konkrete Abbildung des Wertebereichs auf die Bitrepräsentation
- Typsicherheit im Entwurf schützt z.B. davor, dass Geschwindigkeitswerte an einen Farbwert übergeben werden.

Design und Implementierung

- Design und Implementierung des Datenmodells
  - Unterscheidung zwischen Variablen und durch das Programm nicht veränderbaren Parametern
    - Beispiel für Variable: Motortemperatur
    - Beispiel für Parameter: Schiebedach ja / nein
  - Design-Entscheidungen
    - Prozessorinterne Darstellung
    - Speichersegment für die Ablage
      - RAM = Random[-]access memory, Direktzugriffsspeicher:
        jede Speicherzelle kann über ihre Speicheradresse direkt angesprochen werden
      - ROM = Read-only memory; Festwertspeicher:
        ein Datenspeicher, der nur lesbar ist, im normalen Betrieb aber nicht beschrieben werden kann und nicht flüchtig ist.

Design und Implementierung

- Beispiel Motortemperatur:
  Abbildung der physikalischen Spezifikation auf die Implementierung

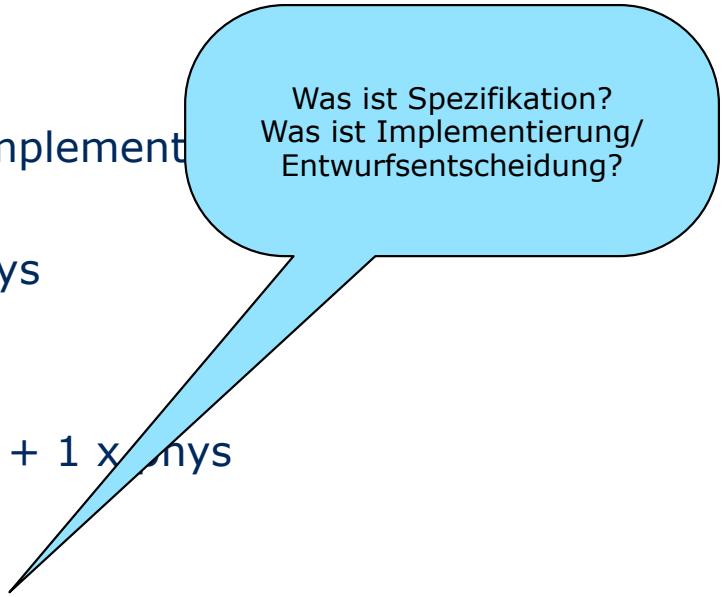


Design und Implementierung

- Beispiel Motortemperatur:
  Abbildung der physikalischen Spezifikation auf die Implementierung
- Physik
  - Bezeichnung im Klartext: Motortemperatur phys
  - Physikalische Einheit: °C
- Umrechnung
  - Umrechnungsformel: impl = f(phys) = 40 + 1 x phys
  - Quantisierung: 1 Bit = 1 °C
  - Offset: 40 °C
  - Minimal-/Maximalwert
    Physik: -40 °C - 215 °C
    Implementierung: 0 - 255
- Imlementierung
  - Bezeichnung im Code: T_mot
  - Wortlänge: 8 Bit
  - Speichsegment: Internes RAM

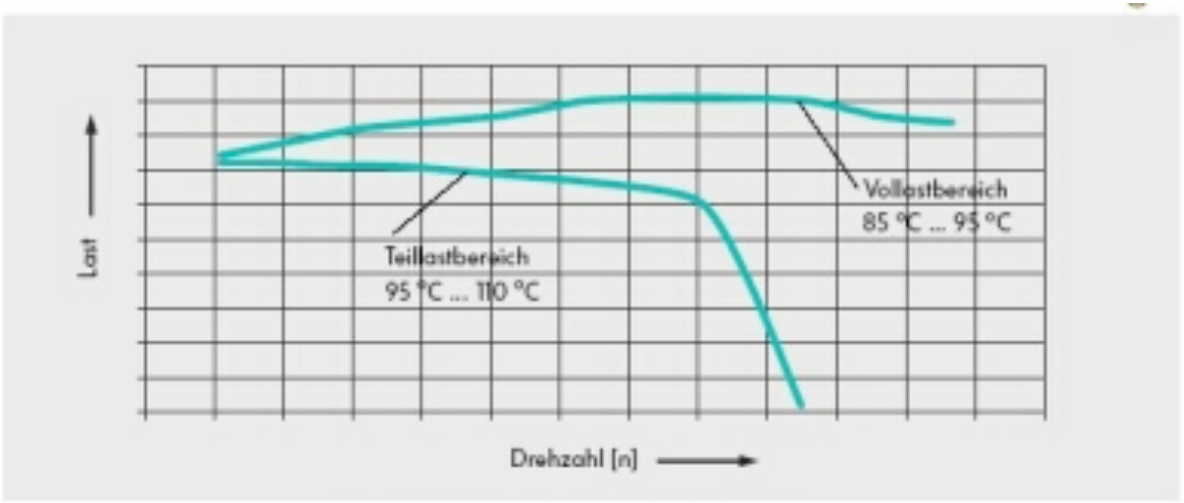
Design und Implementierung

- Beispiel Motortemperatur:
  Abbildung der physikalischen Spezifikation auf die Implement
- Physik
  - Bezeichnung im Klartext: Motortemperatur phys
  - Physikalische Einheit: °C
- Umrechnung
  - Umrechnungsformel: impl = f(phys) = 40 + 1 x phys
  - Quantisierung: 1 Bit = 1 °C
  - Offset: 40 °C
  - Minimal-/Maximalwert
    Physik: -40 °C - 215 °C
    Implementierung: 0 - 255
- Imlementierung
  - Bezeichnung im Code: T_mot
  - Wortlänge: 8 Bit
  - Speichsegment: Internes RAM



Motortemperatur und Kühlmitteltemperatur

- Quelle: http://www.kfztech.de/kfztechnik/motor/kuehlung/wasserkuehlung.htm
- Die im Verbrennungsmotor erzeugten Temperaturen von über 2000°C bedrohen die nur begrenzt hitzebeständigen Motorbauteile. Die überschüssige Wärme muss deshalb schnell und zuverlässig abgeleitet werden. Bei Volllastbetrieb des Motor müssen beispielsweise bis zu 30% der Verbrennungswärme abgeführt werden. Dies gelingt immer noch am besten mit der Flüssigkeitskühlung. Eine gute Kühlung sorgt aber auch für eine bessere Füllung und somit mehr Leistung bei gleichzeitig niedrigerem Kraftstoffverbrauch und weniger Abgasen.



Kühlmittel-Temperaturniveau in Abhängigkeit von der Motorlast bei Kennfeldkühlung

222_013

2.4.3 Architektursichten

- Ein objektiv eindeutiger Sachverhalt lässt mehrere Betrachtungsmöglichkeiten zu.
- Von unterschiedlichen Standpunkten ergeben sich immer unterschiedliche Sichten auf die Dinge.

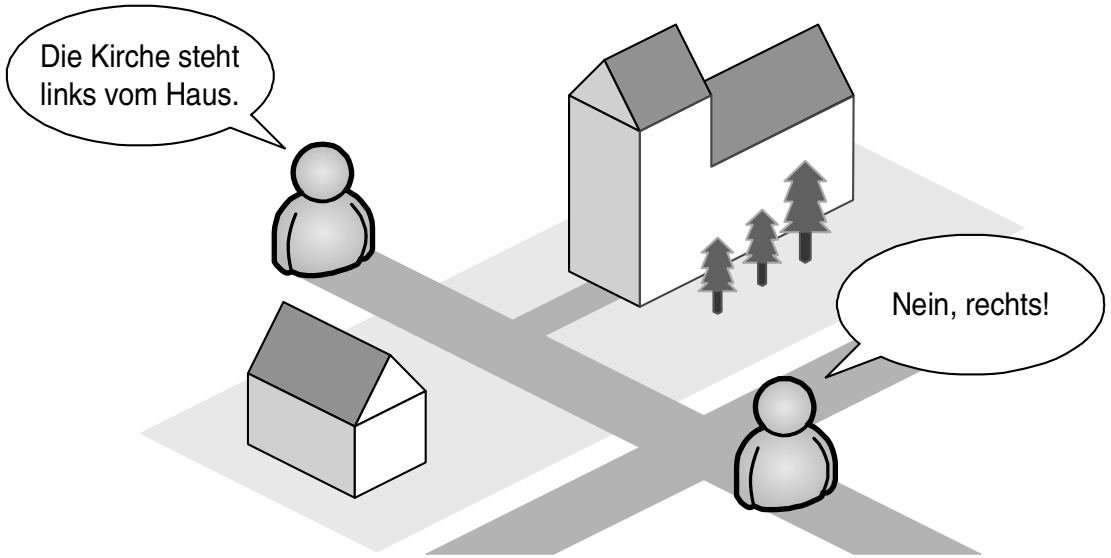


Abb. 2–7

Ein Sachverhalt, zwei Ansichten

Sichten auf die Software-Architektur (1)

- Jeder Stakeholder benötigt eine eigene Sicht auf das System, die nach den für ihn wichtigen Abstraktionskriterien gestaltet sein muss.
- Logische Sicht:
  - Welche Aufgabe haben die Objekte der Anwenderwelt und wie stehen sie untereinander in Beziehung?
  - Welche Daten werden über welche Schnittstellen mit dem Umfeld ausgetauscht?
- Implementierungssicht:
  - Aus welchen Bausteinen, Komponenten, Modulen, Schichten ist das System statisch zusammengesetzt?
  - Aufteilung in Schichten und Code-Units.
  - Die Implementierungssicht betrachtet
    - Wiederverwendung,
    - Hardwareabstraktion,
    - Testbarkeit,
    - Baubarkeit.
  - Ein Ergebnis ist z. B. der Produktstrukturplan (PSP).

Sichten auf die Software-Architektur (2)

- Laufzeitsicht:
  - Wie sind die dynamischen Abläufe zwischen den Teilen?
  - Welche Tasks laufen parallel?
  - Was sind die ausführbaren Einheiten und die ausgetauschten Nachrichten?
  - Gibt es eine synchrone oder asynchrone Interaktion?
  - Welche Ereignisse und Synchronisationspunkte gibt es?
  - Wie startet das System?
  - Wie stoppt es?
- Physikalische Sicht:
  - Wo werden die einzelnen Teile hergestellt?
  - Wo werden sie später technisch installiert?
  - Wie ist die physikalische Verteilung auf die Hardware (engl. deployment)?
  - Über welche Busse und Kanäle läuft die Kommunikation?
  - Die physikalische Sicht betrachtet u. a. Bandbreite und Durchsatz.

Tab. 2–2 Vier Sichten auf eine Softwarearchitektur

Sicht	Fragestellung	Zielgruppe	Elemente	Beziehung
Logische Sicht	Wie ist das System statisch zusammen-gesetzt? Wie sieht das Datenmodell aus? Welche Funktionen, Zustände und Zustandsübergänge gibt es? Wie sieht die Funktionalität nach außen aus?	Anwender, Designer Ziel: Funktionale Anforderungen umsetzen	Datenstrukturen, Funktionen, Klassen, Subsysteme, Komponenten, Layer, Zustandsautomaten	uses, contains
Implementie-rungssicht	Wie wird das System gebaut? Wie lauten Verzeich-nis- und Dateinamen in der Entwicklungs-umgebung? Wie sieht der Produktstrukturplan aus? Zuordnung: Funktion ⇒ Code-Unit	Entwickler, Build-Manager, Integratoren, Projektleiter Ziel: Entwicklung verteilen, Wiederver-wendung, Produkt-linien, Hardwareunab-hängigkeit, Testbarkeit, Baubarkeit	Code-Units, Libraries, Module	includes, uses, depends-on

Tab. 2–2 Vier Sichten auf eine Softwarearchitektur

Sicht	Fragestellung	Zielgruppe	Elemente	Beziehung
Laufzeitsicht	Wie verhält sich das System? In welchem Task läuft welche Funktion? Was läuft parallel? Synchronisation. Wie startet/stoppt das System. Zuordnung: Funktion ⇒ Task	Designer, Integratoren Ziel: Verfügbarkeit, Antwortzeiten und Lastverhalten optimieren	Tasks, Prozesse	Message, Broadcast, Event, Call, Wait for
Physikalische Sicht	Was wird auf welchem Hardwareknoten installiert? Zuordnung: Tasks ⇒ ECU Komponenten ⇒ ECU	Systemdesigner Ziel: Performance, Skalierbarkeit, Verfügbarkeit	Binaries, Images	Bussysteme, CAN, LIN, Flexray, Draht, Funk

2.5 Ein System partitionieren

- Ziel der Partitionierung
  - Wenige Teile mit möglichst wenigen Beziehungen untereinander
  - Neue abstrakte Betrachtungsebene
    - Jedes Teil orientiert sich dazu an bestimmten Merkmalen der enthaltenen Elemente
- Weitere Prinzipien bei der Partitionierung
  - 2.5.1 Separation of Concerns
  - 2.5.2 Lokale und globale Kompliziertheit ausbalancieren
  - 2.5.3 Schichtenarchitekturen
  - 2.5.4 Komponentenarchitekturen
  - 2.5.5 Das KISS-Prinzip

2.5.1 Separation of Concerns

- Separation of Concerns
  - Zerlegung eines Systems dergestalt dass verschiedene Aspekte des Systems jeweils in sauber voneinander getrennten Komponenten des Systems gelöst werden.
  - Konzentration auf die wesentlichen Eigenschaften der Komponenten.
  - E. W. Dijkstra Mitte der 70er-Jahre
  - Wesentliches Kernelement der funktionalen Dekomposition.
  - In der objektorientierten Analyse zum Entwurf von Klassenstrukturen verwendet.
  - Prinzip der Abstraktion:
    - Gruppiere Dinge mit gleichen Eigenschaften und gib dieser Gruppe anschließend einen sinnvollen Namen.
- Ergebnis:
  - Komponenten mit loser Kopplung und starkem Zusammenhalt.

Lose Kopplung (low coupling)

- Zwei Komponenten sind lose gekoppelt, wenn ihre gegenseitigen Abhängigkeiten minimal sind.
  - Die Komponenten haben kleine und stabile Schnittstellen.
  - Es gibt keine gemeinsam genutzten globalen Variablen.
  - Beide Komponenten wissen nichts über die interne Implementierung der jeweils anderen.
  - Es gibt keine zirkularen Abhängigkeiten.
- Vorteile
  - Änderungen an einer Komponente führen nur zu minimalen Auswirkungen bei anderen Komponenten.
    - Bessere Wartbarkeit
    - Grundstein für Wiederverwendbarkeit.
  - Einfacheres Verständnis
    - Weniger Abhängigkeiten vom Umfeld, deswegen weniger Kenntnisse des Umfelds notwendig.

Lose Kopplung (low coupling)

- Vorteile
  - Verbesserung der Testbarkeit
    - Komponenten können isoliert getestet werden
    - Komponenten können früher getestet werden
    - Höhere Stabilität bei der Integration
    - Integration kann sich auf das korrekte Zusammenspiel der Komponenten konzentrieren

Starker Zusammenhalt (high cohesion)

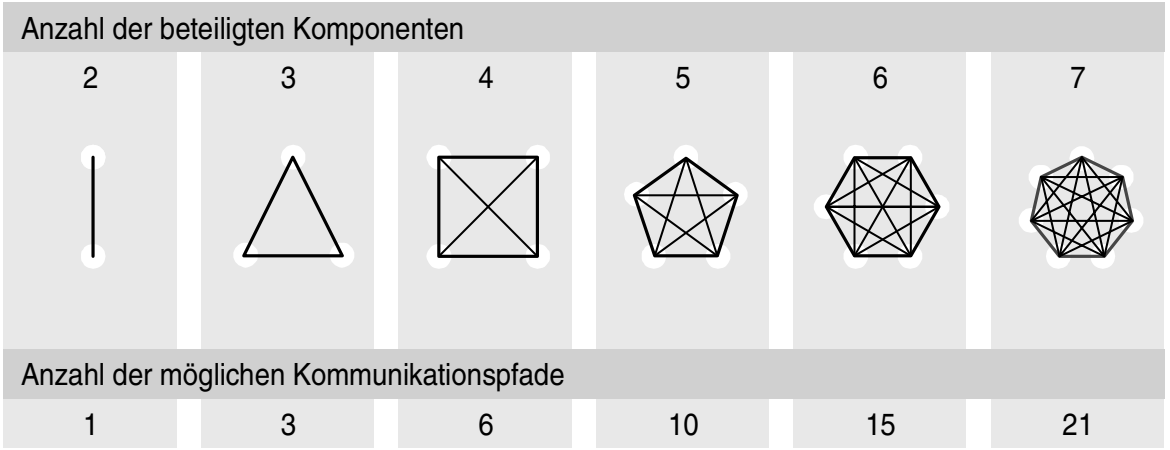
- Eine Komponente hat starken Zusammenhalt, wenn alle ihre Funktionen einen gemeinsamen Aspekt teilen. Die Funktionen
  - benötigen als Input den Output der Vorgängerfunktion,
  - werden immer in der gleichen Reihenfolge aufgerufen,
  - operieren auf denselben Daten oder
  - führen ähnliche Aktivitäten aus.
- Starker Zusammenhalt führt zu Kapselung der Daten.
- Wenn alle Änderungen an einer Datenstruktur grundsätzlich nur über Funktionen in einem Modul laufen, wird es möglich, die Integrität dieser Datenstruktur bei jeder Operation optimal zu gewährleisten.
- Je einheitlicher und fokussierter die Verantwortlichkeit eines Moduls ist, desto einfacher lässt es sich auch dokumentieren.
  - … und verstehen!

Starker Zusammenhalt und das DRY-Prinzip

- DRY steht für „don’t repeat yourself“
  - Jede Information sollte grundsätzlich immer nur an genau einer Stelle im System verantwortlich gehalten werden.
  - Grund
    - Wird ein bestimmtes Wissen mehrfach im System gehalten, so müssen bei einer Änderung auch alle diese Stellen gefunden und geändert werden.
      - Gewährleistung der Konsistenz schwieriger.
      - Wartungsaufwand und Fehleranfälligkeit steigen
    - Identisch wiederholte Codesequenzen in Unterprogramme wandeln

2.5.2 Lokale und globale Kompliziertheit

Abb. 2–8
Aufwand für die Kommunikation »jeder mit jedem«



- Nach der Partitionierung
  - Einige Kommunikationspfade verschwinden in den Komponenten (Lokale Kommunikation)
  - Es bleibt die Kommunikation zwischen den Komponenten (Globale Kommunikation)

2.5.3 Schichtenarchitekturen

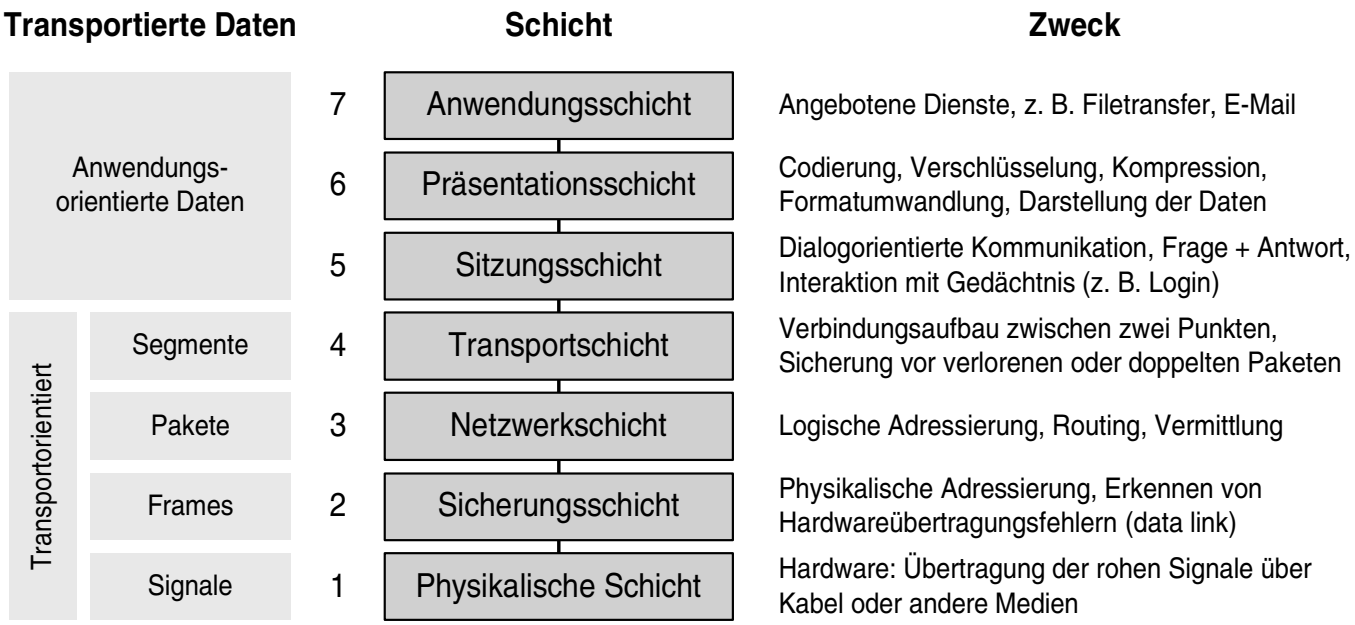


Abb. 2–9

OSI-Schichtenarchitektur

- Abstraktionsprinzip des OSI-Modells: Trennung von fachlichen und technischen Lösungselementen eines netzwerkbasierten Systems
  - Eine beliebige fachliche Anwendung kann mit der Netzwerkinfrastruktur kombinieret werden
  - Die technische Realisierung der Kommunikation kann ausgetauscht werden, ohne bestehende Anwendungen zu beeinträchtigen.

Eigenschaften von Schichtenarchitekturen

- Die einzelnen Schichten sind lose miteinander gekoppelt.
- Eine Schicht kennt ihre direkt darunter liegende Schicht. Funktionsaufrufe nach unten erfolgen durch direkten Aufruf.
- Eine Schicht kennt ihre darüberliegende Schicht nicht. Funktionsaufrufe nach oben erfolgen durch registrierte Callbacks.

- Schichtenmodelle sind die erste Wahl, wenn es darum geht, Hardwareunabhängigkeit in ein System zu integrieren. Daher folgen auch Betriebssysteme einer Schichten-architektur. Auch sie trennen fachlichen Code (die Anwendungen) von technischem Code (z. B. dem Filesystem).

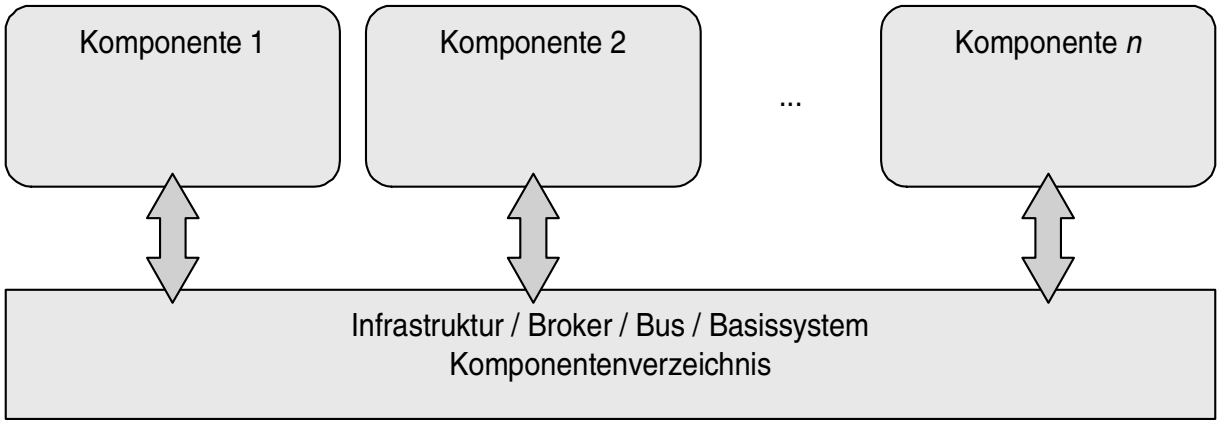
2.5.4 Komponentenarchitekturen

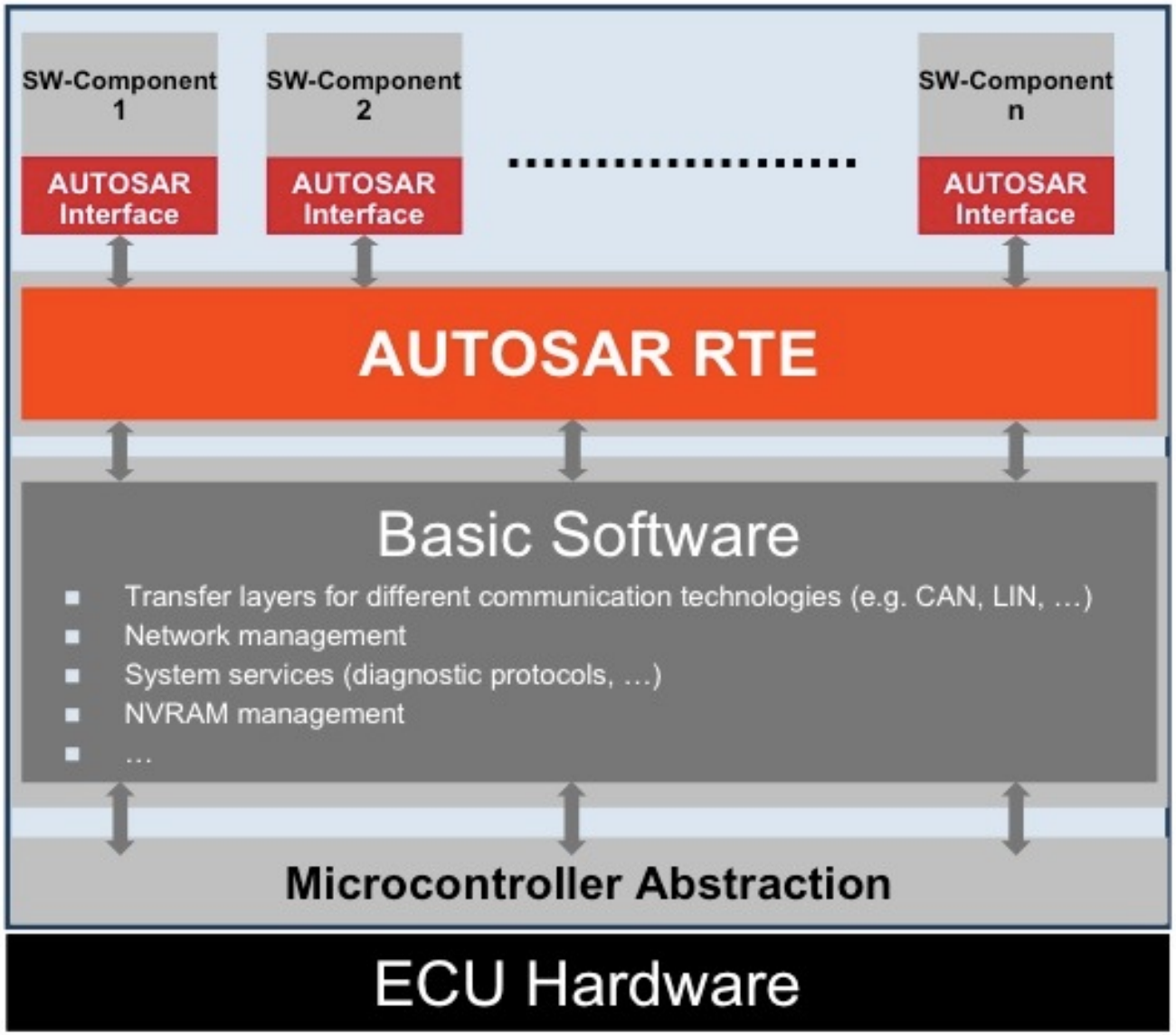
- Die Elemente des Systems sind über eine einheitliche busähnliche Kommunikationsstruktur verbunden (vgl. Abb. 2–10).
- Die Verwaltung der Komponenten erfolgt durch einen zentralen Manager.
- Kommunikation geschieht über zentrale Mechanismen.
- Die einzelnen Komponenten kennen sich untereinander nicht.
- Jede Komponente definiert eine öffentliche Schnittstelle.

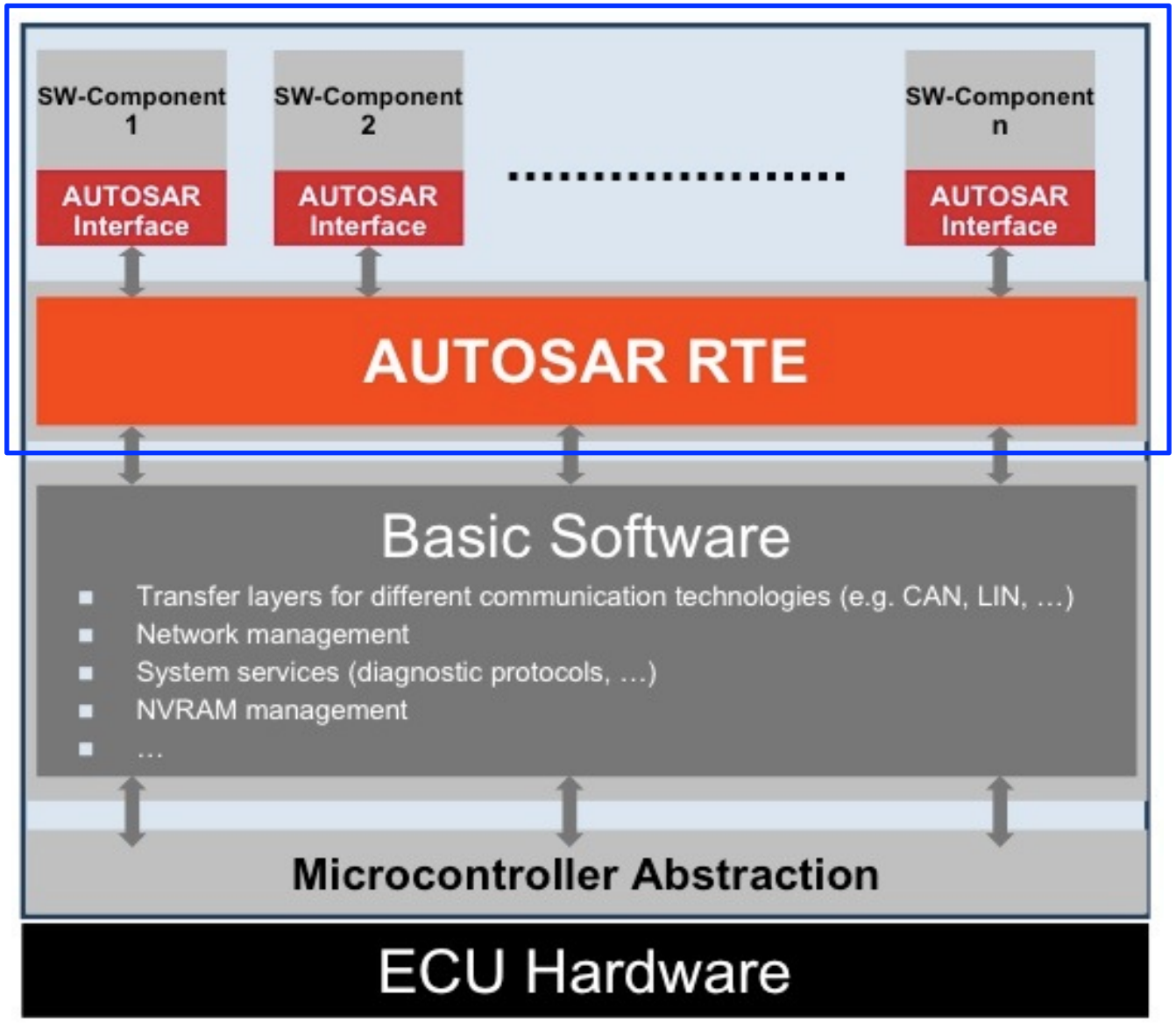
- Variante
  - Serviceorientierte Architekturen (SOA)
  - OSGi-Plattform (Open Services Gateway initiative, www.osgi.org)
    - Komponenten liefern/nutzen Services.
    - Verwaltungsservices bieten die zugehörige Infrastruktur.

Abb. 2–10

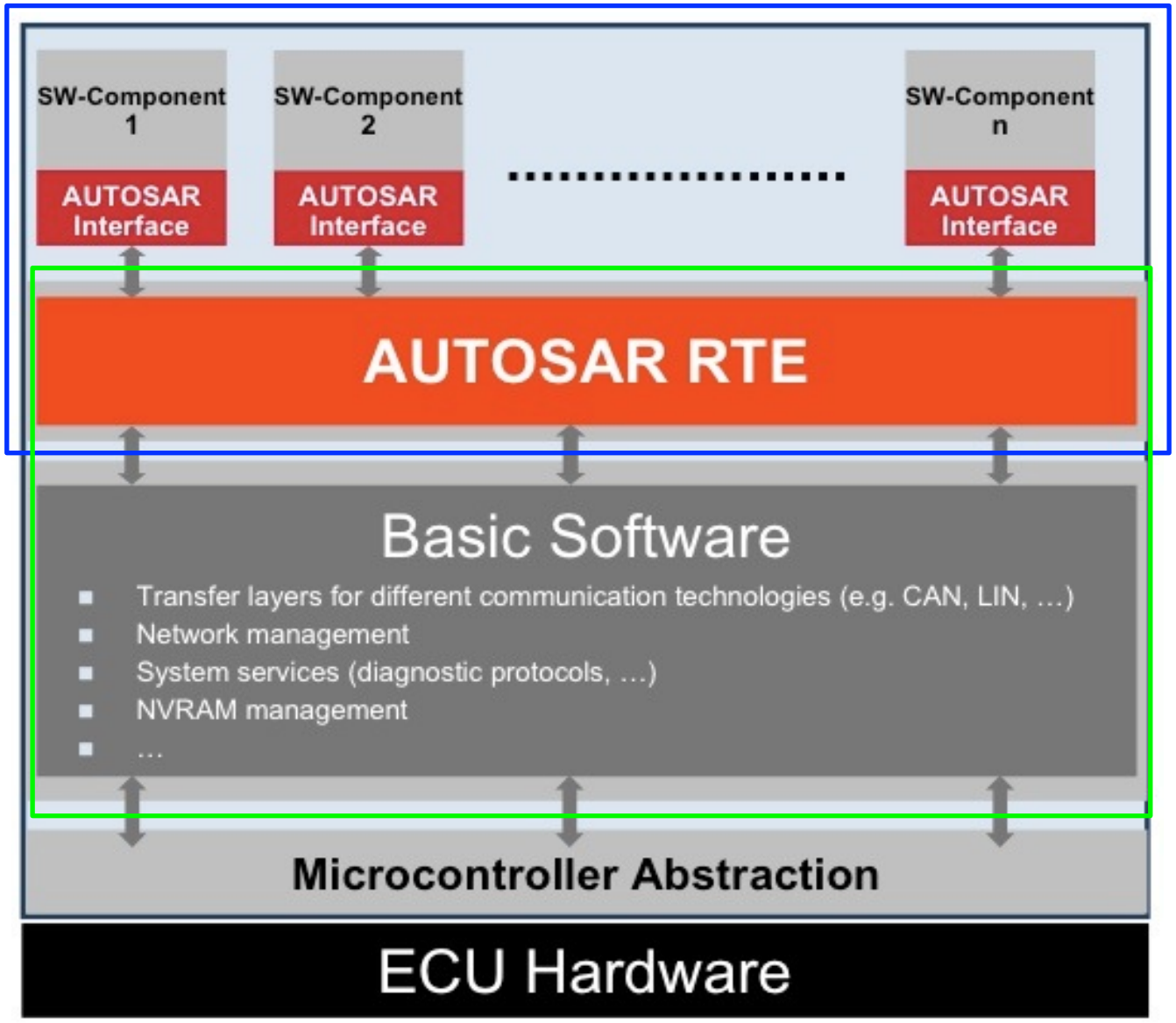
Komponentenarchitektur




SW-Component 1
SW-Component 2
SW-Component n
AUTOSAR Interface
AUTOSAR Interface
AUTOSAR Interface
AUTOSAR RTE
Basic Software
Transfer layers for different communication technologies (e.g. CAN, LIN, …)
Network management
System services (diagnostic protocols, …)
NVRAM management
…
Microcontroller Abstraction
ECU Hardware



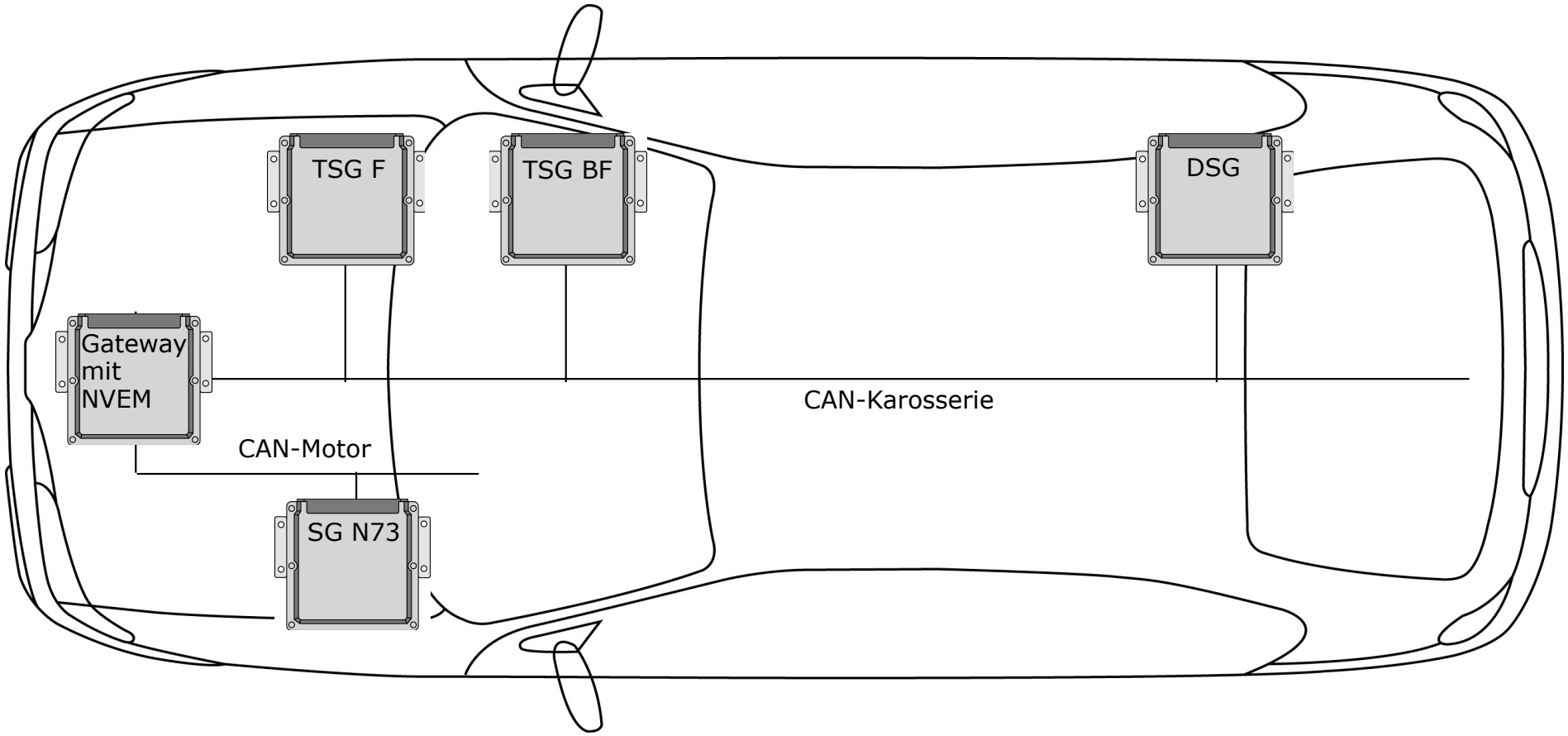
ASW Komponentenarchitektur



ASW Komponentenarchitektur

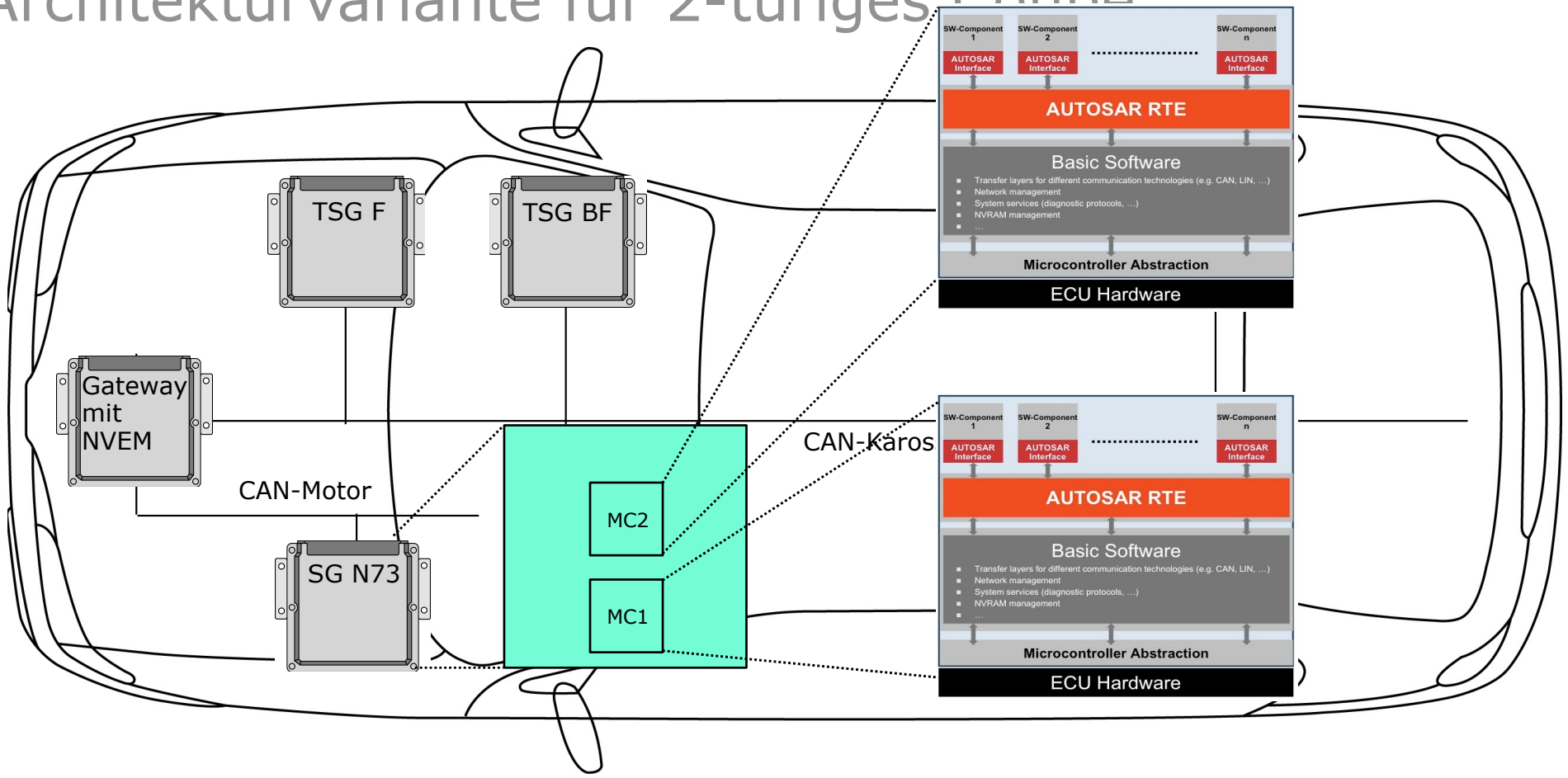
BSW Schichtenarchitektur

Innenraumbeleuchtung
Architekturvariante für 2-türiges Coupé



- Quellen:
  T. Trautmann: Grundlagen der Fahrzeugmechatronik, Vieweg+Teubner Verlag, 2009. (Struktur)
  J. Schäuffele,Th. Zurawka: Automotive SW Engineering, Vieweg+Teubner, 4. Auflage, 2010. (Grafiken)

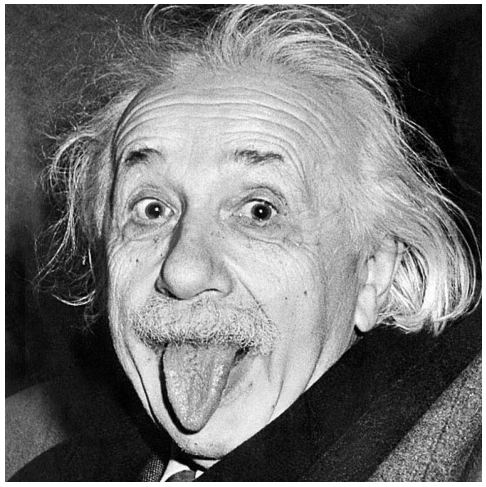
Innenraumbeleuchtung
Architekturvariante für 2-türiges Coupé

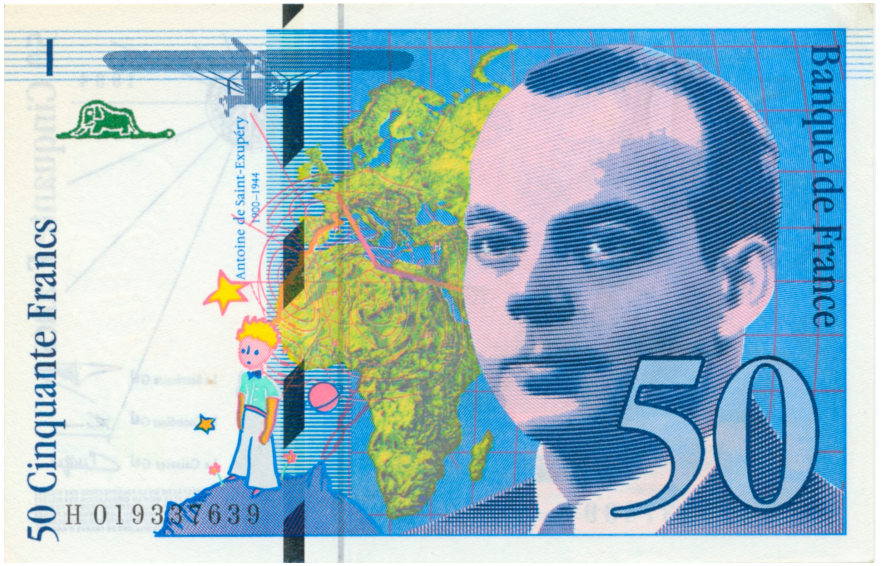


- Quellen:
  T. Trautmann: Grundlagen der Fahrzeugmechatronik, Vieweg+Teubner Verlag, 2009. (Struktur)
  J. Schäuffele,Th. Zurawka: Automotive SW Engineering, Vieweg+Teubner, 4. Auflage, 2010. (Grafiken)

2.5.5 Das KISS-Prinzip

- KISS-Prinzip fordert:
  „Keep it simple and stupid.“
- Viele nett zu lesende Bonmots zu Over Engineering etc
- „So einfach wie möglich, aber nicht einfacher“ (Elbert Einstein)
- „Perfektion entsteht offensichtlich nicht dann, wenn man nichts mehr hinzuzufügen hat, sondern wenn man nichts mehr wegnehmen kann.“
  (*A. de Saint-Exupéry*)





Teil I Grundlagen

3 Motive für den Einsatz von AUTOSAR

Motivation auf Management-Niveau

Teil I Grundlagen

2 Softwarearchitektur in der Fahrzeugentwicklung

3 Motive für den Einsatz von AUTOSAR

4 AUTOSAR im Detail

4 AUTOSAR im Detail

- Inhalt des AUTOSAR-Standards im Überblick
- (Potentielle) Vorteile des AUTOSAR-Standards
- Vergleich zu anderen Standards
- Grundlegende AUTOSAR-Ideen und -Begriffe
- Einstieg in die AUTOSAR-Spezifikation

- AUTOSAR kurz und schmerzlos

4 AUTOSAR im Detail

4.1 Ziele

- Verbesserung der Softwareentwicklung im Automotivebereich
  - Kostensenkung für Automobilhersteller (OEM) und Zulieferer
  - Erhöhung der Software-Qualität
- Heute ([FKFS07])
  - Elektrik und Elektronik: 30% der Wertschöpfung eines Mittelklassefahrzeugs
  - Treiber für etwa 90% aller Innovationen im Automobil
- Künftig
  - Noch komplexere Elektronik- und Softwaresysteme
  - Weiterhin sehr starker Anstieg der Funktionen, die durch Elektronik und Software realisiert werden

- siehe Abschnitt „Motivation AUTOSAR“

Innovationstreiber (1)

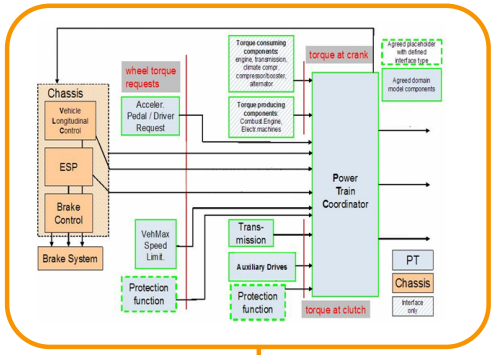
- Komfortfunktionen
  - Infotainment-/Navigationssysteme
  - Personalisierung
  - Einparkautomatik
- Fahrersicherheit / Fahrerassistenzsysteme
  - lane assistant/lane departure warning (Spurhalteassistent)
  - automatic cruise control (automatische Geschwindigkeitsregelung) oder
  - ganz traditionell ABS, ESP etc.
  - Zukünftig Nutzung Fahrzeug-zu-Fahrzeug-Kommunikation (Car2Car, allgemeiner C2X) zur Verhinderung von Kollisionen auf Kreuzungen oder beim Einfädeln zu verhindern.

- Welche Domänen?
- Was wird von AUTOSAR unterstützt?

Application Interfaces

AUTOSAR Application Interfaces Compositions under Consideration

- **Body Domain**
  - Central Locking
  - Interior Light
  - Mirror Adjustment
  - Mirror Tinting
  - Seat Adjustment
  - Wiper/Washer
  - Anti Theft Warning System
  - Horn Control
  - Exterior Lights
  - Defrost Control
  - Seat climatization
  - Cabin climatization
  - Steering wheel climatization
  - Window Control
  - Sunroof/Convertible control
  - Steering column adjustment
  - Roller blind control



- **Chassis Control Domain**
  - Vehicle Longitudinal Control
  - Electronic Stability Program
  - Electronic Parking Brake
  - Adaptive Cruise Control
  - Roll Stability Control
  - Steering System
  - Suspension System
  - Stand Still Manager
  - High Level Steering
    - Vehicle Stability Steering
    - Driver Assistance Steering
  - All Wheel Drive/ Differential Lock

- **Powertrain Domain**
  - Powertrain Coordinator
  - Transmission System
  - Combustion Engine
    - Engine torque and mode management
    - Engine Speed And Position
    - Combustion Engine Misc.
  - Electric Machine
  - Vehicle Motion Powertrain
    - Driver Request
    - Accelerator Pedal Position
    - Safety Vehicle Speed Limitation

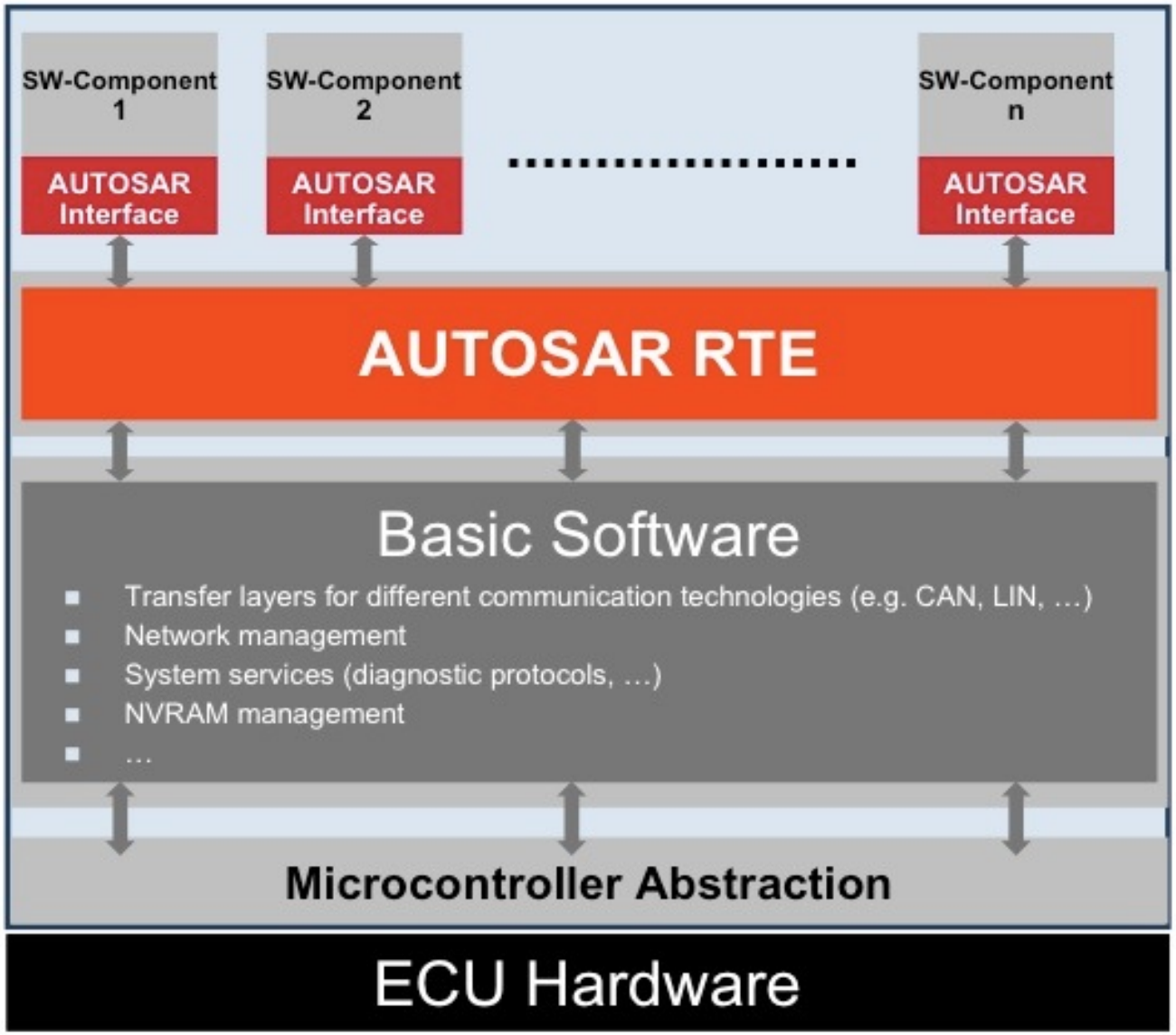
Innovationstreiber (2)

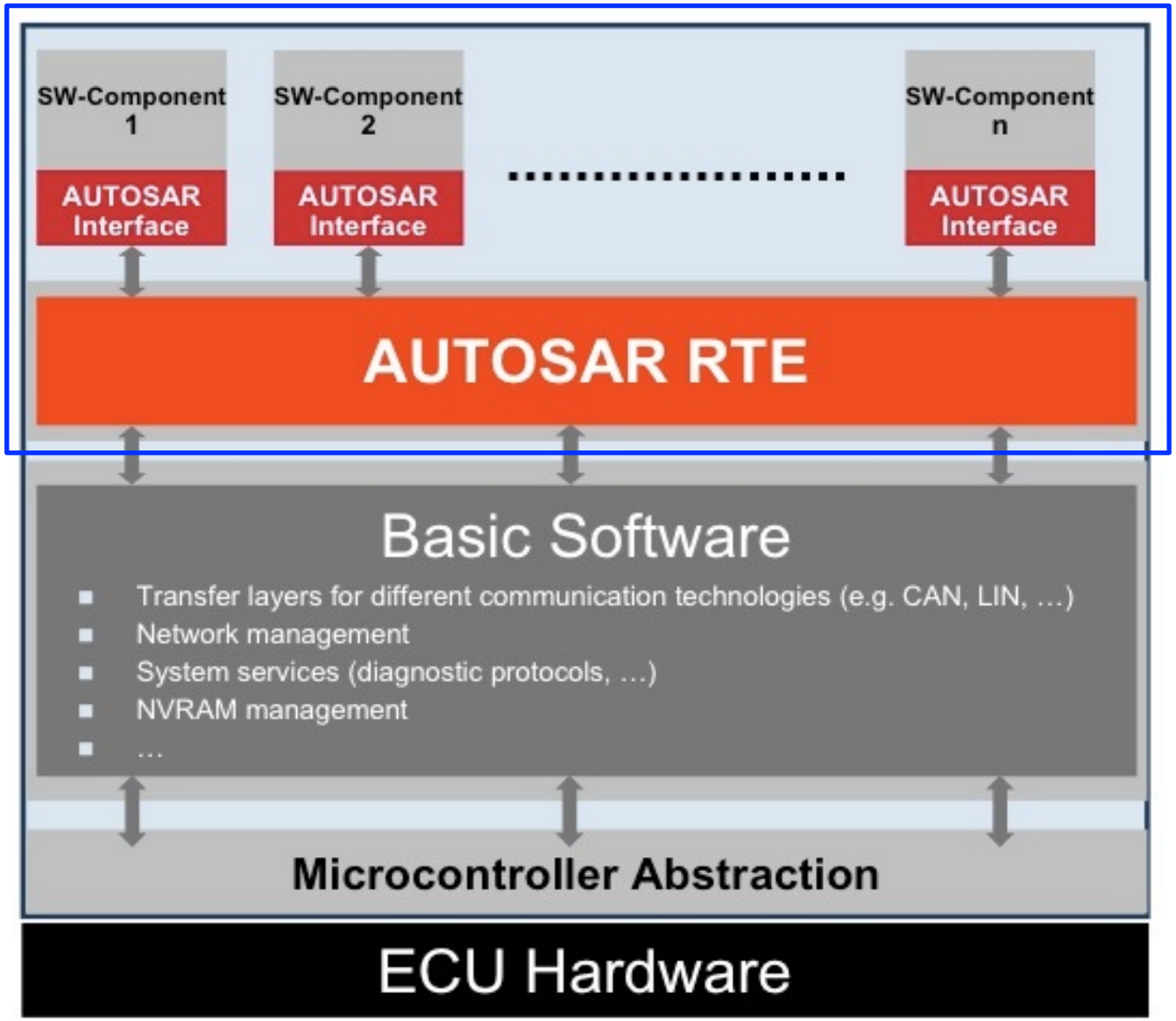
- Reduzierung von Kraftstoffverbrauch und CO2-Ausstoß
  - Verbrauchsreduzierung durch optimierte elektronische Motorsteuerungen
  - Start/Stop-Automatik
    - Im Fahrzeugstillstand wird der Motor gestoppt und nach Ende der Stillstandsphase automatisch wieder gestartet.
  - Alternative Kraftstoffe (Gas, Biodiesel, Ethanol, …)
  - Alternative Antriebe (Elektro, Wasserstoff, Hybridantrieb, …)
  - Elektrifizierung von Nebenaggregaten
- Ausgeklügelte Steuerung Elektrik/Elektronik/Software

- siehe Abschnitt „Motivation AUTOSAR“

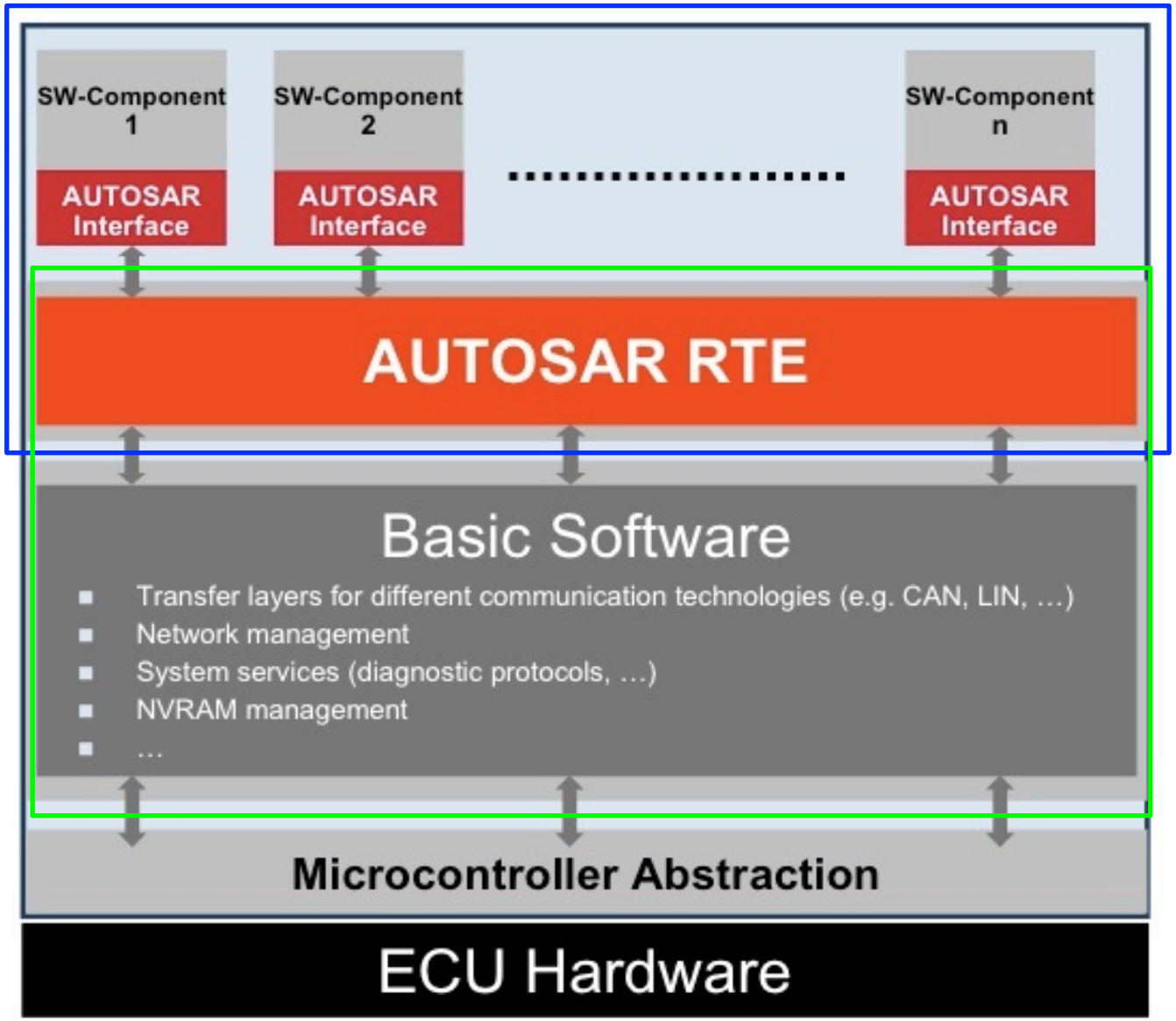
4.2 Schwerpunkte (1)

- Architektur (Architecture)
  - Ziel: Unabhängigkeit der Anwendungssoftware von der benutzten Hardware
  - Schichtenmodell für Software
    - Anwendungssoftware (Application Software, ASW)
    - Run-Time Environment (RTE)
    - Basissoftware (Basic Software, BSW)


SW-Component 1
SW-Component 2
SW-Component n
AUTOSAR Interface
AUTOSAR Interface
AUTOSAR Interface
AUTOSAR RTE
Basic Software
Transfer layers for different communication technologies (e.g. CAN, LIN, …)
Network management
System services (diagnostic protocols, …)
NVRAM management
…
Microcontroller Abstraction
ECU Hardware



ASW Komponentenarchitektur



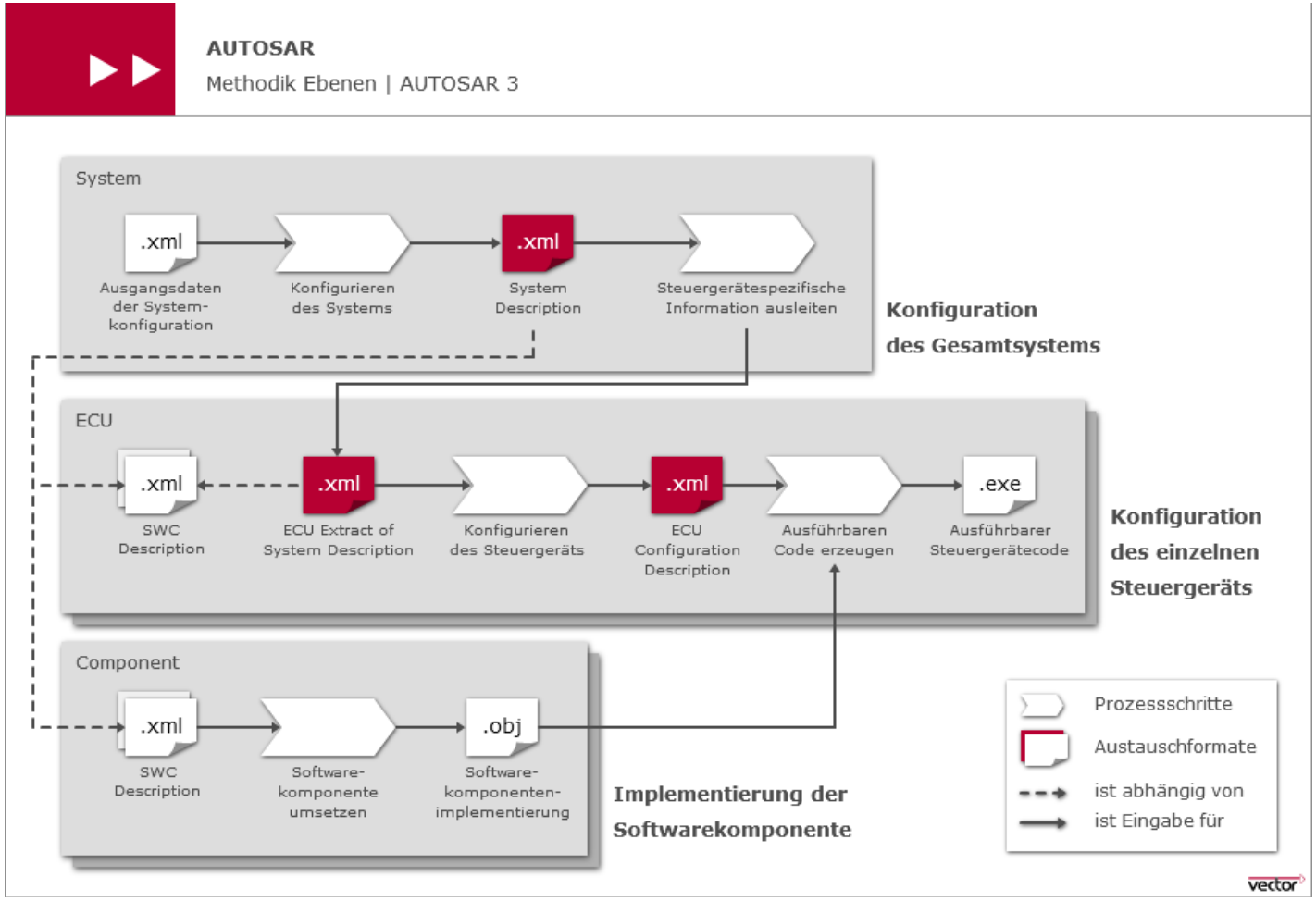
ASW Komponentenarchitektur

BSW Schichtenarchitektur

4.2 Schwerpunkte (2)

- Methodik (Methodology)
  - Einheitliche Beschreibungsformate (in XML)
    - Verteilung der Software über ECU-Grenzen
    - Konfiguration der Basissoftware der einzelnen ECUs
    - Austausch und Integration von Software zwischen den beteiligten Partnern (Hersteller/OEM und Zulieferer)

Methodik


AUTOSAR
Methodik Ebenen | AUTOSAR 3
System
.xml
Ausgangsdaten der System-konfiguration
Konfigurieren des Systems
.xml
System Description
Steuergerätespezifische Information ausleiten
Konfiguration des Gesamtsystems
ECU
.xml
SWC Description
.xml
ECU Extract of System Description
Konfigurieren des Steuergeräts
.xml
ECU Configuration Description
Ausführbaren Code erzeugen
.exe
Ausführbarer Steuergerätecode
Konfiguration des einzelnen Steuergeräts
Component
.xml
SWC Description
Software-komponente umsetzen
.obj
Software-komponenten-implementierung
Implementierung der Softwarekomponente
Prozessschritte
Austauschformate
ist abhängig von
ist Eingabe für
vector

4.2 Schwerpunkte (3)

- Anwendungsschnittstellen (Application Interfaces)
  - Festlegung der Schnittstellen typischer Automotive Software Anwendungen
  - Erleichterung der Integration
  - Keine Festlegung der funktionalen Eigenschaften
    - Wettbewerb gewünscht

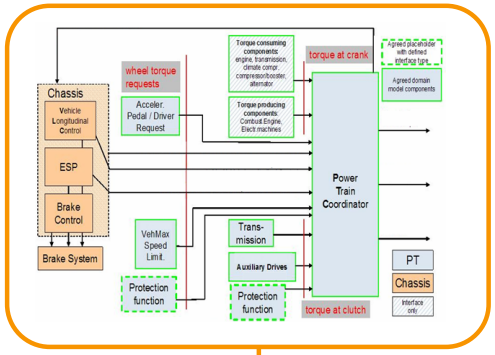
- AUTOSAR jeweils nur den Rahmen / die Schnittstellen vor und lässt genügend Spielraum für Innovationen in wettbewerbsrelevanten Bereichen.

- Der AUTOSAR-Leitspruch:
  „Cooperate on standards, compete on implementation.“

Application Interfaces

AUTOSAR Application Interfaces Compositions under Consideration

- **Body Domain**
  - Central Locking
  - Interior Light
  - Mirror Adjustment
  - Mirror Tinting
  - Seat Adjustment
  - Wiper/Washer
  - Anti Theft Warning System
  - Horn Control
  - Exterior Lights
  - Defrost Control
  - Seat climatization
  - Cabin climatization
  - Steering wheel climatization
  - Window Control
  - Sunroof/Convertible control
  - Steering column adjustment
  - Roller blind control



- **Chassis Control Domain**
  - Vehicle Longitudinal Control
  - Electronic Stability Program
  - Electronic Parking Brake
  - Adaptive Cruise Control
  - Roll Stability Control
  - Steering System
  - Suspension System
  - Stand Still Manager
  - High Level Steering
    - Vehicle Stability Steering
    - Driver Assistance Steering
  - All Wheel Drive/ Differential Lock

- **Powertrain Domain**
  - Powertrain Coordinator
  - Transmission System
  - Combustion Engine
    - Engine torque and mode management
    - Engine Speed And Position
    - Combustion Engine Misc.
  - Electric Machine
  - Vehicle Motion Powertrain
    - Driver Request
    - Accelerator Pedal Position
    - Safety Vehicle Speed Limitation

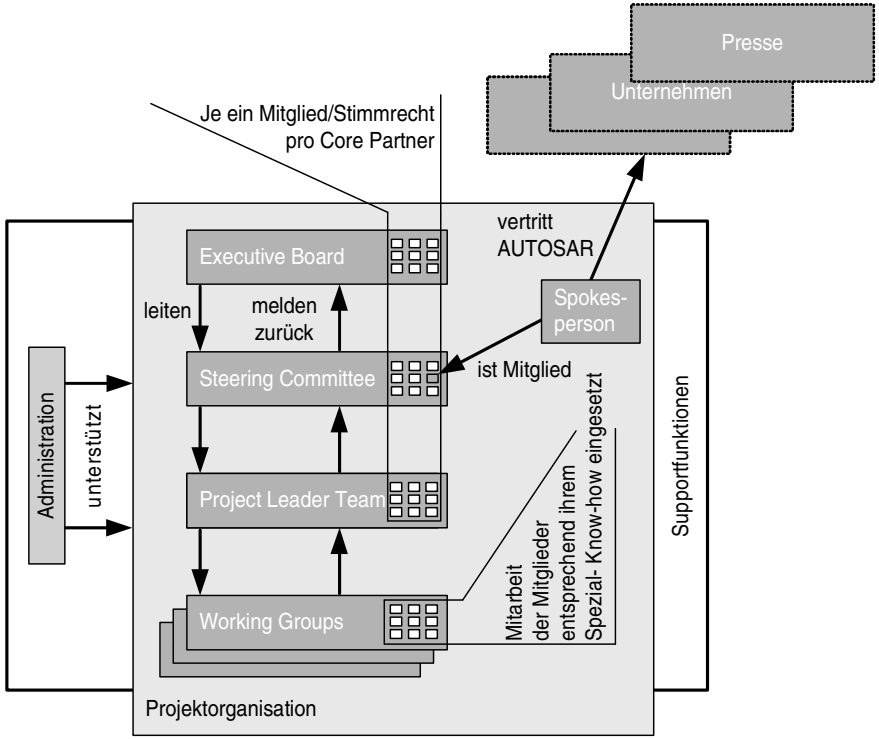
4.3 Organisation

- AUTOSAR
  - Internationale Entwicklungspartnerschaft
  - Finanzierung aus Mitgliedsbeiträgen
  - Ergebnisse werden den Mitgliedern zur kommerziellen Nutzung bereitstellt

4.3.1 Struktur (1)

- ExecutiveBoard (EB)
- SteeringCommittee (SC)
- ProjectLeaderTeam (PLTeam)
- WorkingGroups (WG)

Abb. 4–1
AUTOSAR-Organisationsstruktur



4.3.1 Struktur (2)

- ExecutiveBoard (EB)
  - Vorgabe von Strategie und Zielen
- SteeringCommittee (SC)
  - Organisatorische Ebene
    - Aufnahme neuer Mitglieder
    - Öffentlichkeitsarbeit
    - Rechtsfragen
- ProjectLeaderTeam (PLTeam)
  - Technische Ebene
    - Anleitung und Überwachung der Working Groups
- WorkingGroups (WG)
  - Erarbeitung der Spezifikation

- Jeder Core Partner hat einen Sitz / einfaches Stimmrecht in Executive Board, Steering Committee und Project Leader Team

4.3.1 Struktur (3)

- Ergänzende und unterstützende Rollen
  - Administration
    - Technische Unterstützung, z.B. Pflege der Internetseiten
    - Spokesperson
      - Eigenverantwortliche Pressearbeit
      - Unterzeichnung von Mitgliedsverträgen (im Auftrag der Entwicklungs-partnerschaft)
      - Die Spokesperson ist Steering-Committee-Mitglied und führt diese Tätigkeit für ein Jahr aus

AUTOSAR

Media Release

October 17, 2014

AUTOSAR Names Thomas Rüping as New Spokesperson

On October 1, 2014, the AUTOSAR (AUTomotive Open System ARchitecture) development partnership appointed Thomas Rüping as its new spokesperson. As part of the general nine-monthly change of the AUTOSAR spokesperson, Rüping, project director at the Cross Divisional Group Software, Methods and Tools at Robert Bosch GmbH and chairman of the board of COMASSO e.V., takes over from Rick Flores, Global Lead of Model-based Electrical System and Software Engineering at General Motors. The new deputy spokesperson will be Tony Jaux, Senior Manager Powertrain and Chassis Electronic Core Modules from PSA, who will succeed Demetrio Aiello, Head of the AUTOSAR Center at Continental Engineering Services.

4.3.2 Mitglieder (1)

- Mitglieder
  - Finanzierung der Organisation und Mitarbeit in den Arbeitsgruppen
- Core Partner
  - Leitung und aktive Beteiligung in unterschiedlichen Arbeitspaketen
  - Lieferant und Empfänger von technischen Informationen und Ideen
  - Kommerzielle Nutzung der Ergebnisse
- Premium Member
  - Aktive Beteiligung in unterschiedlichen Arbeitspaketen
  - Lieferant und Empfänger von technischen Informationen und Ideen
  - Kommerzielle Nutzung der Ergebnisse
- Associate Member
  - Kommerzielle Nutzung der Ergebnisse
- Development Member
  - Technisches Spezial-Know-how
  - Keine finanziellen Beiträge, (bezahlte) Mitarbeit in den Arbeitsgruppen
  - Kommerzielle Nutzung der Ergebnisse

4.3.2 Mitglieder (2)

- Juni 2008:
  9 Core Partner, 53 Premium Member, 76 Associate Member, 6 Development Member
- Aktueller Stand: http://www.autosar.org/partners/current-partners/
- Typische Mitglieder
  - CorePartner
    OEMs und Tier-1-Supplier (liefern direkt an die OEMs). Dies waren im Juni 2008: BMW, Bosch, Continental, Daimler, Ford, Opel, PSA, Toyota und Volkswagen.
  - PremiumMember
    Tier-1-Supplier und Werkzeughersteller.
  - AssociateMember:
    Sonstige Zulieferer oder allgemein Unternehmen, die eine kommerzielle Nutzung des Standards anstreben.

4.4 Geltungsbereich

- Internationaler Standard
  - Schwerpunkt in Deutschland / Europa
    - BMW
    - Daimler
    - Volkswagen
    - Bosch
    - …
  - Nordamerika
    - FMC (Ford Motor Company)
    - GM (General Motors Corporation; vertreten durch Opel)
    - …
  - Asien
    - Toyota (Toyota Motor Corporation)
    - Weitere japanische und chinesische Hersteller

4.5 Standardkontext

- Im Automotive-Bereich bestehende und sich stetig weiter entwickelnde Landschaft von Standards
- Kaum große Überschneidungen
- Im Allgemeinen keine Konkurrenz, sondern Ergänzung

MOST

CAN

AUTOSAR

lin

HIS

OSEK/ VDX

ASAM

ISO 26262
Road vehicles - Functional safety

FlexRay

Herstellerinitiative Software (HIS)

- http://www.automotive-his.de
- Aktivitäten
  - Standardsoftwaremodule für Netzwerke
  - Prozessreifegradermittlung
  - Softwaretest
  - Softwaretools
  - Programmieren von Steuergeräten
- Beteiligung an der Entwicklung von Standards
  - Automotive SPICE (Prozessreifegrad)
  - ASAM-Standardisierung (u.a. Diagnose, Kalibrierung)
  - The open Requirements Interchange Format (RIF)
  - EXERPT (RIF-Austauschwerkzeug für DOORS)
- Aussage HIS [HIS_PR07]:
  The results of the standard software group are intermediate solutions. They are in use, but they are being extended by AUTOSAR; The standard software group is therefore currently inactive, the group members contribute actively to AUTOSAR.

4.6 Zeitliche Einordnung

- Im Buch Kindel/Friedrich bis 2009 / Release 3.1
  - Release 3.1 ist z. B. bei Daimler aktuell im Einsatz
- Aktuell verfügbare SPECIFICATIONS (http://www.autosar.org, Stand Juni 2015)
  - Release 4.1
  - Release 4.0
  - Release 3.2
  - Release 3.1
  - Release 3.0
  - Release 2.0

4.7 Einsatzmöglichkeiten von AUTOSAR

- AUTOSAR ist in KEIN Universalansatz für beliebige Softwareprojekte
- Wofür ist AUTOSAR geeignet?
- Wofür ist AUTOSAR nicht geeignet?

4.7.1 Tief eingebettete Automotivesysteme

- Ziel von AUTOSAR: Effiziente Software-Entwicklung für tief eingebettete Systeme im Automotivebereich speziell für den Einsatz im PKW
- PKW: System von vernetzten Steuergeräten mit folgenden Eigenschaften
  - Starke Verteilung
  - Geringe Ressourcen, verursacht durch
  - Sehr hohen Kostendruck

- Herausforderung
  - Entwicklung von aufwendigen Softwaresystemen für eingebettete Systeme mit Prozessoren mit wenig Rechenleistung und geringem Speicher

Herausforderungen

- Umfang der Software im Fahrzeug: Heute oft deutlich mehr als 200 MB Binärcode.
- Realisierung der Funktionen durch ein stark verteiltes System aus Steuergeräten
- Variantenvielfalt
  Ein Zulieferer vertreibt ein Produkt typischerweise an mehrere OEMs mit einer Vielzahl von Anpassungen an die einzelnen Fahrzeugmodelle.
- Starke Heterogenität
  - Unterschiedliche ECUs
    - Von unterschiedlichen Herstellern, entwickelt und produziert an verschiedenen Standorten
    - Mit unterschiedlichen (und alle 2 Jahre wechselnden) Prozessortypen
    - Mit unterschiedlichsten Ressourcen wie Rechenleistung und Speicherplatz
  - Nutzung verschiedener Kommunikationsbusse, die über Gateways verbunden sind
- Die folgenden Ressourcen sind extrem gering:
  - Rechenleistung
  - Speicher (ROM und RAM)
  - Netzwerkbandbreite

Extrem hoher Kostendruck

- Ursache für extrem geringe Ressourcen
- Ursachen für extrem hohen Kostendruck
  - Automobile
    - Fertigung in sehr hohen Stückzahlen über einen verhältnismäßig langen Zeitraum
    - Damit prädestiniert für eine ständige Kostenoptimierung
  - Kundenwahrnehmung
    - Tief eingebettete Systeme (wie Tür-, Licht- oder Motorsteuergeräte) werden nicht als eigenständige Geräte wahrgenommen, sondern vorausgesetzt
    - Kunde ist nicht bereit, dafür mehr zu bezahlen
  - Hohe Stückzahlen
    - 1 Cent für eine höhere Speicherausstattung ergibt 10.000 Euro höhere Produktionskosten für 1 Million Fahrzeuge

AUTOSAR Lösungsansatz

- Klar strukturierte Basismechanismen
- Methoden zur effektiven Nutzung dieser Mechanismen
- Arbeitsschwerpunkte und Ergebnis (vgl. Abschnitt 4.2)
  - Architektur
    - Hochmodulare Struktur nach klarem Konzept zur Realisierung typischer Aufgaben eines verteilten Automotivesystems
      - Kommunikation über Bussysteme wie CAN, LIN oder FlexRay
      - Diagnose
      - Energiemanagement
      - …
  - Methodik
    - Betrachtung des Fahrzeugs zunächst als Ganzes (Systemsicht)
    - Verteilung der Funktionen auf Steuergeräte in einem nachfolgenden Konfigurationsschritt (ECU-Sicht)
  - Anwendungsschnittstellen
    - Standardisierung zur Gewährleistung der Austauschbarkeit von Anwendungen

4.7.2 Nicht im Fokus des Standards

- AUTOSAR wurde nicht für Infotainment-Systeme entwickelt.
- Infotainment-Systeme weisen wesentliche Unterschiede zu tief eingebetteten Systemen auf
  - Eigenschaften von Infotainment-Systemen
    - Geringere Verteilung
      - Eine oder zumindest wenige zentralen Einheiten - Head Unit
        - AUTOSAR: Methodik für verteilte Systeme
      - Verarbeitung von höheren Datenmengen
        - Übertragung von höheren Datenmengen über MOST (nicht in BSW)
          - AUTOSAR (Karosserie, Fahrwerk, Antriebsstrang): Einzelne Werte müssen schnell und sicher übertragen werden, aber keine hochvolumigen Video- oder Audioströme
      - Einbindung mobiler Endgeräte über USB oder Bluetooth
        - AUTOSAR:
          - Dynamische Einbindung von Geräten zur Laufzeit ist nicht vorgesehen
          - Nicht realisierbar mit den extremen Ressourceneinschränkungen

Weitere Gründe AUTOSAR - nicht Infotainment

- Hersteller - Zulieferer
- Kurzlebigkeit insbesondere mobiler Endgeräte im Vergleich zu PKW
- siehe Preislisten

4.7.3 Weitere potenzielle Einsatzgebiete

- Aus technischer Sicht ist AUTOSAR auch für den Einsatz in eingebetteten Systemen anderer Branchen grundsätzlich geeignet
  - Automatisierungstechnik
  - Schienenfahrzeuge
  - Defense (Verteidigungssektor)
- Architektur und Methodik geeignet
  - Teilweise andere BSW, z.B. branchenspezifische Bussysteme
    - AUTOSAR-konforme Anpassung möglich
  - Andere Anwendungsschnittstellen

- Aktuell: Einsatz bei Nutzfahrzeugen
  - LKW
  - Bus
  - Landmaschinen

4.7.4 Rechtliche Aspekte

- Vor Nutzung von AUTOSAR
  - Klärung der rechtlichen Lage
  - Im Zweifelsfall Genehmigung der AUTOSAR-Entwicklungspartnerschaft einholen
- Spezifikationen sind nur zu informatorischen Zwecken veröffentlicht
- Sie dürfen nur von Mitgliedern KOMMERZIELL genutzt werden
- Einschränkung auf Nutzung im Automotivebereich
  - „... the purpose of commercially exploiting AUTOSAR for Automotive Applications ..."
  - Definition des Begriffs „Automotive Applications" in [AS_PMEAGRE08]
    - „(Automotive applications) means applications related to engine powered, land-based, non-railed vehicles for primary transportation purposes."
    - Ausschluss von Schienenfahrzeugen

4.8 Grundlegende Begriffe

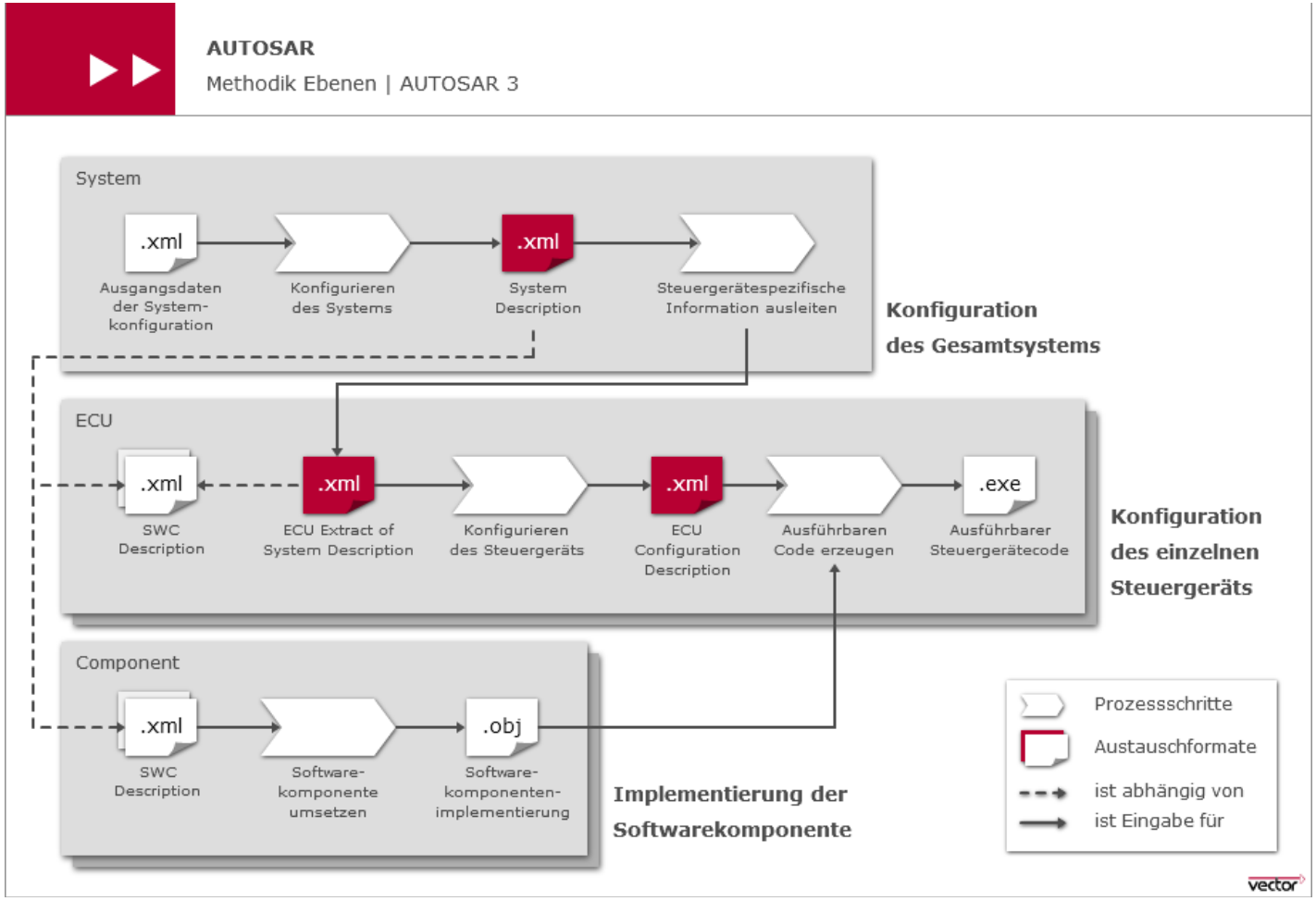
- Kurze Erläuterung der grundlegenden AUTOSAR-Begriffe
- Detaillierung in späteren Abschnitten

Teil II Engineering
5 Die AUTOSAR-Methodik
6 Die Systemsicht/der Virtual Functional Bus
7 Kommunikationsmechanismen
8 Die Steuergerätesicht (ECU-Sicht)
9 Die Basissoftware
10 Performance – oder »Was kostet AUTOSAR?«
11 Variantenmanagement

4.8.1 Methodology

- English: Methodology, Deutsch: Methodik
- Die Methodik beschreibt grundlegende technische Schritte für AUTOSAR-Entwicklungsprojekte.
- Keine komplette Prozessbeschreibung
  - Beispiel: Keine Rollen und Verantwortlichkeiten
- Beschreibung eines Work Product Flow (Arbeitsproduktflusses)
- Abhängigkeiten von Aktivitäten bei der Umwandlung von Arbeitsprodukten
  - Beispiel
    Umwandlung einer Systemsicht (Sicht über alle Funktionen in einem Fahrzeug) in einzelne ECU-Sichten
    Details in Kapitel 5
- Grafische Notation für die Methodik
  - siehe nächste Folie

Methodik


AUTOSAR
Methodik Ebenen | AUTOSAR 3
System
.xml
Ausgangsdaten der System-konfiguration
Konfigurieren des Systems
.xml
System Description
Steuergerätespezifische Information ausleiten
Konfiguration des Gesamtsystems
ECU
.xml
SWC Description
.xml
ECU Extract of System Description
Konfigurieren des Steuergeräts
.xml
ECU Configuration Description
Ausführbaren Code erzeugen
.exe
Ausführbarer Steuergerätecode
Konfiguration des einzelnen Steuergeräts
Component
.xml
SWC Description
Software-komponente umsetzen
.obj
Software-komponenten-implementierung
Implementierung der Softwarekomponente
Prozessschritte
Austauschformate
ist abhängig von
ist Eingabe für
vector

4.8.2 Virtual Functional Bus (VFB)

- Deutsch: Virtueller Funktionsbus, Abkürzung: VFB
- Beschreibung der Kommunikationsbeziehungen zwischen Softwarekomponenten auf der Systemebene
- Details in Kapitel 6
- Grafische Notation

4.8.3 Software Component (SW-C)

- Deutsch: Softwarekomponente, Abkürzung: SW-C
- Modellierungselemente des VFB
- Container (Strukturelemente), die über Ports mit der Außenwelt kommunizieren
- SW-Cs kommunizieren miteinander ebenfalls über Ports

4.8.4 Port

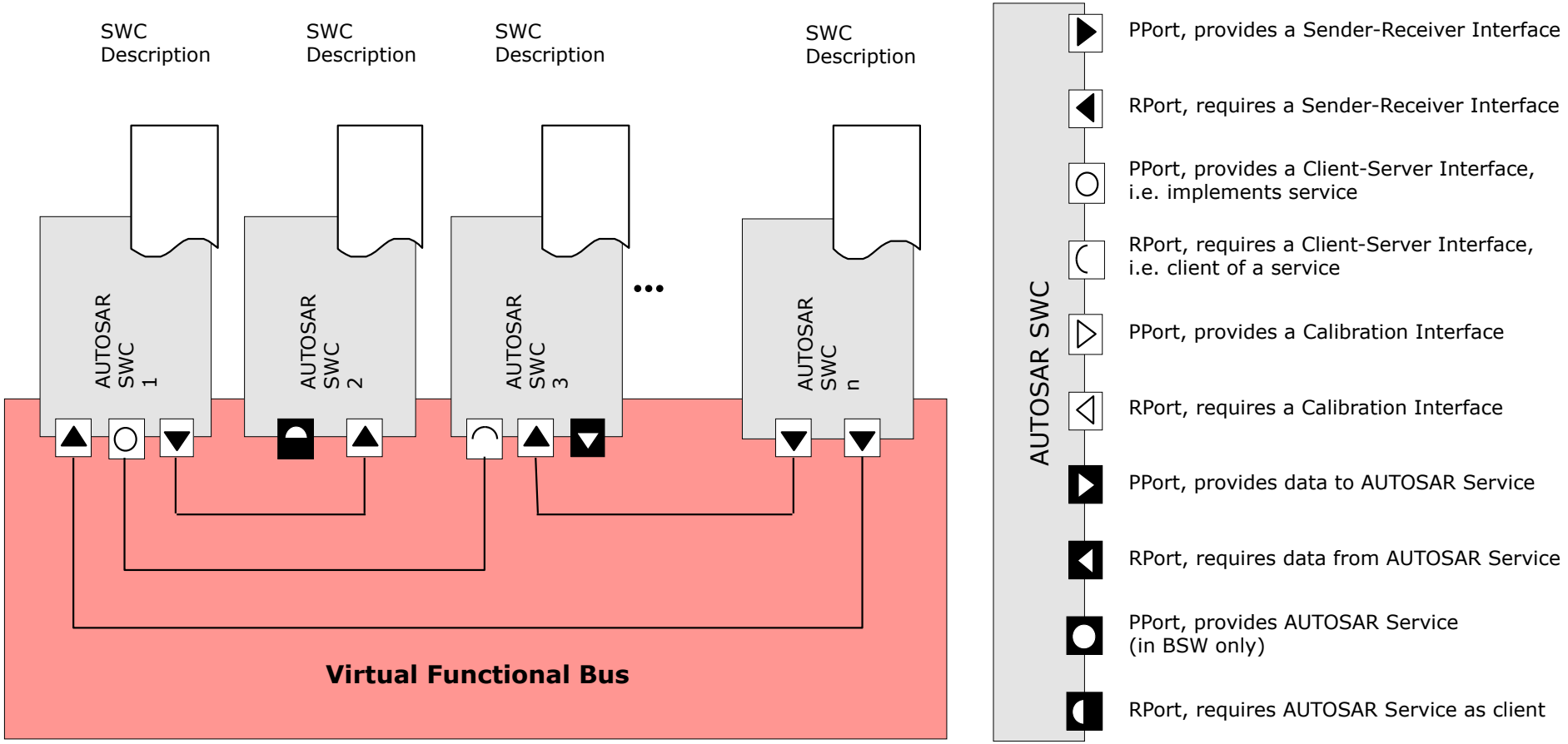
- Deutsch: Port, Abkürzung: Keine
- Interaktionspunkte von Softwarekomponenten (SW-C).
- PPorts (provide ports): Stellen bereit
- RPorts (require ports): Benötigen
- Näher bestimmt wird der Port durch das Port Interface (oder kurz Interface). Die Zuordnung findet mittels Konfiguration und nicht im Programmcode statt.
- Nur Ports, die kompatible Interfaces besitzen, können miteinander verbunden werden.

4.8.5 Port Interface (Interface)

- Deutsch: Interface, Abkürzung: Keine
- Drei Typen von Interfaces in AUTOSAR
  - Client/Server (C/S)
    Bei C/S-Interfaces können Clients bei einem Server Operationen ausführen.
  - Sender/Receiver (S/R)
    Über S/R-Interfaces werden Daten ausgetauscht, dabei sind beliebige (1:m und n:1) Beziehungen zwischen Sendern und Empfängern möglich.
  - Calibration
    Mithilfe des Calibration Interfaces können statische Kalibrierungsdaten abgefragt werden
- Der Interfacetyp gibt die verwendete Kommunikationsmethode und/oder den geplanten Verwendungszweck vor. Jedes einzelne Interface wird dann genauer durch die konkreten Operationen (bei C/S) oder Datenelemente (bei S/R oder Calibration), die es bereitstellt, bestimmt.

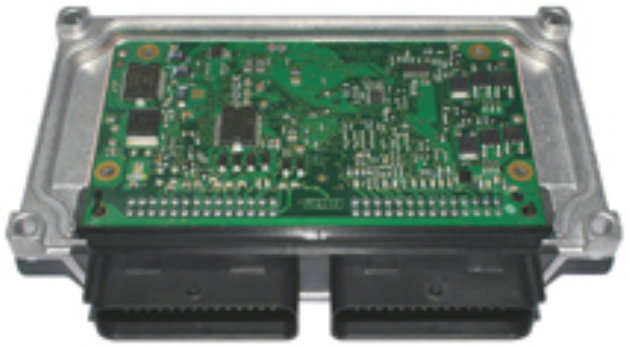
- Stand 2009, aktueller Stand siehe nächste Folie

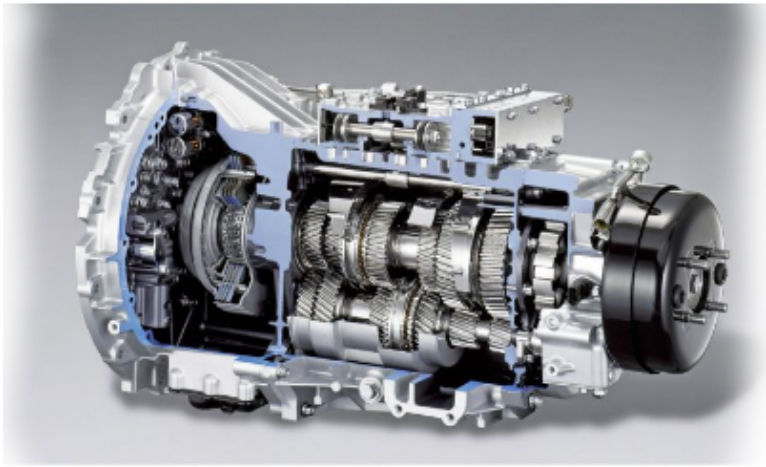
Ports and Interfaces



Application Interfaces - Getriebesteuerung

- Kommunikationsbeziehungen der Getriebesteuerung
- ?
- ?





Application Interfaces - Getriebesteuerung

- Kommunikationsbeziehungen der Getriebesteuerung
  - Mögliche Kommunikations-beziehungen (Auswahl)
    - Tempomat
    - Motorsteuerung
    - Sollwertgeber (Gangschaltung)
    - ESP (oft über Tempomat)
    - Rückfahrkamera
    - Rückfahrscheinwerfer
    - Kombiinstrument
    - Einparkhilfe
    - Rückspiegel
    - …
  - Realisierte Kommunikations-beziehungen je nach Hersteller und Modell
- Kommunikationsmechanismen
  - Sender / Receiver
    - Getriebesteuerung sendet „Rückwärtsgang eingelegt“
    - Getriebesteuerung ist egal wer das liest bzw. ob das überhaupt jemanden interessiert
  - Client / Server
    - Tempomat gibt an Getriebesteuerung Anweisungen
      - „4. Gang einlegen“
      - „Hochschalten“
    - Tempomat erwartet Ausführung der Anweisung und ggf. Rückmeldung

4.8.6 Runnable Entity (Runnable)

- Deutsch: Runnable, Abkürzung: Keine
- Runnables sind Codesequenzen in den Softwarekomponenten, die durch Events, wie beispielsweise Timer oder den Empfang von Daten, aktiviert werden können.
- Sie bedienen die Ports der Softwarekomponenten und senden oder empfangen Daten und implementieren Client- oder Serveroperationen.
- Details in Kapitel 7

4.8.7 Run-Time Environment (RTE)

- Deutsch: Laufzeitumgebung, Abkürzung: RTE
- Beschreibung der Systemsicht mithilfe des VFB
- Umsetzung auf die Sichten der einzelnen ECUs:
  - Erzeugung der RTE mithilfe des sogenannten RTE-Generators
- Aufgabe der RTE
  - Umsetzung der abstrakt auf VFB-Ebene modellierten Kommunikationsverbindungen
    - Softwarekomponenten untereinander
    - Softwarekomponenten mit der Basissoftware.
  - Die Basissoftware liegt unter der RTE und abstrahiert von der jeweils ECU-spezifischen Hardware.
- Details RTE: Abschnitt 8.3.2
- Details Basissoftware: Kapitel 9

4.9 Die Spezifikation

- Effektiver Einstieg in die Dokumente der AUTOSAR-Spezifikation
  - Ablagestruktur
    - Verzeichnisse
    - „Codierung“ der Dateinamen
  - Empfehlung von Dokumenten für einen ersten Überblick
- Entwickler von Basissoftwaremodulen
  - Erläuterung zum Aufbau der Softwarespezifikationsdokumente.
- Systematische Einarbeitung in die AUTOSAR-Spezifikation sehr wichtig
  - Release 3.1 fast 7900 Seiten in 141 Dateien
  - Kaum Orientierungshilfen wie z. B. übergeordnete Inhaltsbeschreibung
- Laden der AUTOSAR-Spezifikation von www.autosar.org zu Informationszwecken
- Kommerzielle Nutzung nur durch AUTOSAR-Mitglieder

4.9.1 Struktureller Aufbau

- Strukturierung: Kategorien, Verzeichnisse und Dateinamen
- Kategorien der Dateien auf der AUTOSAR-Internetseite
  - Main
  - SW-Architecture
  - Conformance Testing
  - Application-Interfaces
  - Darunter weitere Strukturierungsebenen
  - Kategorien für einzelne Dokumente: Standard und Auxiliary

4.9.1 Struktureller Aufbau

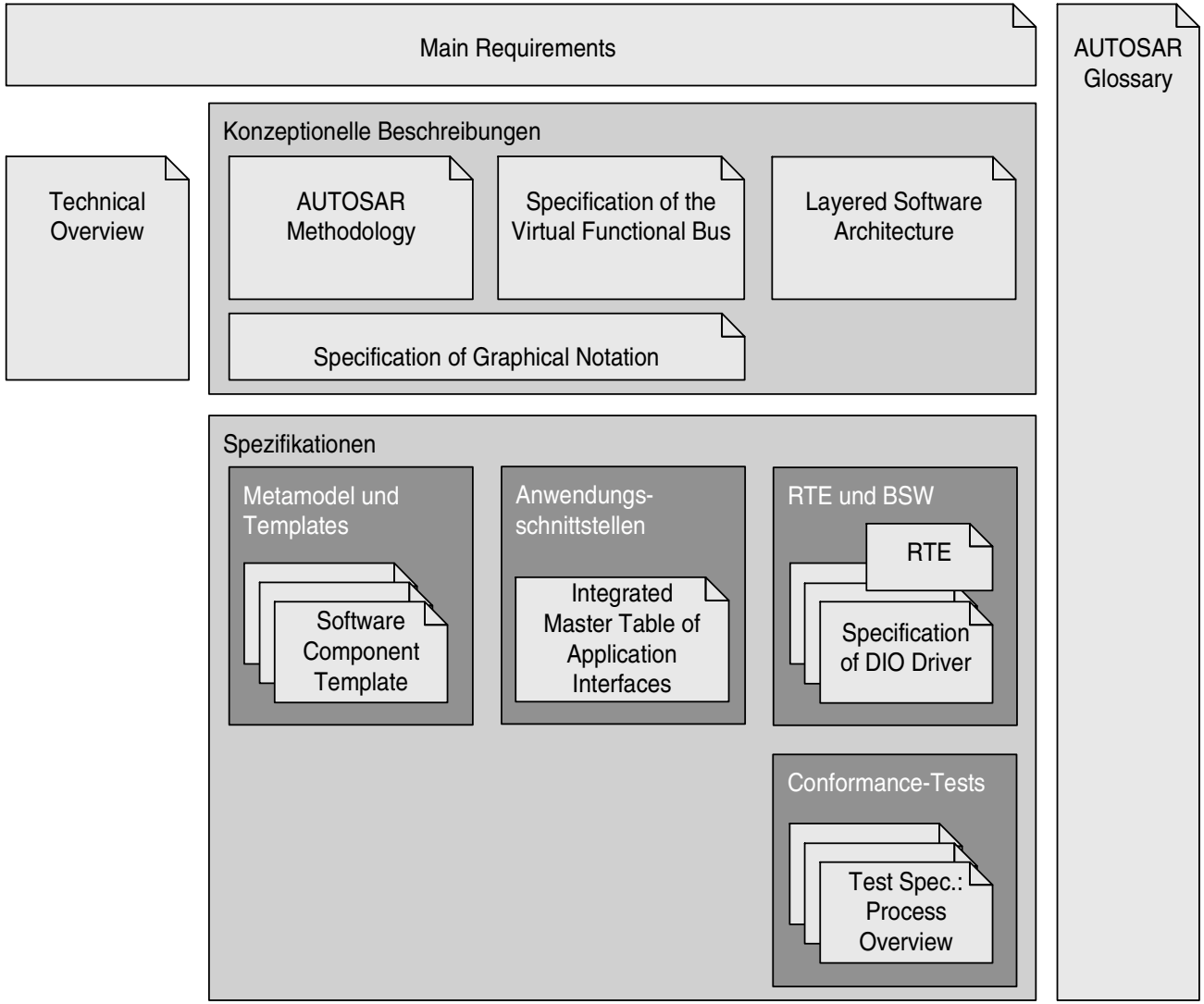
- Codierung der Dateinamen: Schluss auf Inhalt
  - Alle Dateinamen beginnen mit dem Präfix AUTOSAR_
  - Im Allgemeinen zweites Präfix
    - SRS_/RS_: Requirements
    - SWS_: Spezifikationen
    - DS_/CTSpec_: Konformitätstests
    - ApplicationInterfaces_: Anwendungsschnittstellen
  - Ausnahmen
    - Einige Dateien haben kein zweites Präfix, dafür aber ein Suffix wie z. B. „Template“: AUTOSAR-Templates, Details in Kapitel 5
    - Einige grundlegende AUTOSAR-Dokumente wie AUTOSAR_Methodology.pdf oder AUTOSAR_Glossary.pdf besitzen weder ein zweites Präfix noch ein Suffix

4.9.2 Inhaltlicher Aufbau

- Einstiegspunkte und weitere Orientierung
- Verzeichnisstruktur und Dateinamen reichen im Falle von AUTOSAR nicht aus, um den besten Einstiegspunkt zu finden
- Gerade die Dateien, die keinem Schema entsprechen, sind hier oft besonders hilfreich.

Abb. 4–4

Überblick AUTOSAR-Dokumentation



Einstiegsdokumente (1)

- AUTOSAR_MainRequirements.pdf
  - Grundlegende AUTOSAR-Ziele
  - Formale Beschreibung
    - Wichtige Grundlage für die Referenzierung in den übrigen Dokumenten
    - Nur bedingt für den Einstieg geeignet
- AUTOSAR_TechnicalOverview.pdf
  - Grober Abriss der AUTOSAR-Ziele und der benutzten Methoden
  - Bester Einstieg in die AUTOSAR-Dokumentation

Einstiegsdokumente (2)

- Einarbeitung in AUTOSAR unabhängig von der späteren Spezialisierung
  - AUTOSAR_Methodology.pdf (AUTOSAR Methodology)
    Beschreibt alle wichtigen Schritte, um in einem AUTOSAR-Projekt von der Systemsicht bis zu den Executables auf den konkreten ECUs zu gelangen.
  - AUTOSAR_SWS_VFB.pdf (Specification of the Virtual Functional Bus):
    Beschreibt die Modellierungsmöglichkeiten, die der VFB bietet (wie Softwarekomponenten, Runnables, Kommunikation etc.).
  - AUTOSAR_LayeredSoftwareArchitecture.pdf (Layered Software Architecture):
    Beschreibt die geschichtete AUTOSAR-Architektur als eines der wichtigsten AUTOSAR-Konzeptelemente. Es erläutert auch die Zusammenhänge von Modulen in speziellen Bereichen wie z. B. der Kommunikation, der Diagnose und der Speicherabstraktion.

- AUTOSAR_GraphicalNotation.pdf (Specification of Graphical Notation)
  Extrakt der grafischen Notationen aus Methodik und VFB

Dokumente zu Spezifikationsschwerpunkten

- Metamodell und Template-Spezifikationen
  - Suffix Template
  - Beispiel: AUTOSAR_SoftwareComponentTemplate.pdf
- ApplicationInterfaces
  - Zweites Präfix ApplicationInterfaces_
  - Beispiel AUTOSAR_ApplicationInterfaces.xls
- RTE und BSW-Spezifikationen
  - Zweites Präfix SWS_
  - Beispiele: AUTOSAR_SWS_RTE.pdf, AUTOSAR_SWS_DIO_Driver.pdf)
  - Auflistung aller Basissoftwaremodule in AUTOSAR_BasicSoftwareModules.pdf.
  - Details Basissoftware: Abschnitt 8.4
  - Details RTE: Abschnitt 8.3.2
- Conformance-Tests (für RTE und BSW-Module)
  - Zweites Präfix DS_ oder CTSpec_
  - Beispiel AUTOSAR_CTSpec_Process_Overview.pdf
  - Details in Abschnitt 9.6

4.9.3 Aufbau der Softwarespezifikationen

- Spezifikationen der 46 Basissoftwaremodule (Stand 2009, Release 3.1)
  - Requirementsdokument (SRS): Anforderungen
  - Spezifikationsdokument (SWS): Umsetzung der Anforderungen
- Für die Entwicklung von Basissoftwaremodule reichen normalerweise die Spezifikationsdokumente
- Requirementsdokument bei Unstimmigkeiten
- Einheitlicher Aufbau der Dokumente.
  - Alle Dokumente beginnen mit einer genauen Identifikation des Dokumentes durch Titel, Version, Autor usw. Danach folgen Änderungsverfolgung und rechtliche Hinweise.
  - Der eigentliche Inhalt beginnt hinter dem Inhaltsverzeichnis, das auf Seite 4 oder 5 zu finden ist. Der Inhalt ist immer wie in Tabelle 4–1 dargestellt gegliedert.

Tabelle 4–1 AUTOSAR-Spezifikationsdokument (1)

Kapitel		Inhalt
Nr.	**Name**	
1	Introduction and functional overview	Ein kurzer Überblick, der die Funktion des Moduls beschreibt und meist einen Hinweis auf seine Einordnung in die AUTOSAR-Architektur gibt.
2	Acronyms and abbreviations	Abkürzungen, die im Dokument verwendet werden und nicht in *AUTOSAR_Glossary.pdf* zu finden sind oder diese neu definieren.
3	Related documentation	Dokumente, die mit diesem Dokument in Verbindung stehen. Dies können AUTOSAR-Dokumente, aber auch andere Standards und Normen sein.
4	Constraints and assumptions	Annahmen, die getroffen wurden, beispielsweise dass nur eine Instanz dieses Moduls existiert oder besondere zeitliche Anforderungen oder Einschränkungen. Des Weiteren wird in diesem Kapitel angegeben, in welchen Automotivbereichen dieses Modul einsetzbar ist (wie beispielsweise Innenraum, Motorsteuerung oder Infotainment).
5	Dependencies to other modules	Beziehungen zu Quellcode und Header-Dateien anderer Module.
6	Requirements traceability	Zeigt die Erfüllung von Requirements durch Spezifikationselemente auf.
7	Functional specification	Spezifiziert das Verhalten des Moduls entsprechend den Requirements. Dies umfasst typischerweise das Verhalten als solches, die Initialisierung und auch die Fehlerbehandlung.
8	API specification	Beinhaltet die Schnittstellenspezifikation. Diese umfasst Typdefinitionen, Funktionsdefinitionen (auch Callback-Funktionen) und Funktionen, die vom Basissoftware-Scheduler zyklisch aufgerufen werden, sowie Schnittstellen von anderen Teilen des Systems, die vorausgesetzt werden.

Tabelle 4–1 AUTOSAR-Spezifikationsdokument (1)

Kapitel		Inhalt
Nr.	**Name**	
9	Sequence diagrams	Mithilfe von Sequenzdiagrammen wird das Verhalten des spezifizierten Moduls veranschaulicht.
10	Configuration specification	Führt die Konfigurationsmöglichkeiten des Moduls anhand von Konfigurationsparametern auf.
11	Changes to Release 1	Änderungen im Vergleich zu AUTOSAR-Release 1. Dies beinhaltet alle Änderungen bis zum aktuellen Release.