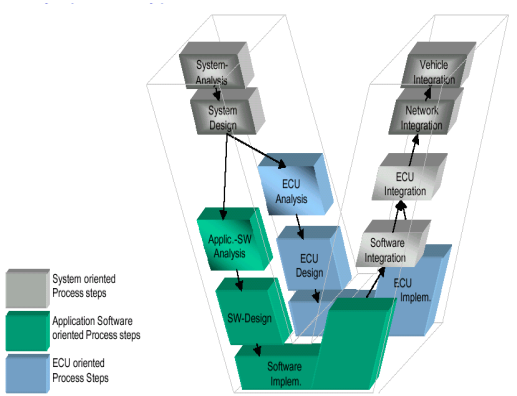
TECHNISCHE UNIVERSITÄT DRESDEN

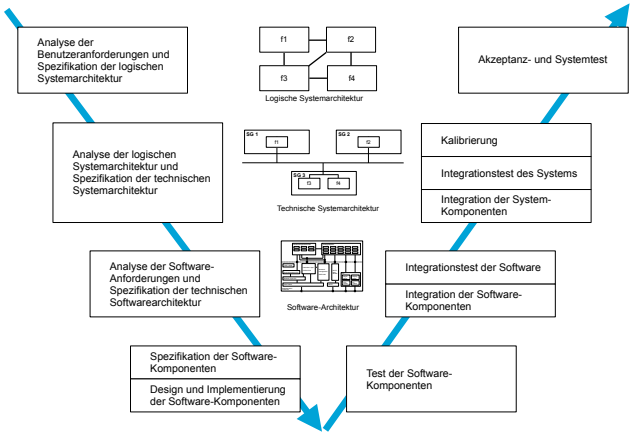
Vorlesung
Automotive Software Engineering
Teil 6 SW-Entwicklung (2-2)
Sommersemester 2015

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

Vorlesung Automotive Software Engineering

Motivation und Überblick		
Beispiele aus der Praxis	SW-Entwicklung	Normen und Standards
	E/E-Entwicklung	
	Das Automobil	
	Die Automobil-herstellung	
	Die Automobil-branche	





Lernziele SW-Entwicklung

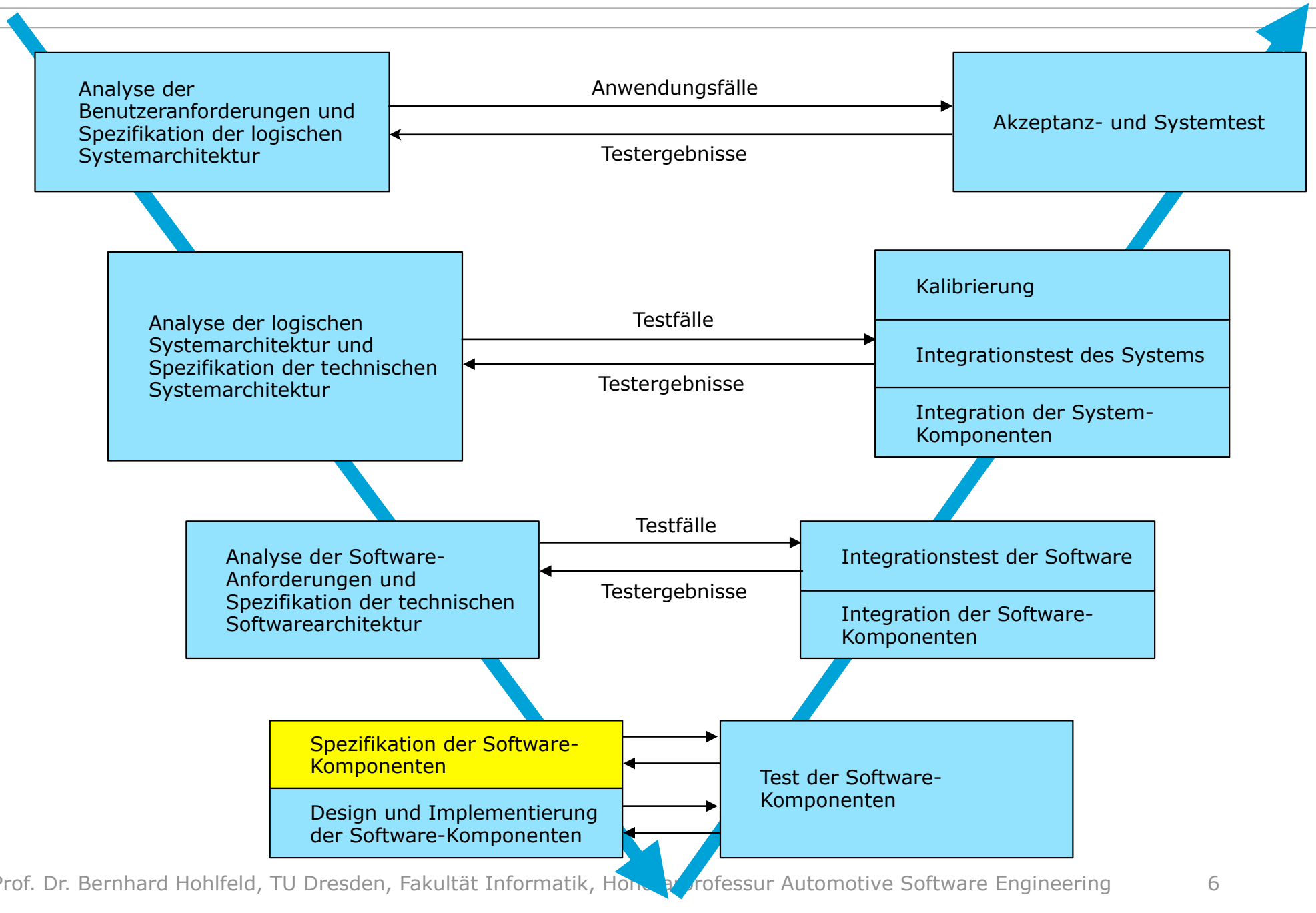
- Vorgehensmodelle kennen
- Grundbegriffe des Systems Engineering kennen
- Den Kernprozess zur Entwicklung von elektronischen Systemen und Software im Fahrzeug verstehen
- Das V-Modell zur Software-Entwicklung in einer speziellen Ausprägung mit zahlreichen Beispielen kennen
- Die Unterstützungsprozesse zur Entwicklung von elektronischen Systemen und Software im Fahrzeug kennen

6. SW-Entwicklung

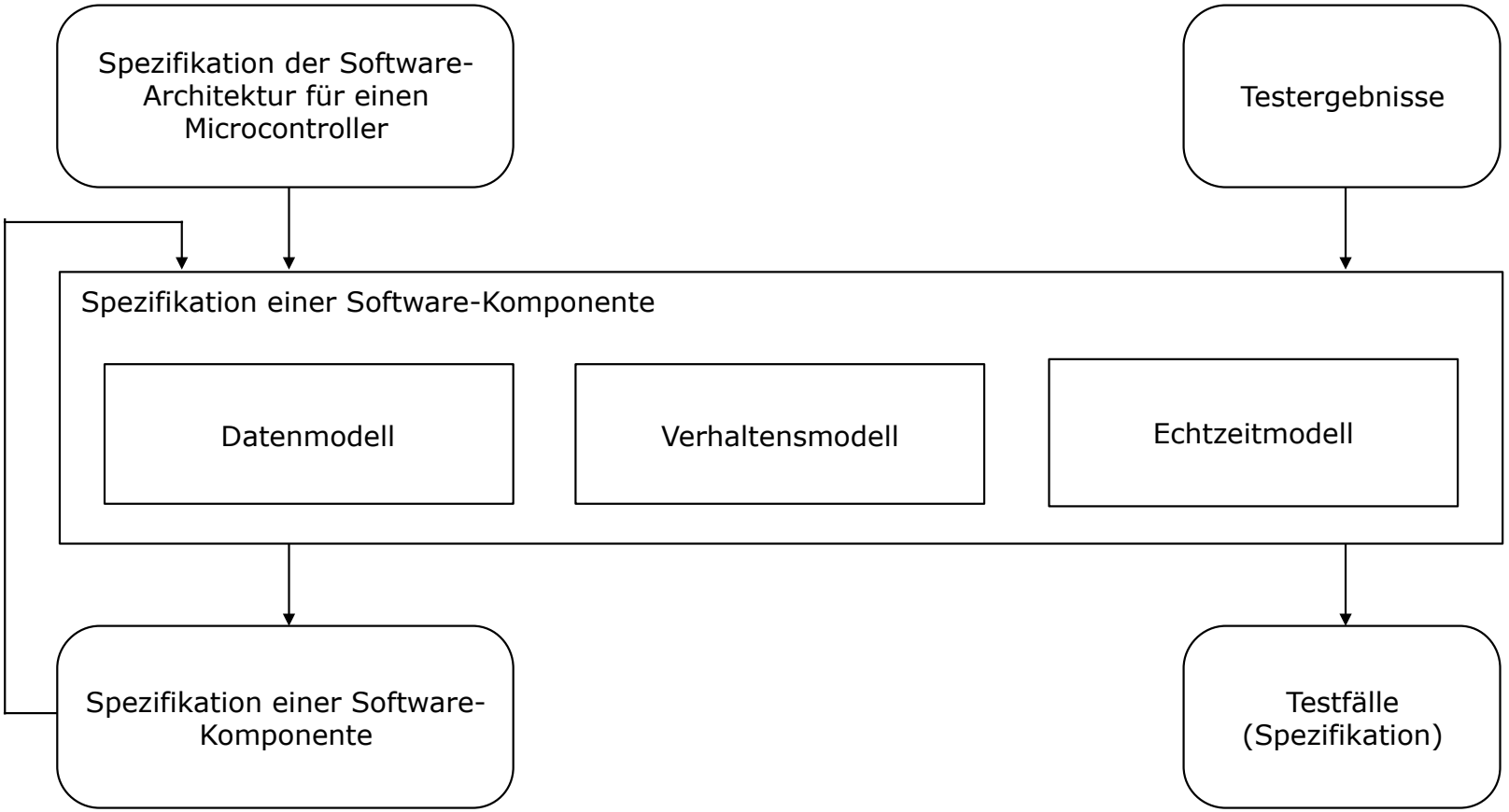
1. Vorgehensmodelle
2. **Kernprozess**
3. Unterstützungsprozesse

6. SW-Entwicklung / 2. Kernprozess

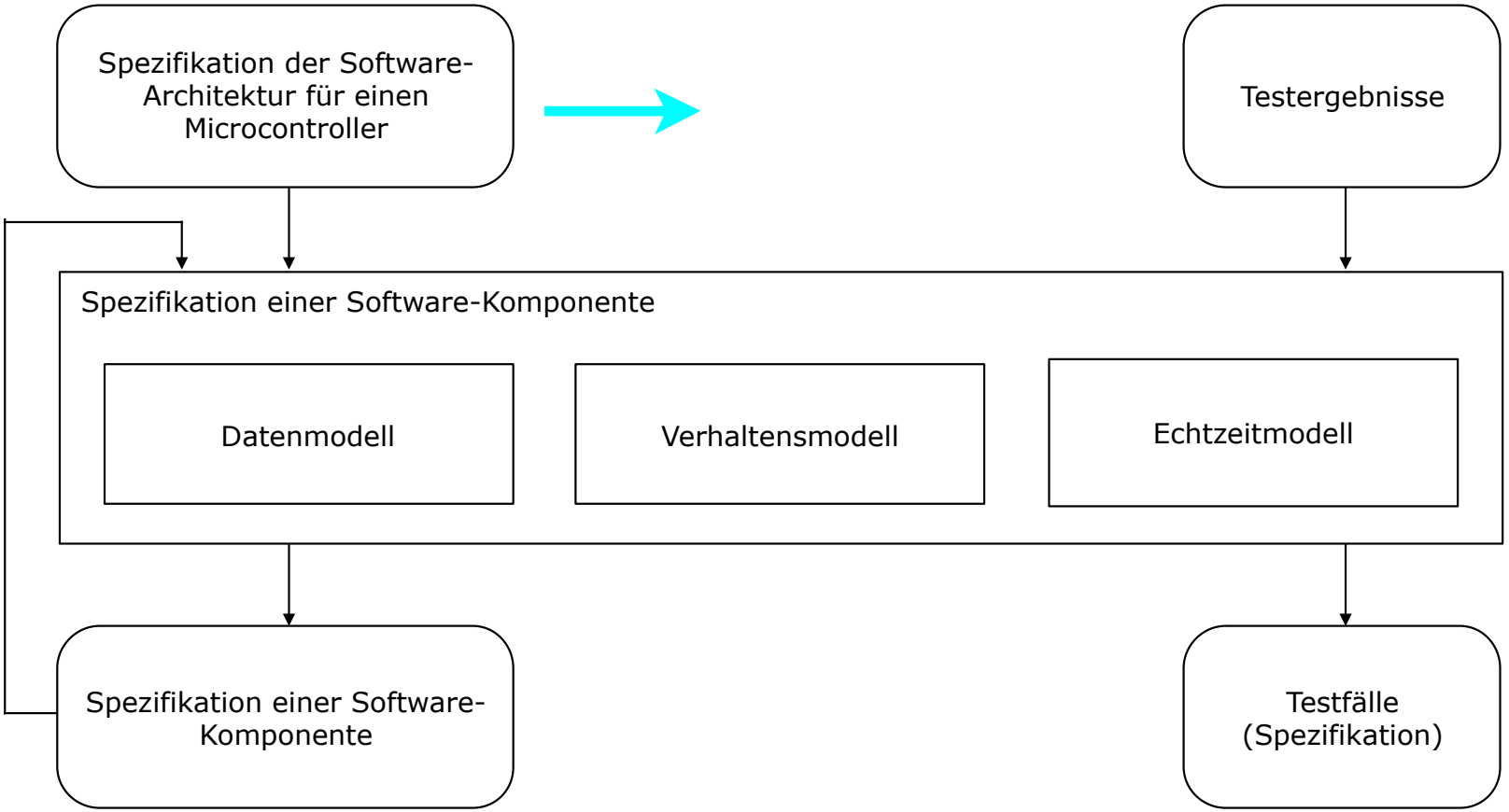
1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. **Spezifikation der Software-Komponenten**
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. Kalibrierung
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

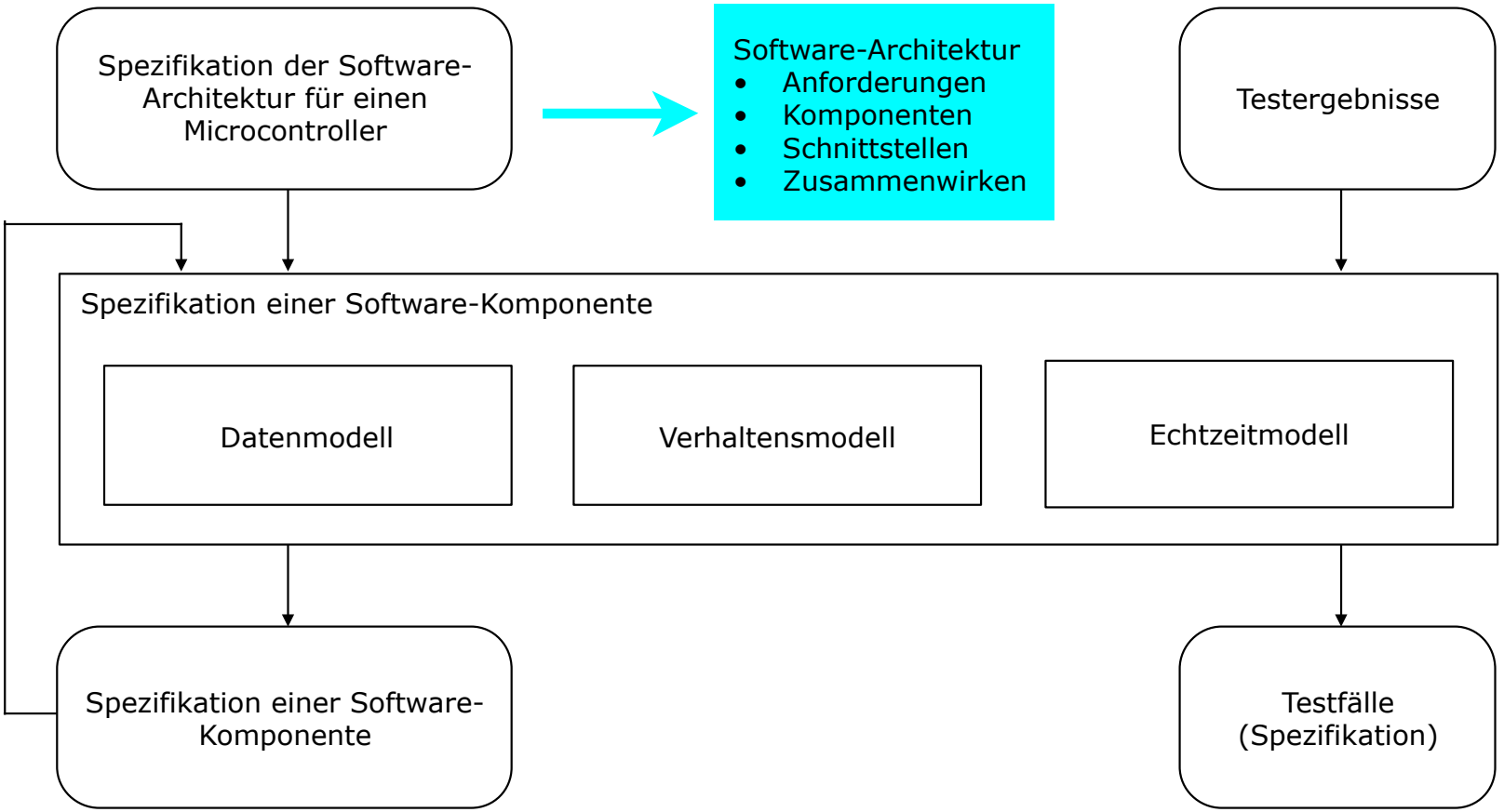
Spezifikation der Software-Komponenten



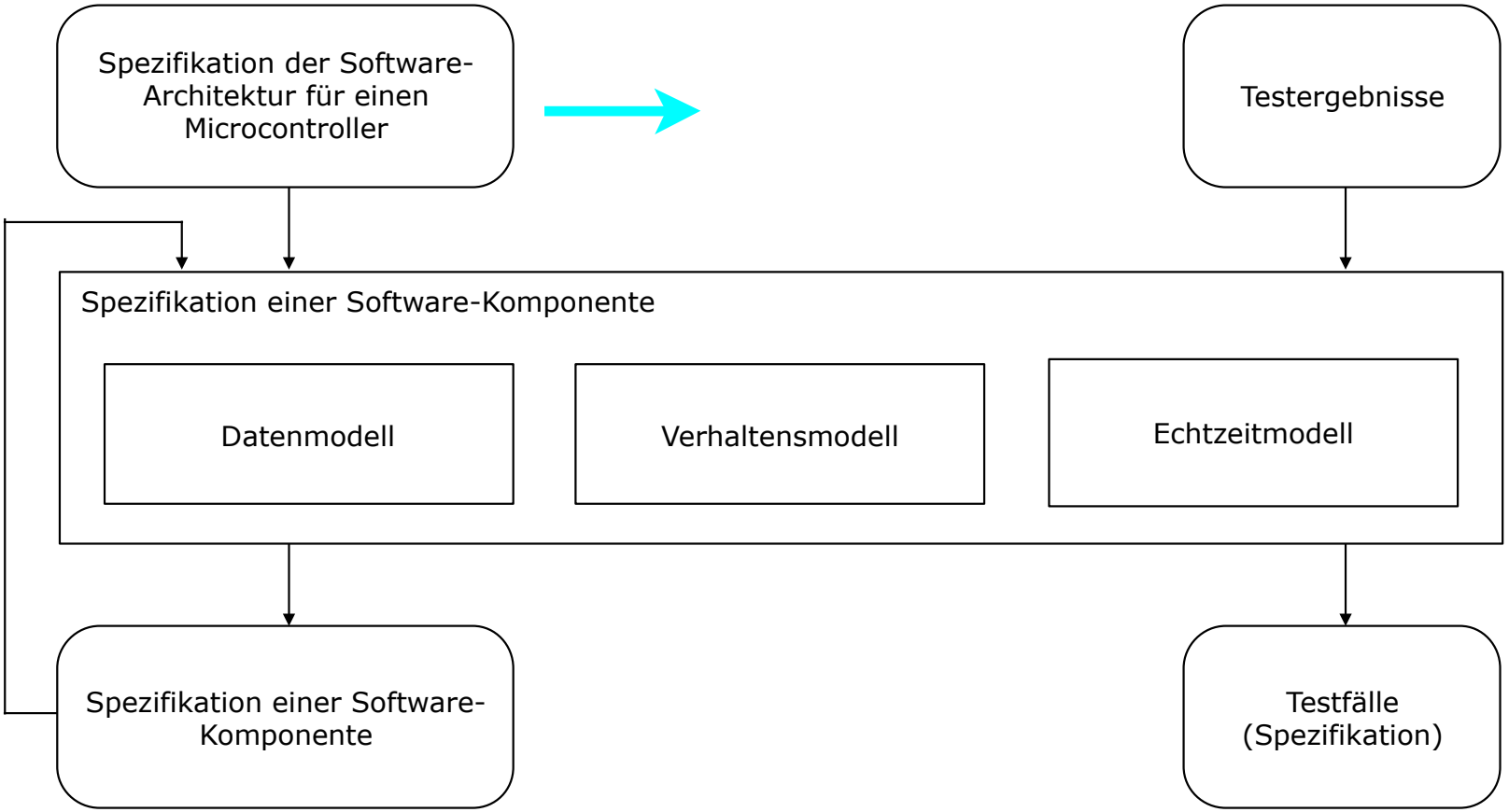
Spezifikation der Software-Komponenten



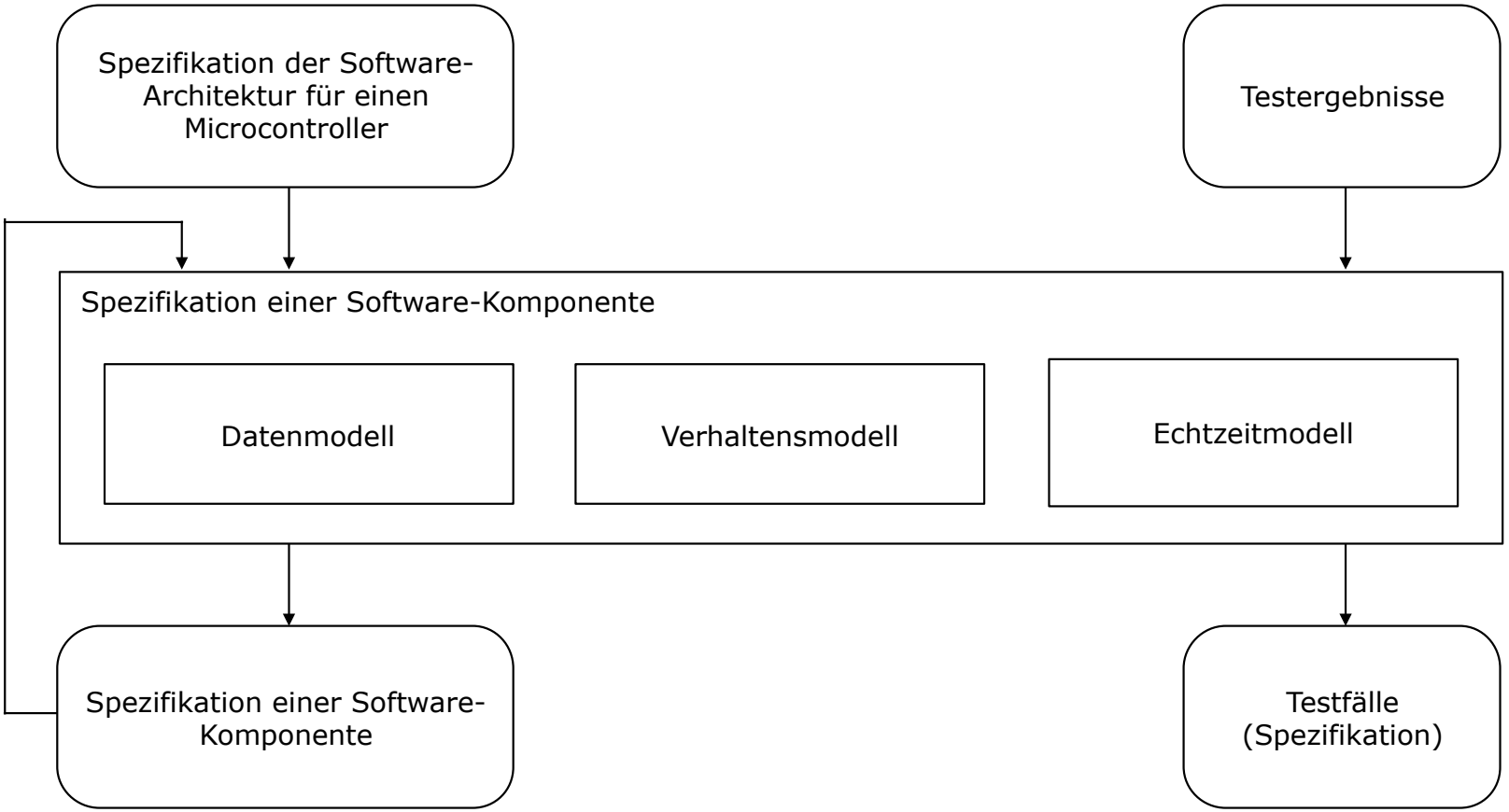
Spezifikation der Software-Komponenten



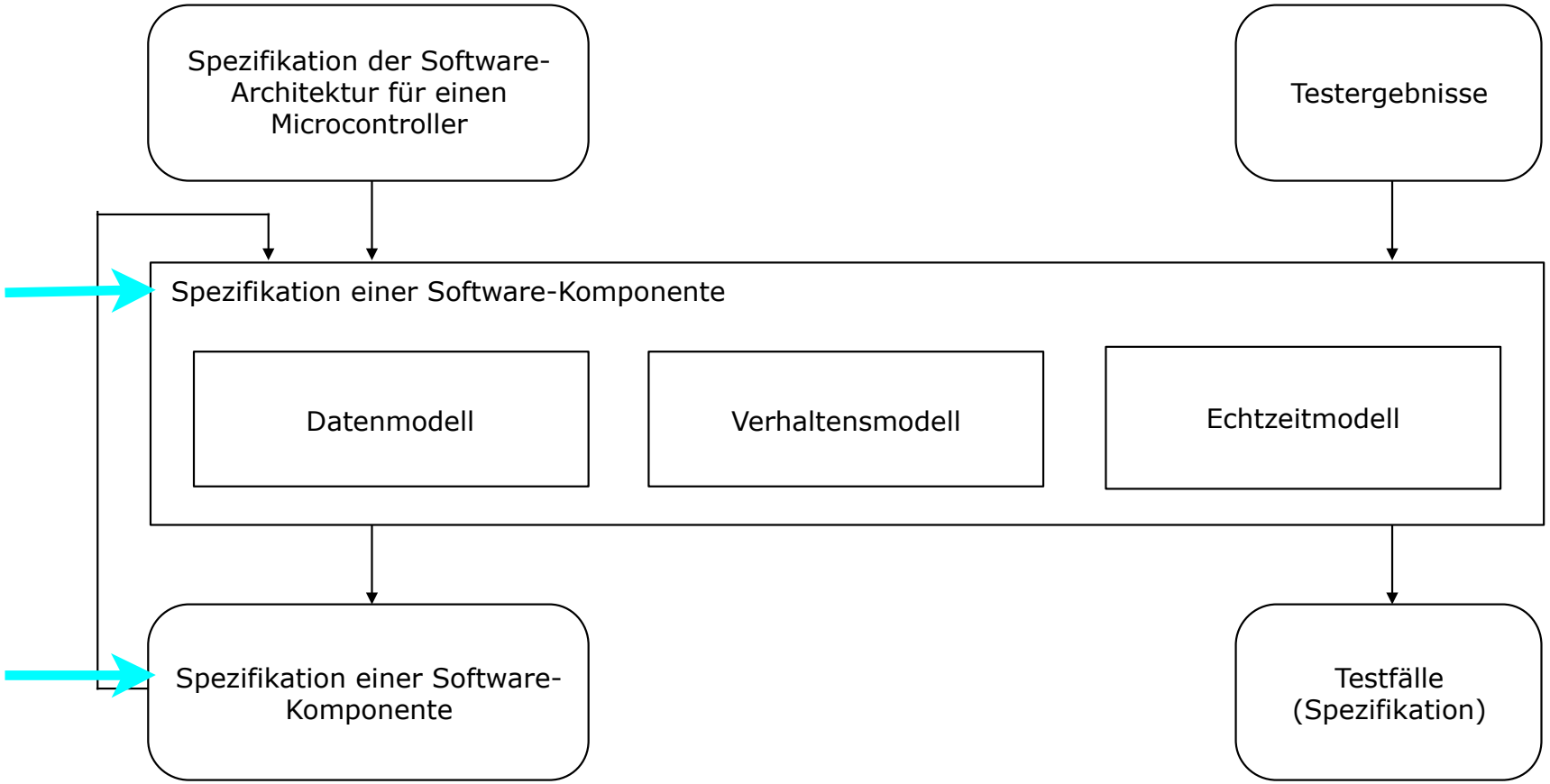
Spezifikation der Software-Komponenten



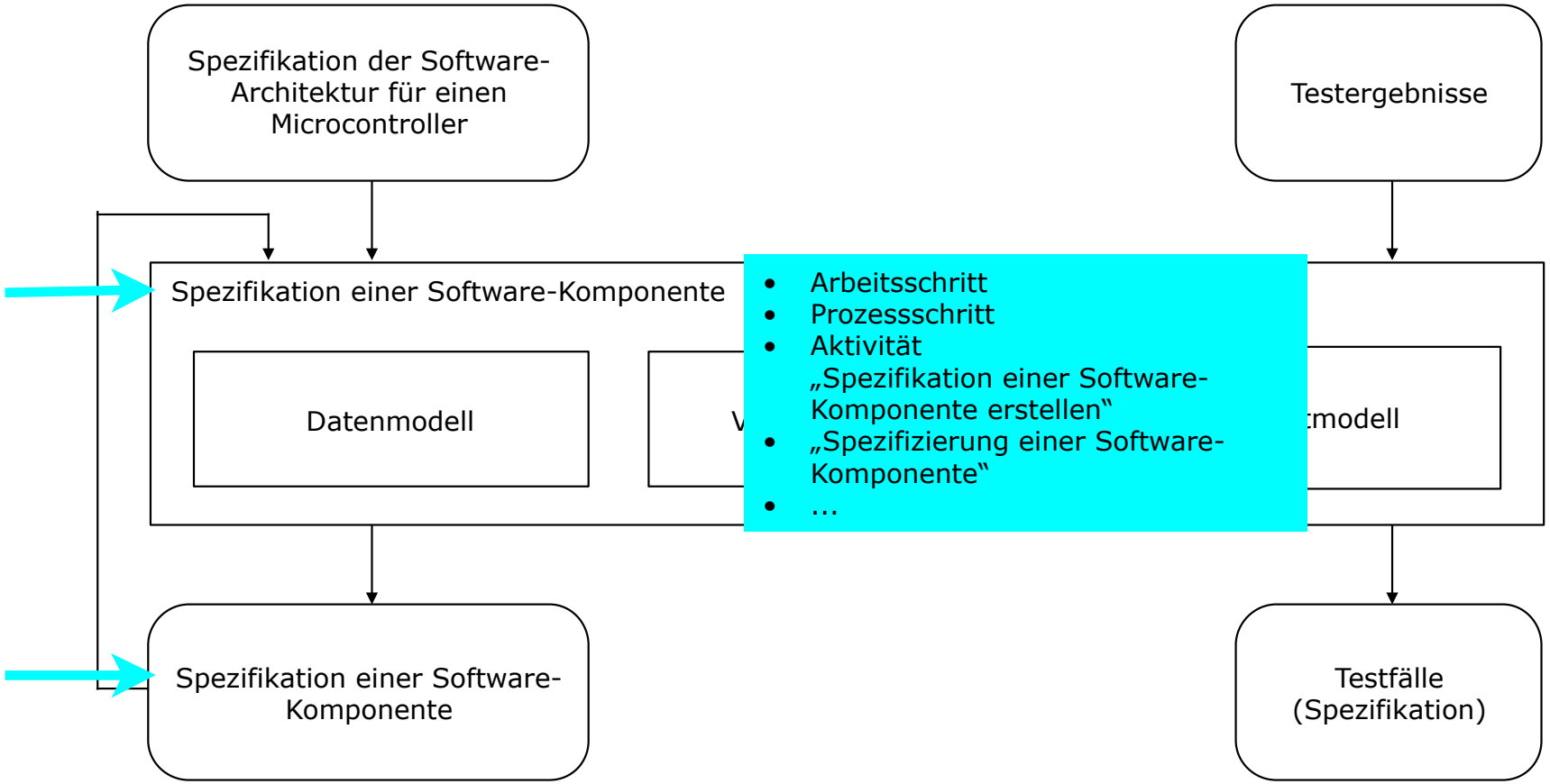
Spezifikation der Software-Komponenten



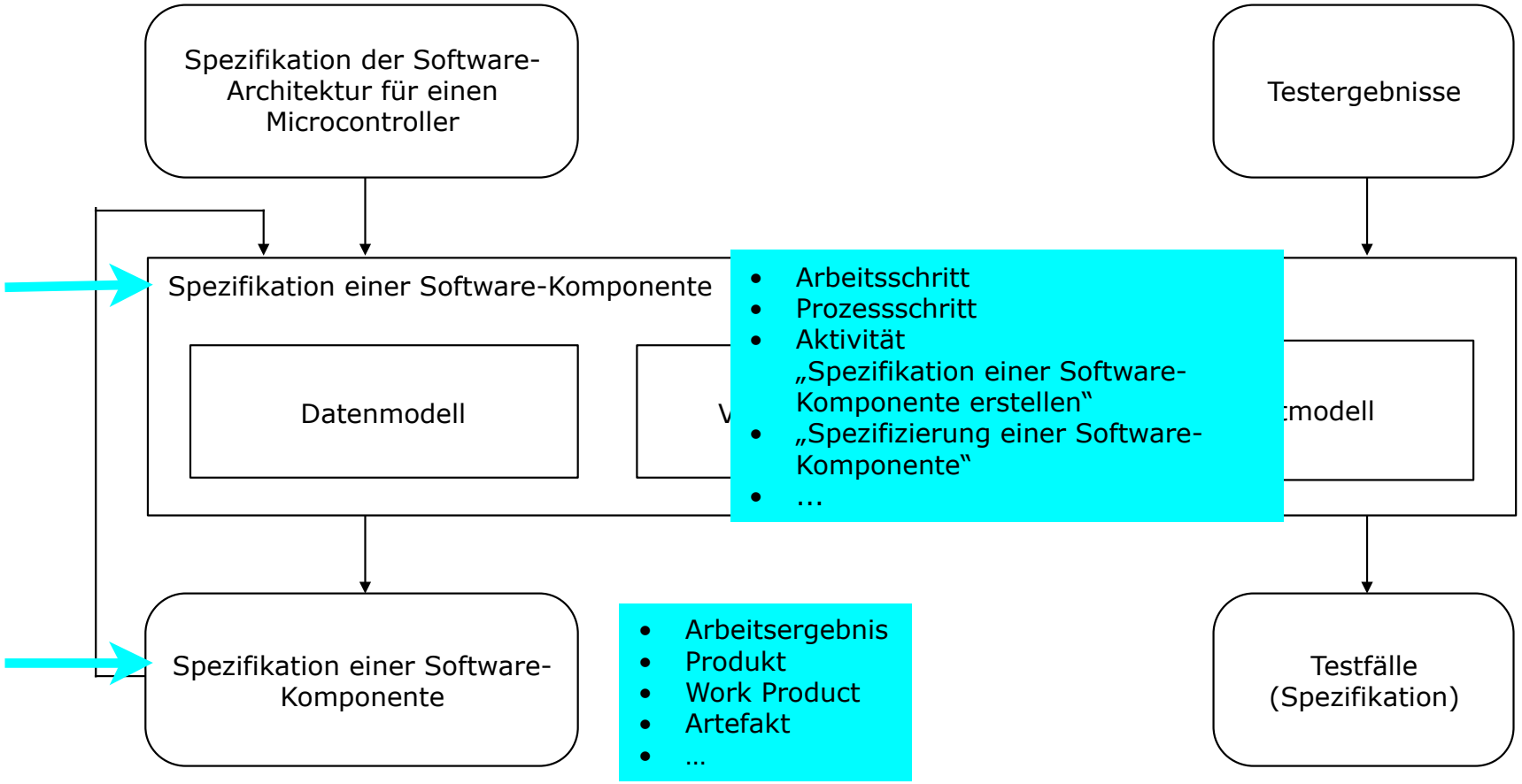
Spezifikation der Software-Komponenten



Spezifikation der Software-Komponenten



Spezifikation der Software-Komponenten



Spezifikation und Spezifizierung

- …ierung bezeichnet den Vorgang und ...ation das Ergebnis.
- siehe Duden Fremdwörterbuch 8. Auflage 2005

...a|ti|on/...ie|rung

Die konkurrierenden Suffixe für abstrakte Substantive stehen oft ohne Bedeutungsunterschied nebeneinander. Sie sind von Verben auf ...ieren abgeleitet und bringen häufig das Ergebnis einer Handlung oder Tätigkeit bzw. die Handlung selbst zum Ausdruck:
- Isolation/Isolierung
- Konfrontation/Konfrontierung

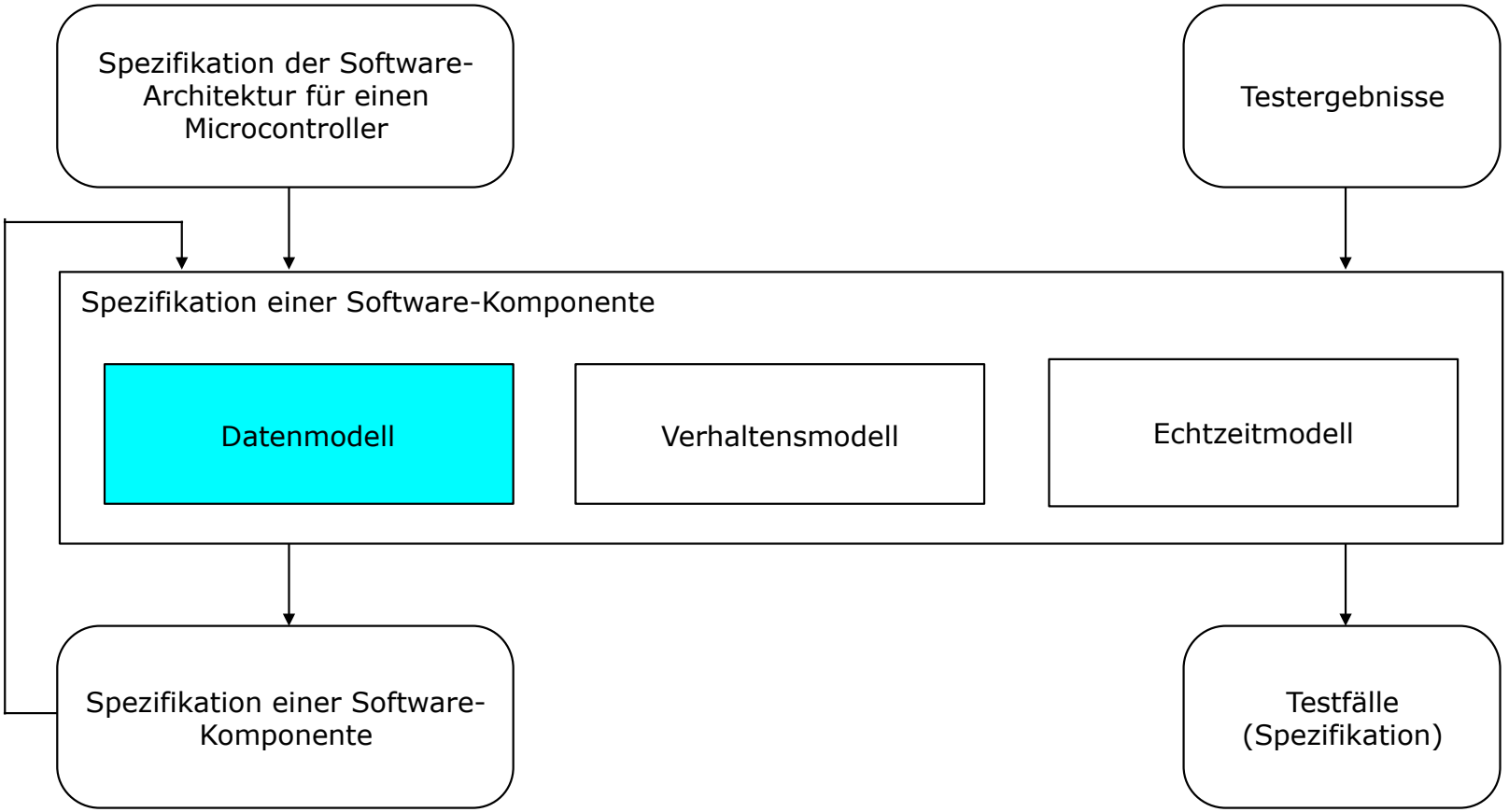
Im Allgemeinen zeigen sich aber Bedeutungsnuancen:

...ation
die; -, -en
⟨*lat.* ...atio, Gen. ...ationis (→ *fr.* ...ation)⟩
Suffix, das vorrangig das Ergebnis einer Handlung oder Tätigkeit, seltener das Geschehen selbst bezeichnet:
- Indikation
- Kanalisation
- Klassifikation
- Sozialisation

...ierung
die; -, -en
⟨*fr.* ...er bzw. ...ir; *dt.* ...ung⟩
Suffix, das eine Handlung oder Tätigkeit, seltener deren Ergebnis bezeichnet:
- Kanalisierung
- Klassifizierung
- Resozialisierung
- Restaurierung

- In der Praxis verschwimmen diese Unterschiede: WM-Qualifikationsspiele

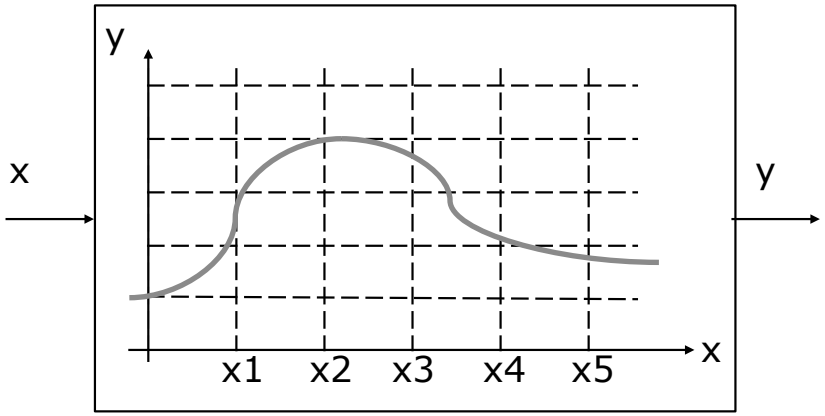
Spezifikation der Software-Komponenten



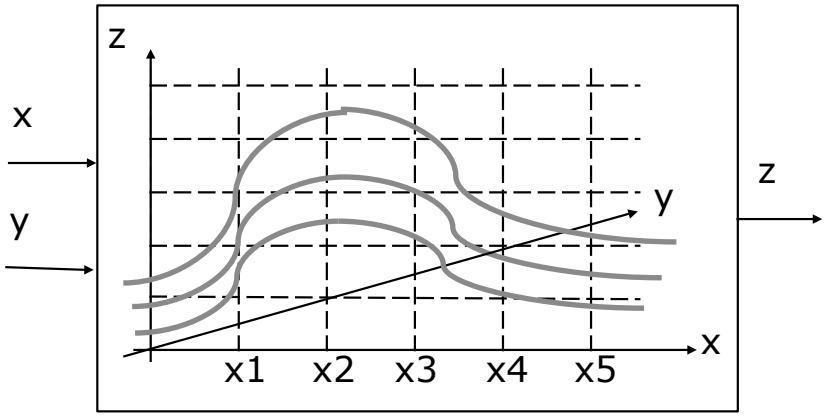
Spezifikation der Software-Komponenten

Grafische Darstellung:

Kennlinie



Kennfeld



Tabellarische Darstellung:

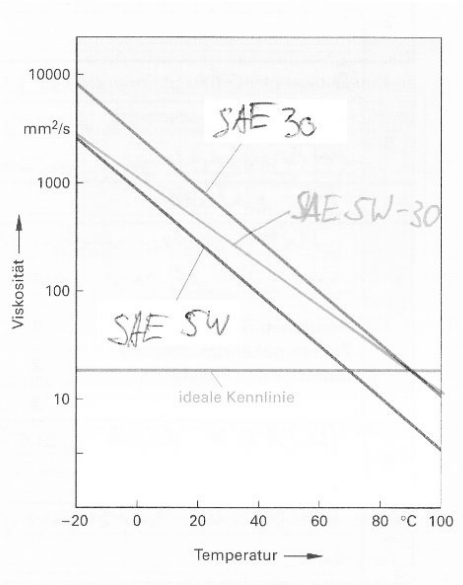
x1	x2	x3	x4	x5
y1	y2	y3	y4	y5

	x1	x2	x3	x4	x5
y1	z11	z12	z13	z14	z15
y2	z21	z22	z23	z24	z25
y3	z31	z32	z33	z34	z35

Beispiele

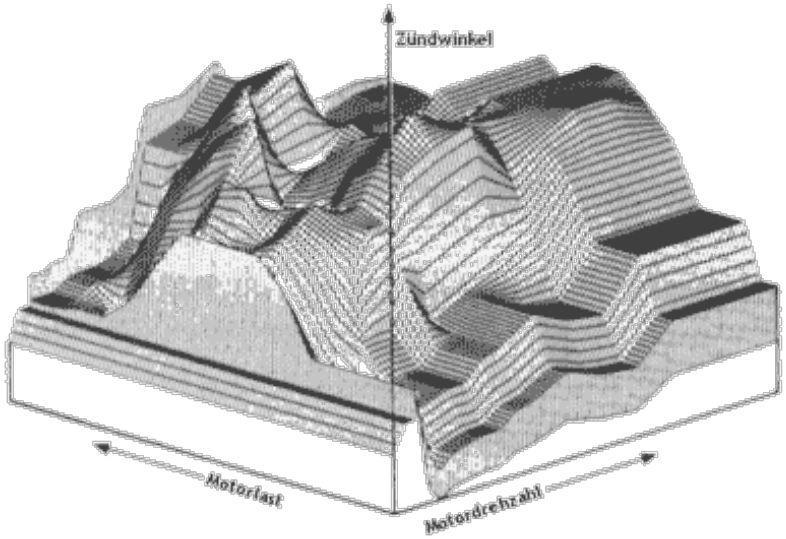
Kennlinie

- 2-dimensional
- Geschwindigkeit y in Abhängigkeit von Motordrehzahl x bei konstantem Gang
- Viskosität in Abhängigkeit von Temperatur

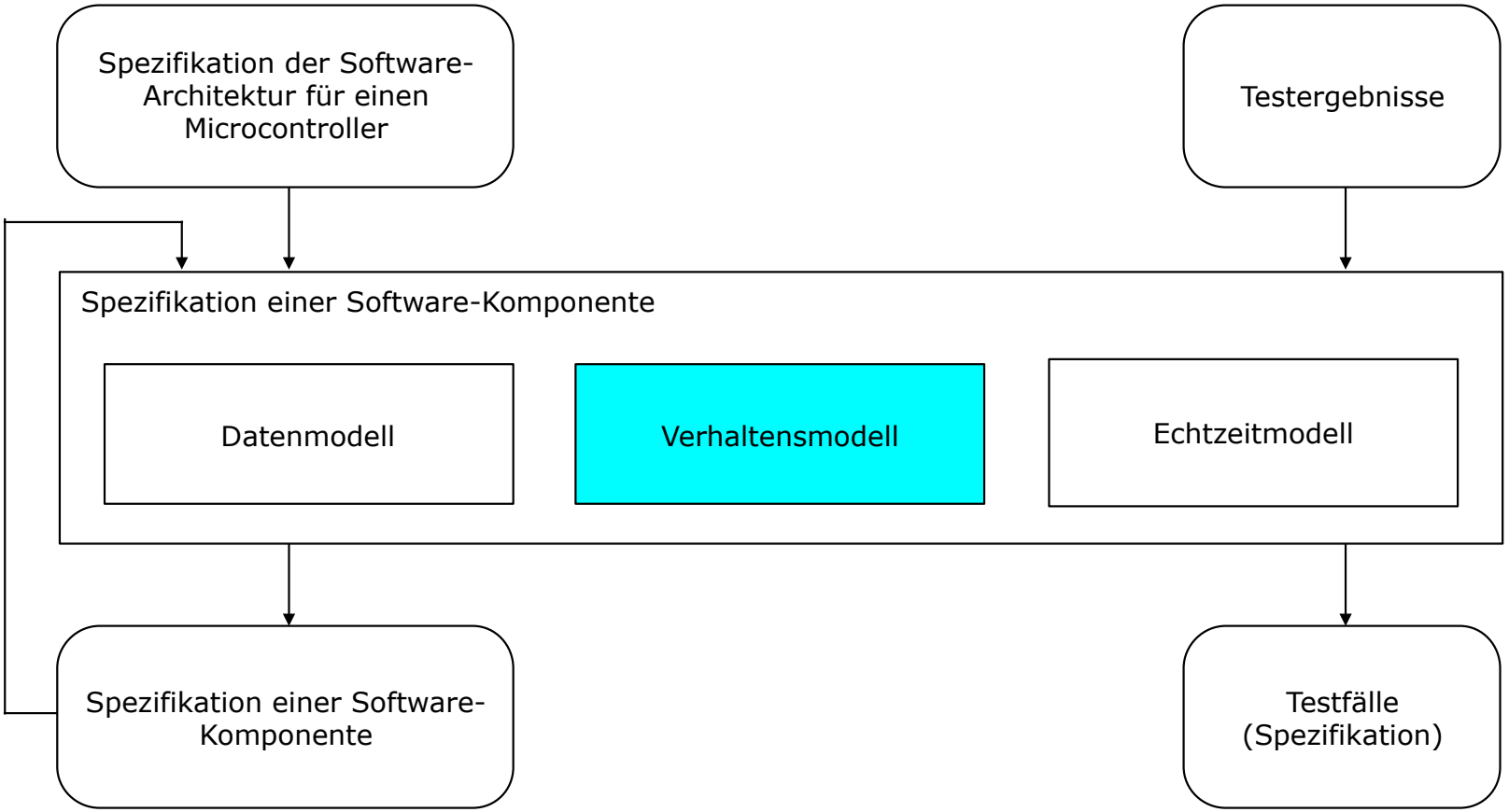


Kennfeld

- mehr-dimensional, i.a. 3-dimensional
- Geschwindigkeit z in Abhängigkeit von Motordrehzahl x und Gang y
- Zündungskennfeld: Zündwinkel in Abhängigkeit von Motordrehzahl und Motorlast



Spezifikation der Software-Komponenten



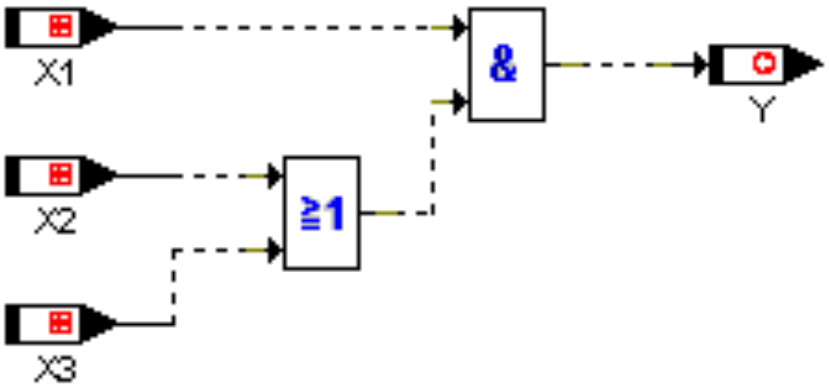
Spezifikation der Software-Komponenten

- Spezifikation des Datenflusses
- Beispiel: Datenfluss für eine boolesche und eine arithmetische Anweisung in ASCET-SD Blockschaltbildern

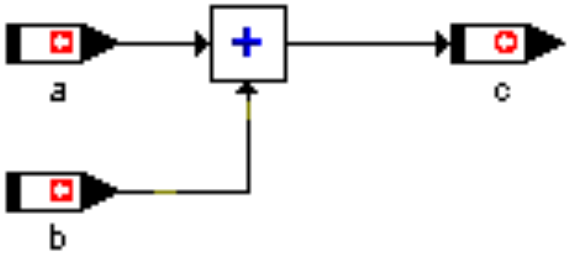
- Y := X1 & (X2 || X3)

c := a + b

Boolesche Anweisung



Arithmetische Anweisung



Erläuterung

Y = X1 & (X2 || X3)

&: Logisches „und“

&	0	1
0	0	0
1	0	1

||: Exclusives „oder“

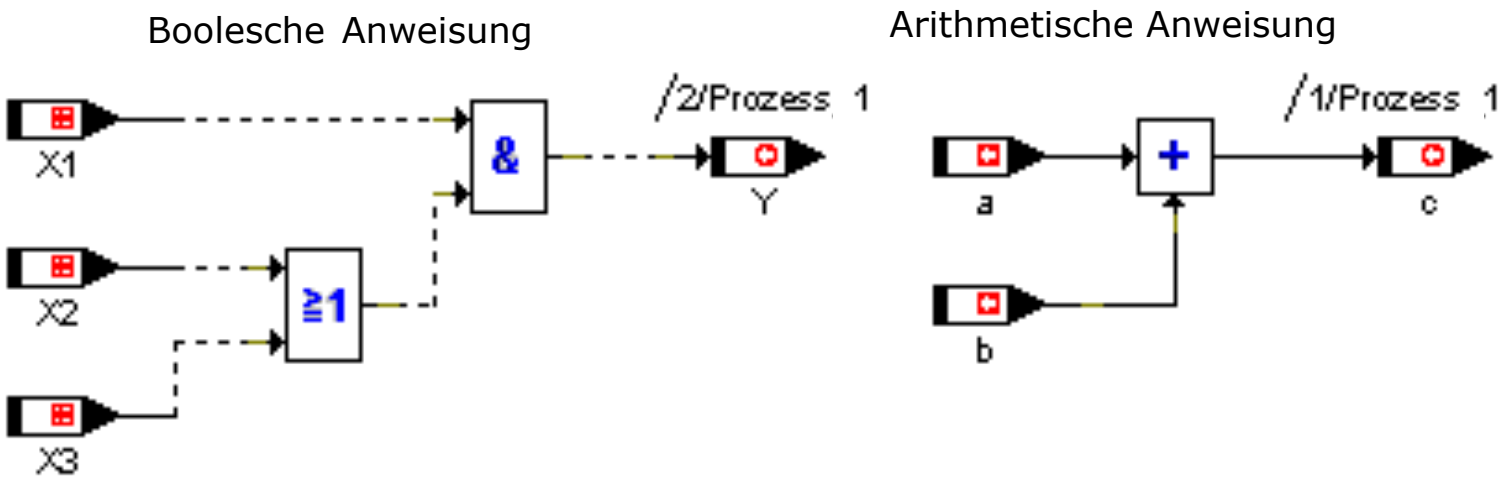
||	0	1
0	0	1
1	1	0

ASCET-SD

- ASCET (Advanced Simulation and Control Engineering Tool, früher: ASCET-SD) von ETAS ist eine Produktfamilie für die modellbasierte Entwicklung eingebetteter Automobilsoftware. Im Gegensatz zu Simulink, bei dem die einfach durchführbare Simulation im Vordergrund steht und eine Codegenerierung möglich ist, steht bei ASCET die Codegenerierung im Vordergrund. Als Entwicklungswerkzeug für Steuergeräte-Software wird es überwiegend von Funktions- und Software-Entwicklern von Automobilherstellern und deren Zulieferern eingesetzt, um eingebettete Software für Steuerfunktionen und Regelalgorithmen zu entwickeln. Solche Steuergeräte sind im Fahrzeug dann später verantwortlich für Funktionen wie Steuerung oder Regelung von Verbrennungsmotoren oder Hybridantrieben, Automatikgetrieben, Antiblockiersystemen (ABS), Fahrstabilitätssystemen (ESP), Fensterhebern, Schiebedächern, Scheibenwischern, etc.
- Quellen:
  - http://de.wikipedia.org/wiki/ASCET
  - http://www.etas.com/ascet

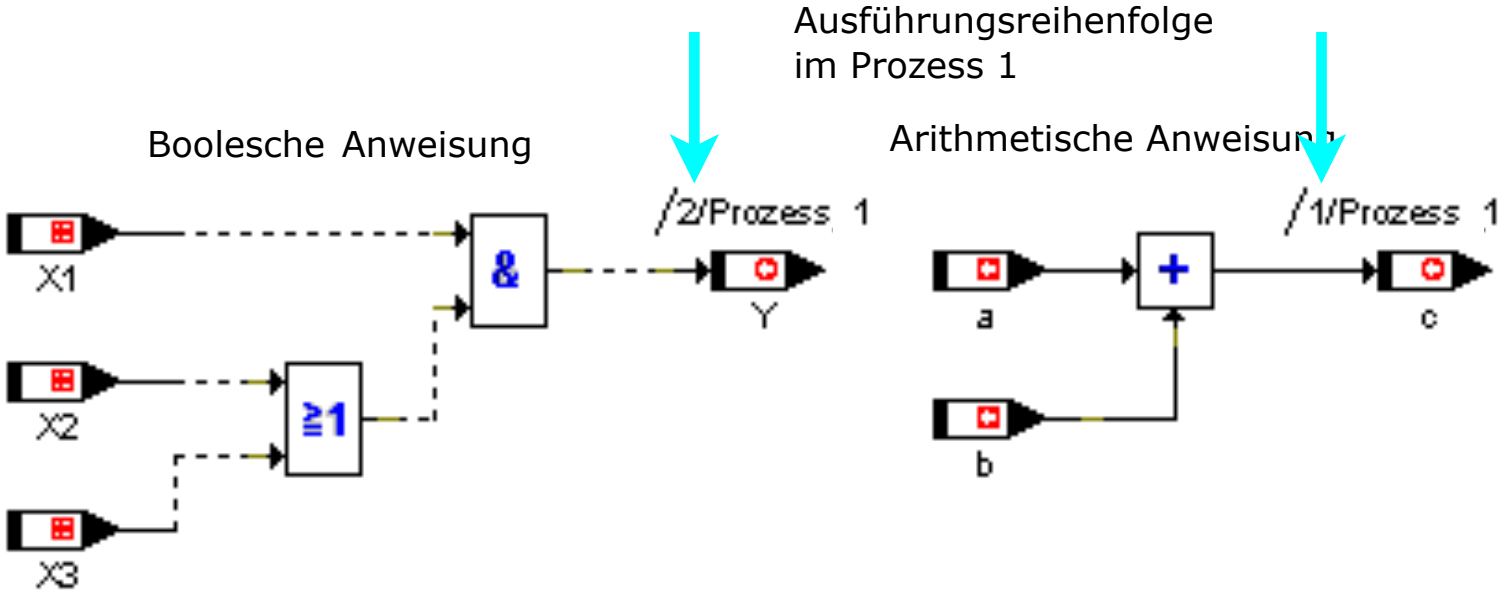
Spezifikation der Software-Komponenten

- Spezifikation des Kontrollflusses
  - Klassische Ansätze aus der Programmierung, z. B. Struktogramme
  - Ansätze aus dem objekt-orientierten Entwurf, z. B. für die Aufrufstruktur zwischen Software-Komponenten
- Kontrollfluss zur Festlegung der Ausführungsreihenfolge in ASCET-SD

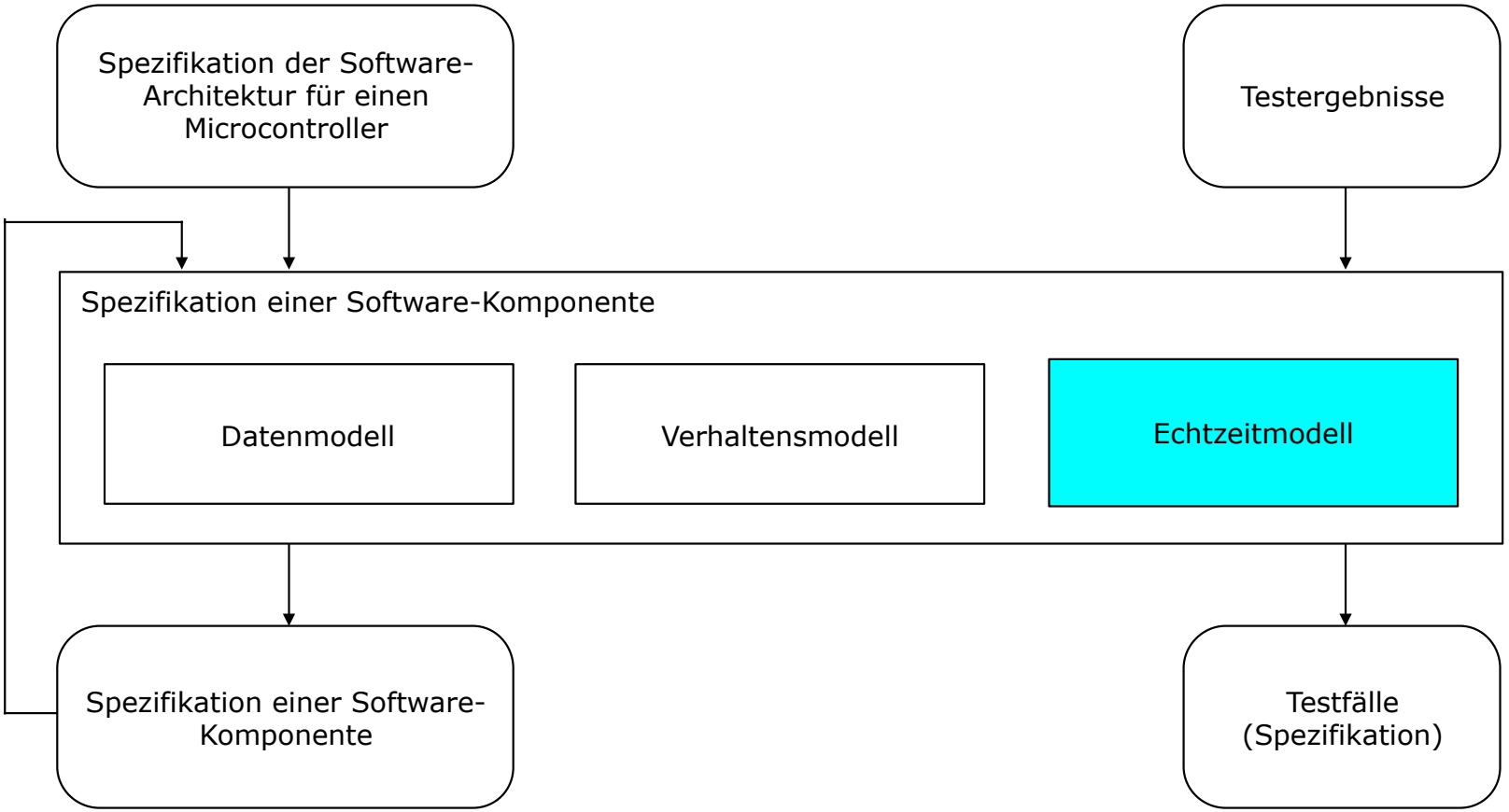


Spezifikation der Software-Komponenten

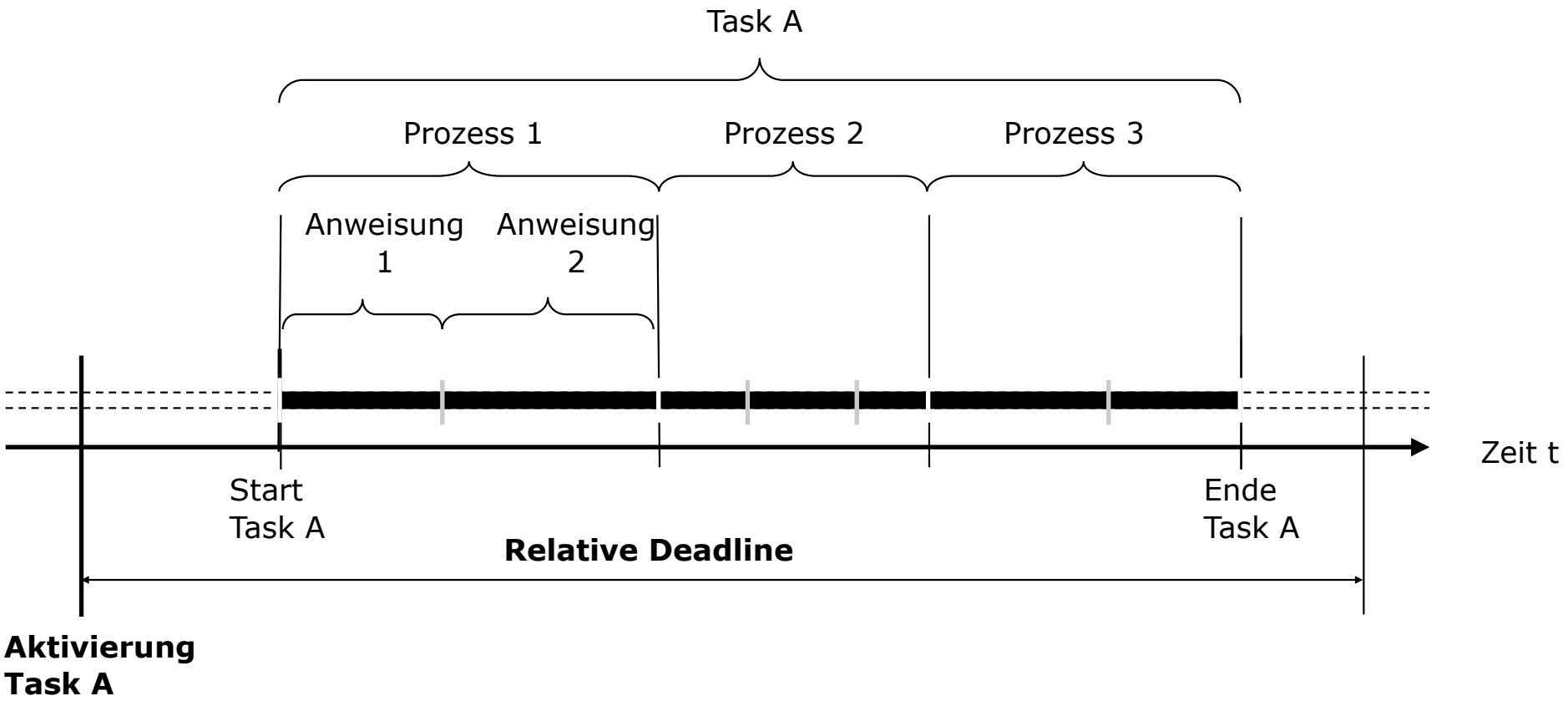
- Spezifikation des Kontrollflusses
  - Klassische Ansätze aus der Programmierung, z. B. Struktogramme
  - Ansätze aus dem objekt-orientierten Entwurf, z. B. für die Aufrufstruktur zwischen Software-Komponenten
- Kontrollfluss zur Festlegung der Ausführungsreihenfolge in ASCET-SD



Spezifikation der Software-Komponenten



Zuordnung von Berechnungen zu Prozessen und Tasks



Definition: Prozesse und Tasks

- Nach Schäuffele/Zurawka
- Task
  - Aufgabeneinheit, die potentiell parallel mit anderen Aufgabeneinheiten auf einem Netzwerk von Prozessoren bearbeitet werden kann
  - Jede Task ist genau einem Prozessor zugeordnet
  - Auf einem Prozessor können mehrere Tasks bearbeitet werden
    - Nacheinander
    - In einzelnen Zeitscheiben
- Prozess
  - Aufgabeneinheiten mit identischen Echtzeitanforderungen
    - Realisierung jeweils als Task
    - Zusammenfassung zu einer Task: Die Aufgabeneinheit wird dann als Prozess bezeichnet. Die Prozesse einer Task werden in einer statisch festgelegten Reihenfolge nacheinander ausgeführt.

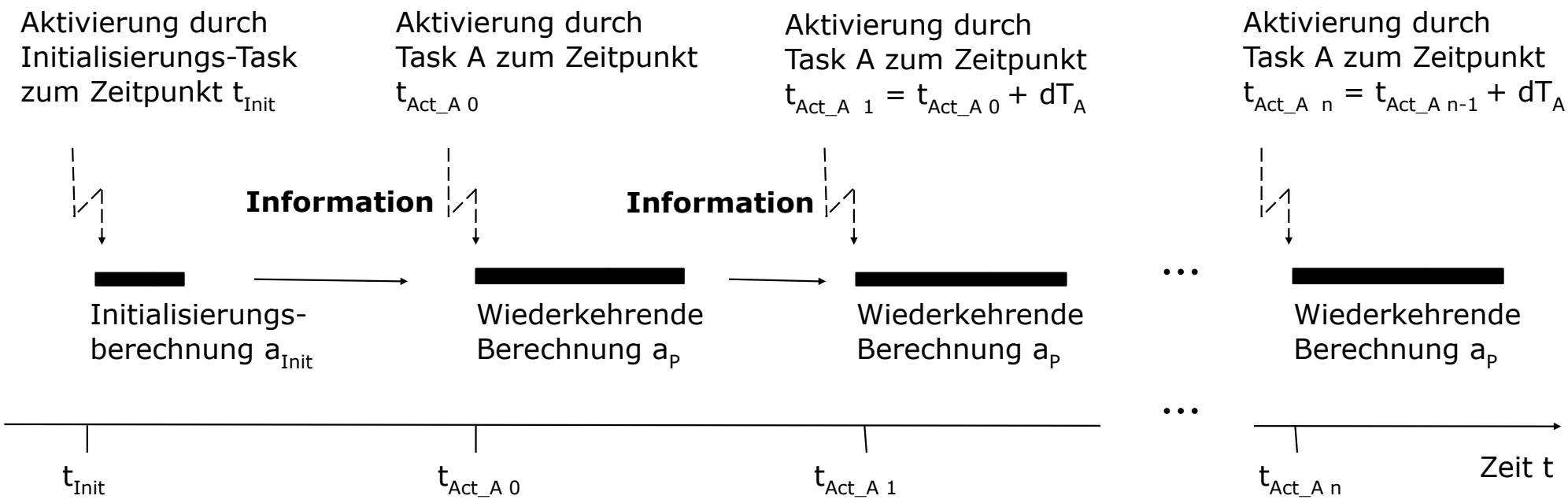
- ACHTUNG: Auch andere Definitionen von Task und Prozess

Ausführungsmodelle

- Zeitabhängige Ausführung („Gedächtnis“):
  Zustandsabhängiges, reaktives Ausführungsmodell
- Ereignisabhängige Ausführung:
  Zustandsunabhängiges, reaktives Ausführungsmodell
- In der Praxis Mischformen

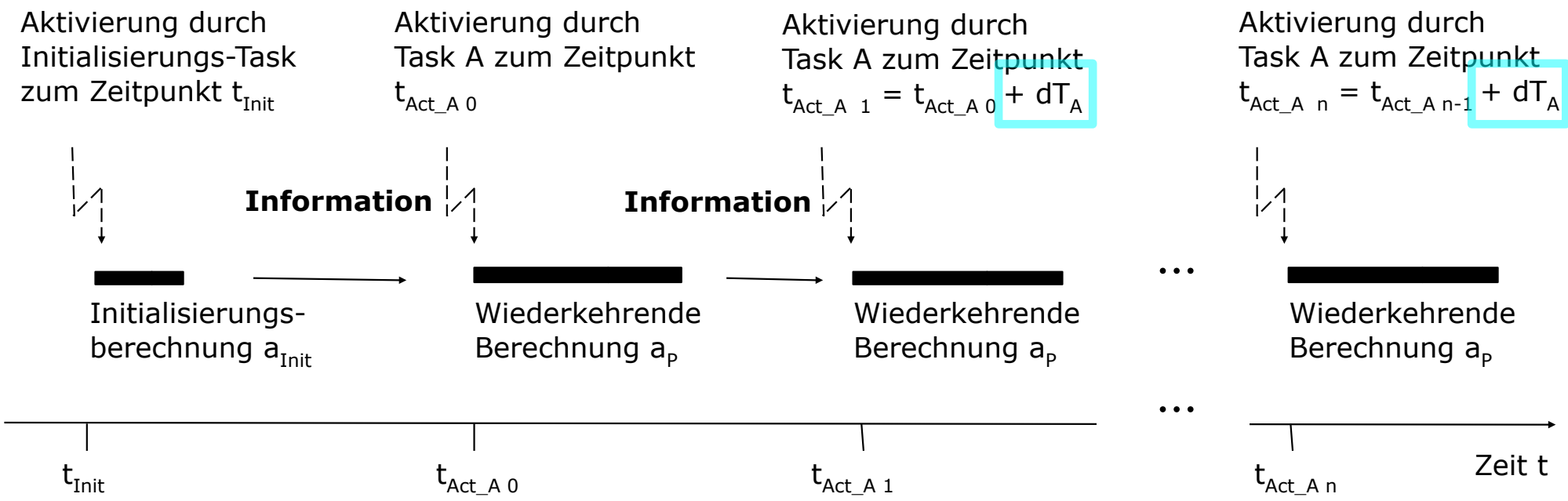
Zeitabhängige Ausführung („Gedächtnis“)

Zustandsabhängiges, reaktives Ausführungsmodell für Software-Funktionen



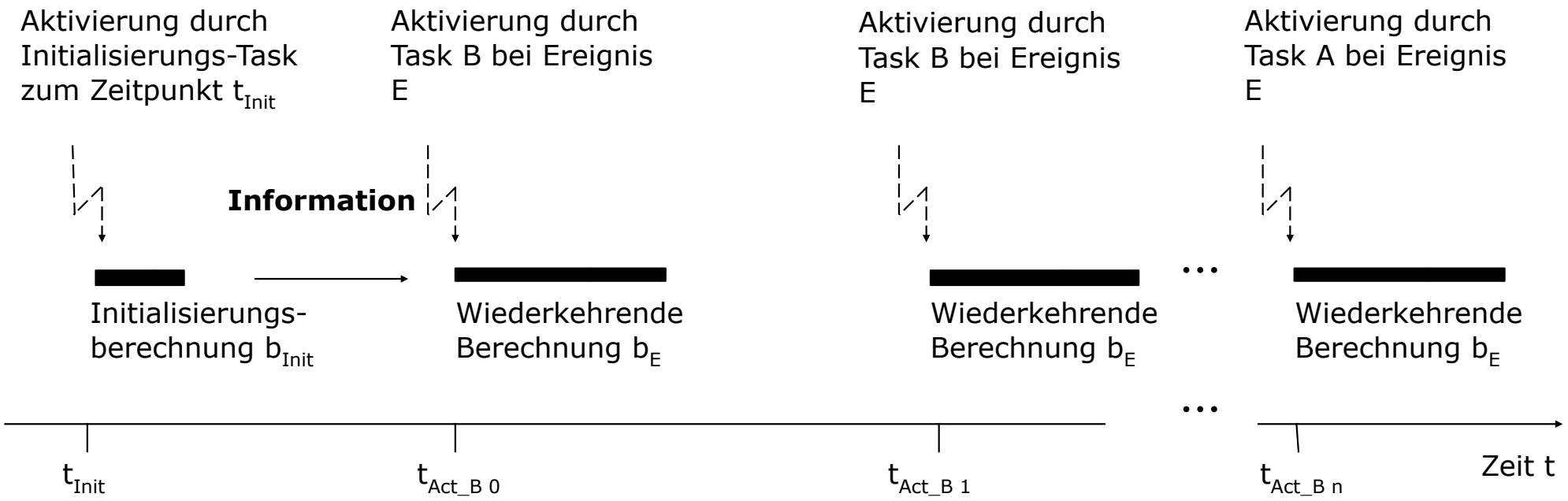
Zeitabhängige Ausführung („Gedächtnis“)

Zustandsabhängiges, reaktives Ausführungsmodell für Software-Funktionen



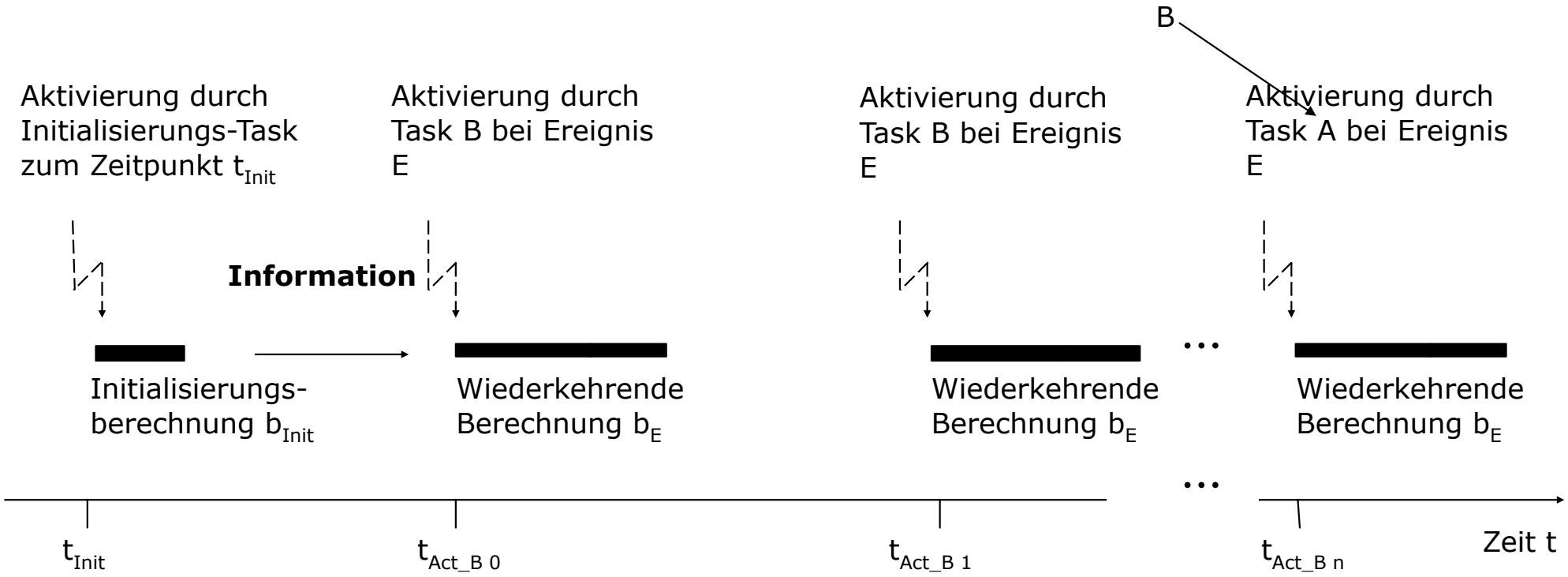
Ereignisabhängige Ausführung

Zustandsunabhängiges, reaktives Ausführungsmodell für Software-Funktionen



Ereignisabhängige Ausführung

Zustandsunabhängiges, reaktives Ausführungsmodell für Software-Funktionen



Ausführungsmodelle

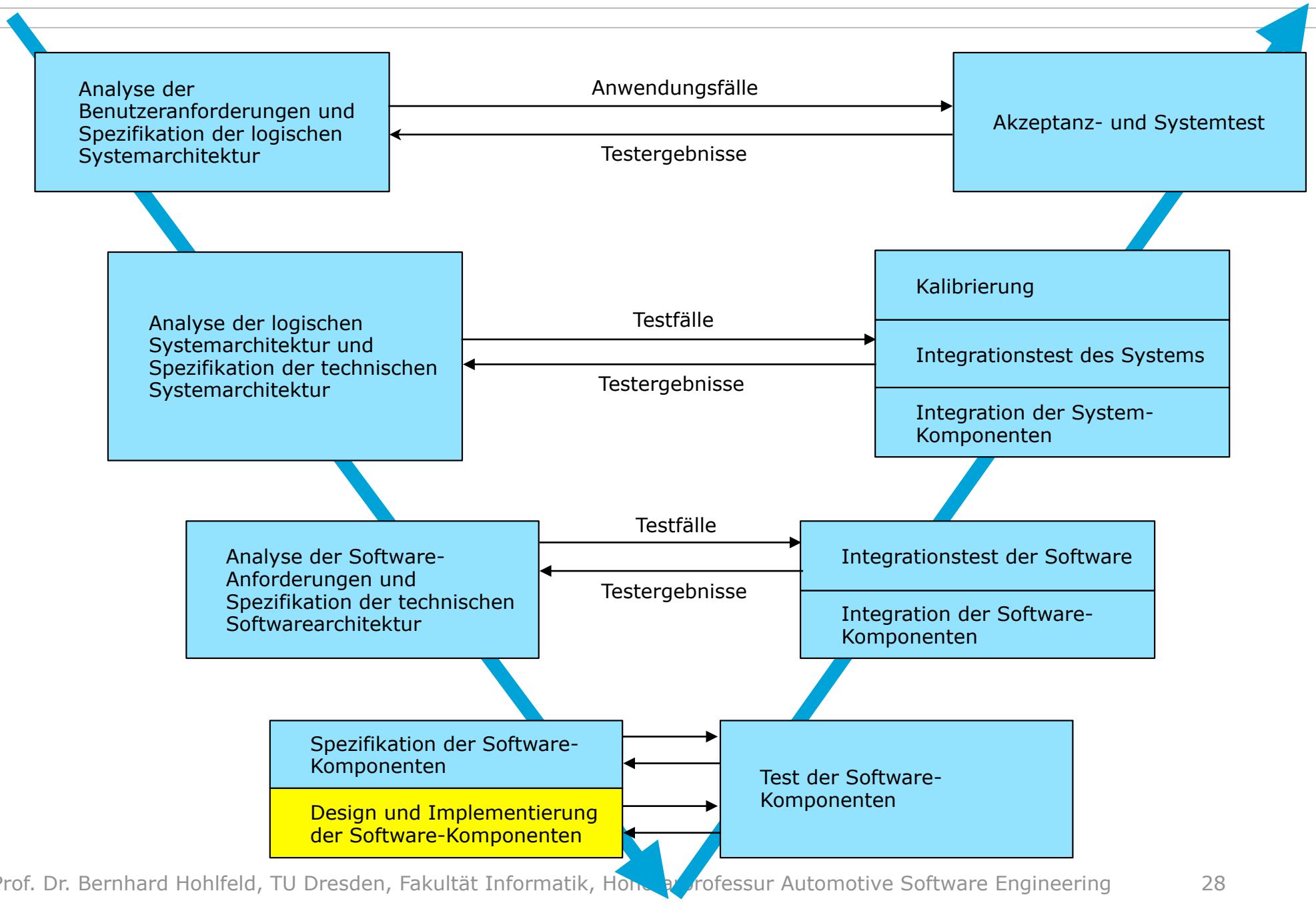
- Zeitabhängige Ausführung („Gedächtnis“):
  Zustandsabhängiges, reaktives Ausführungsmodell
- Ereignisabhängige Ausführung:
  Zustandsunabhängiges, reaktives Ausführungsmodell
- In der Praxis Mischformen

Beispiele

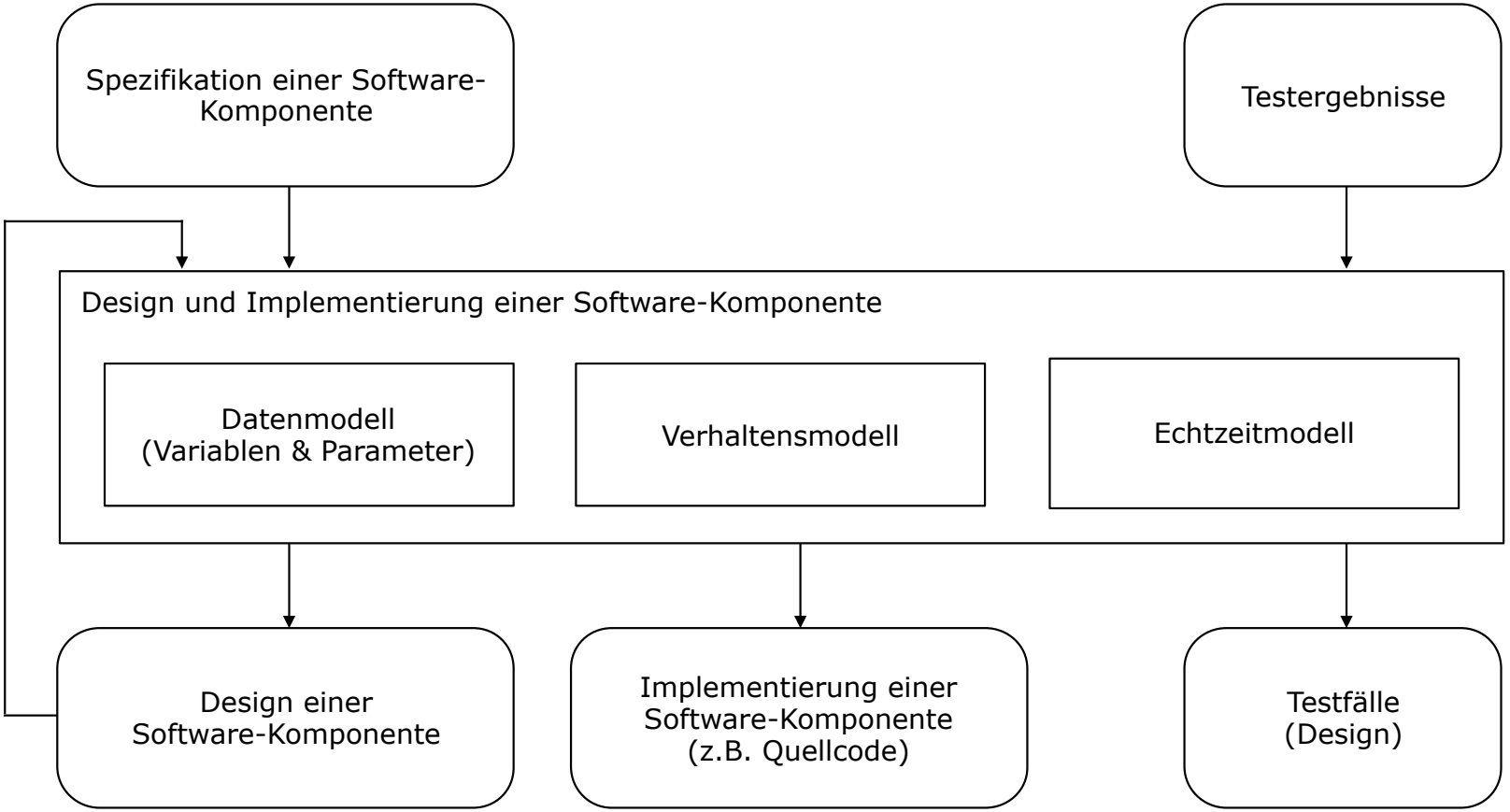
	Zustandsunabhängig	**Zustandsabhängig**
Zeitabhängig ohne Berechnung	Periodisches Auslesen eines Sensors Beispiel: Regensensor	Aktionen bei bestimmten Sensorwerten Beispiel: Wischen wenn Sensor Regen meldet
Zeitabhängig mit Berechnung	Periodische Berechnung eines Wertes Beispiel: Odometer	Darstellung von (neuen) Werten Beispiel: Neuer km-Stand im Kombiinstrument (ganzzahlig)
Ereignisabhängig ohne Berechnung	Benutzeraktionen Beispiel: Hupen	Benutzeraktionen Beispiel: Öffnen/Schliessen der Heckklappe
Ereignisabhängig mit Berechnung	Benutzeraktionen Beispiel: Abfrage Ölstand	Benutzeraktionen Beispiel: Einstellen der Innentemperatur (Soll), Vergleich (Ist), Kühlen/Heizen

6. SW-Entwicklung / 2. Kernprozess

1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. **Design und Implementierung der Software-Komponenten**
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. Kalibrierung
14. Akzeptanz- und Systemtest

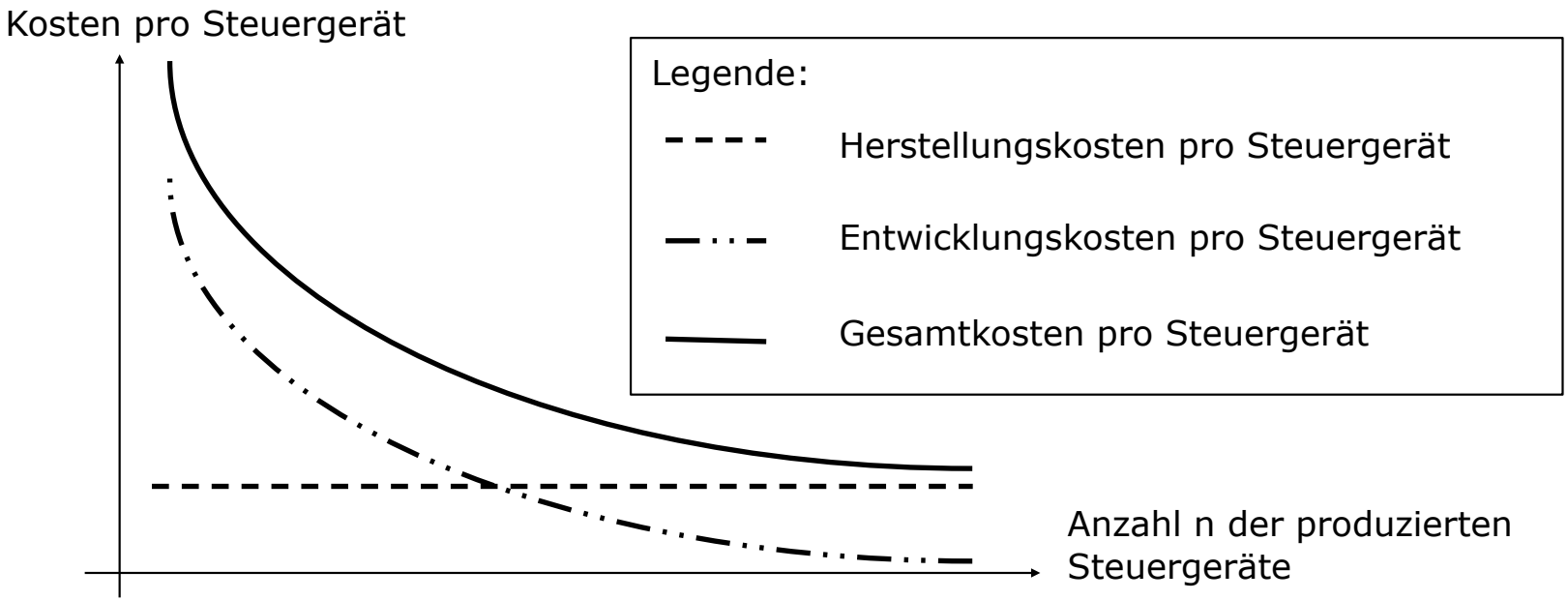

Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

Design und Implementierung



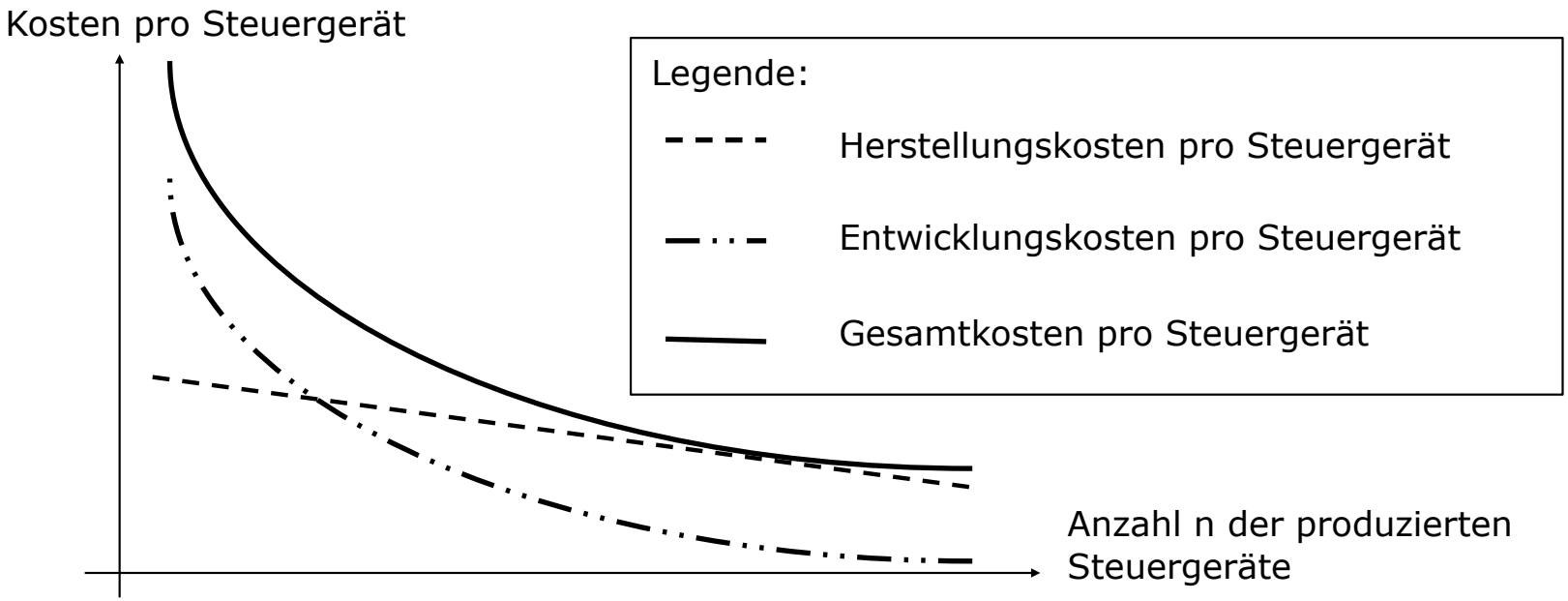
Design und Implementierung

- Berücksichtigung der geforderten nichtfunktionalen Produkteigenschaften
  - Unterschied zwischen Programm- und Datenstand
  - Beschränkung der Hardware-Ressourcen
- Beispiel: Kostenschranken bei Steuergeräten
  - Gesamtkosten pro Steuergerät bei n Steuergeräten
    = Entwicklungskosten / n + Herstellungskosten pro Steuergerät

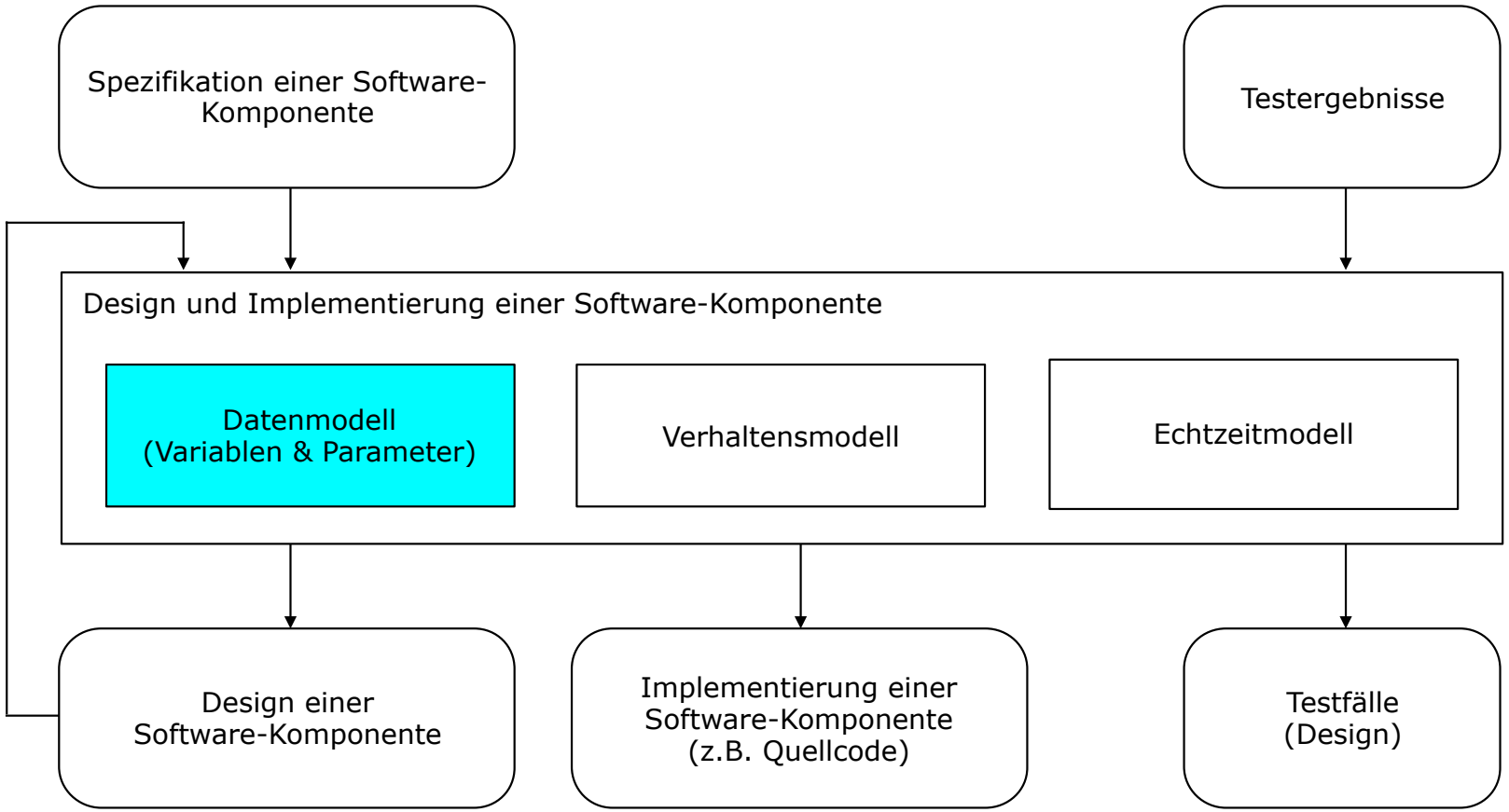


Design und Implementierung

- Berücksichtigung der geforderten nichtfunktionalen Produkteigenschaften
  - Unterschied zwischen Programm- und Datenstand
  - Beschränkung der Hardware-Ressourcen
- Beispiel: Kostenschranken bei Steuergeräten
  - Gesamtkosten pro Steuergerät bei n Steuergeräten
    = Entwicklungskosten / n + Herstellungskosten pro Steuergerät



Design und Implementierung

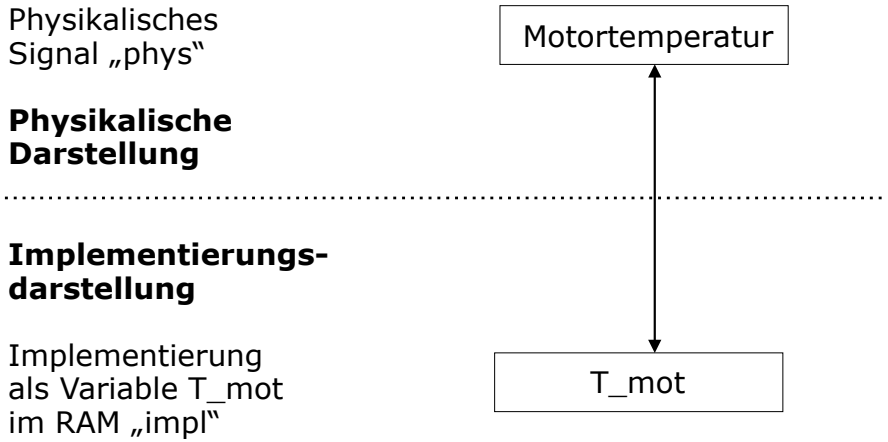


Design und Implementierung

- 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

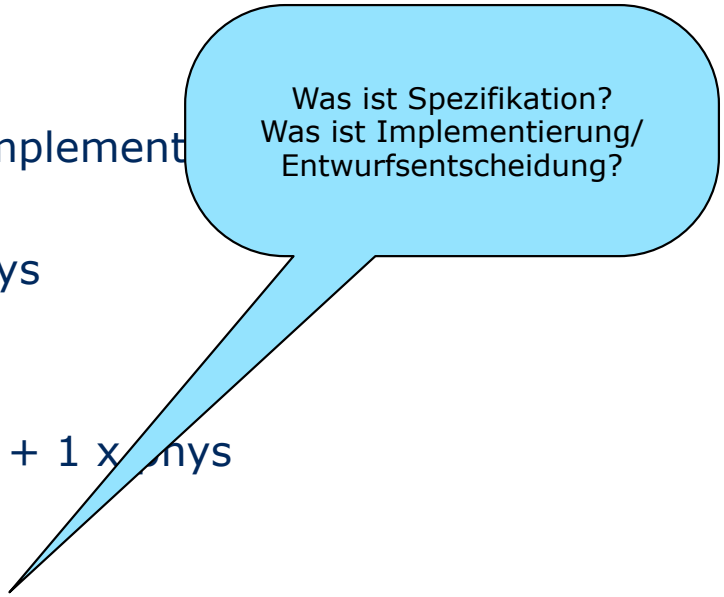


Design und Implementierung

- 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
- Implementierung
  - Bezeichnung im Code: T_mot
  - Wortlänge: 8 Bit
  - Speichersegment: Internes RAM

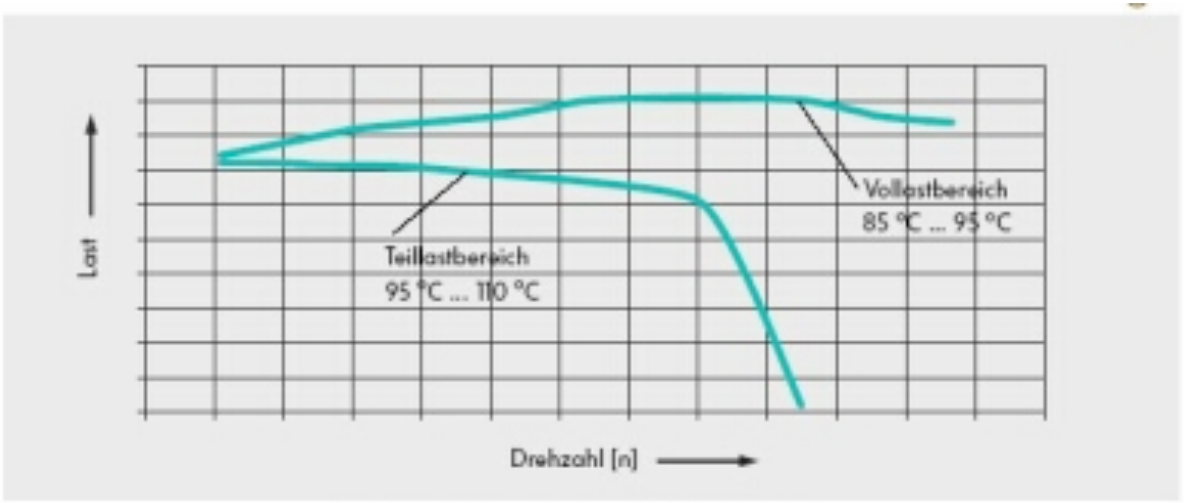
Design und Implementierung

- 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
- Implementierung
  - Bezeichnung im Code: T_mot
  - Wortlänge: 8 Bit
  - Speichersegment: Internes RAM



Motortemperatur und Kühlmitteltemperatur

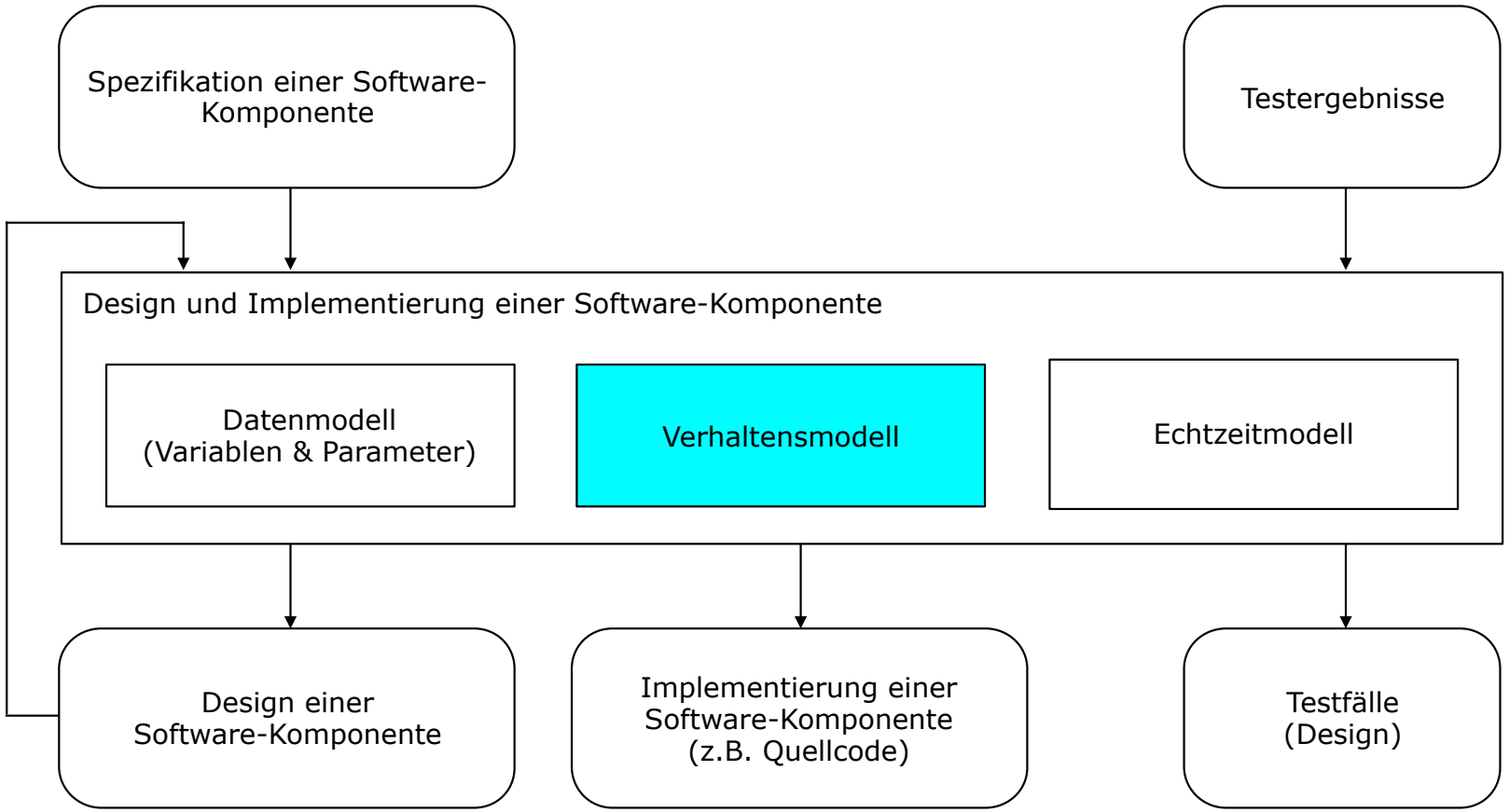
- Quelle: http://www.kfztech.de/kfztechnik/motor/kuehlung/wasserkuehlung.htm
- Die im Verbrennungsmotor erzeugten Temperaturen von über 2000°C bedrohen die nur begrenzt hitzebeständigen Motorbauteile. Die überschüssige Wärme muss deshalb schnell und zuverlässig abgeleitet werden. Bei Volllastbetrieb 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

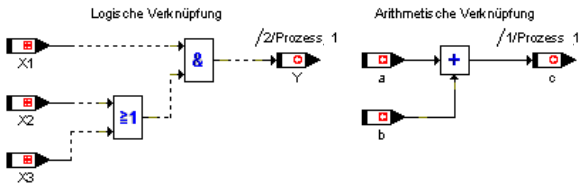
222_013

Design und Implementierung



Spezifikation des Verhaltensmodells

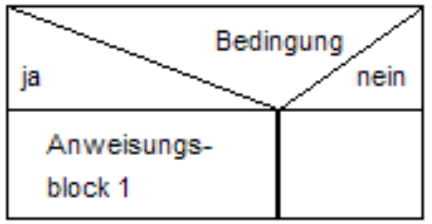
- Datenfluss
  - Änderung der Werte von Variablen
  - Blockdiagramm



- Programmiersprache
  - Wertzuweisung
    - Y := X1 & (X2 || X3)
    - c := a + b

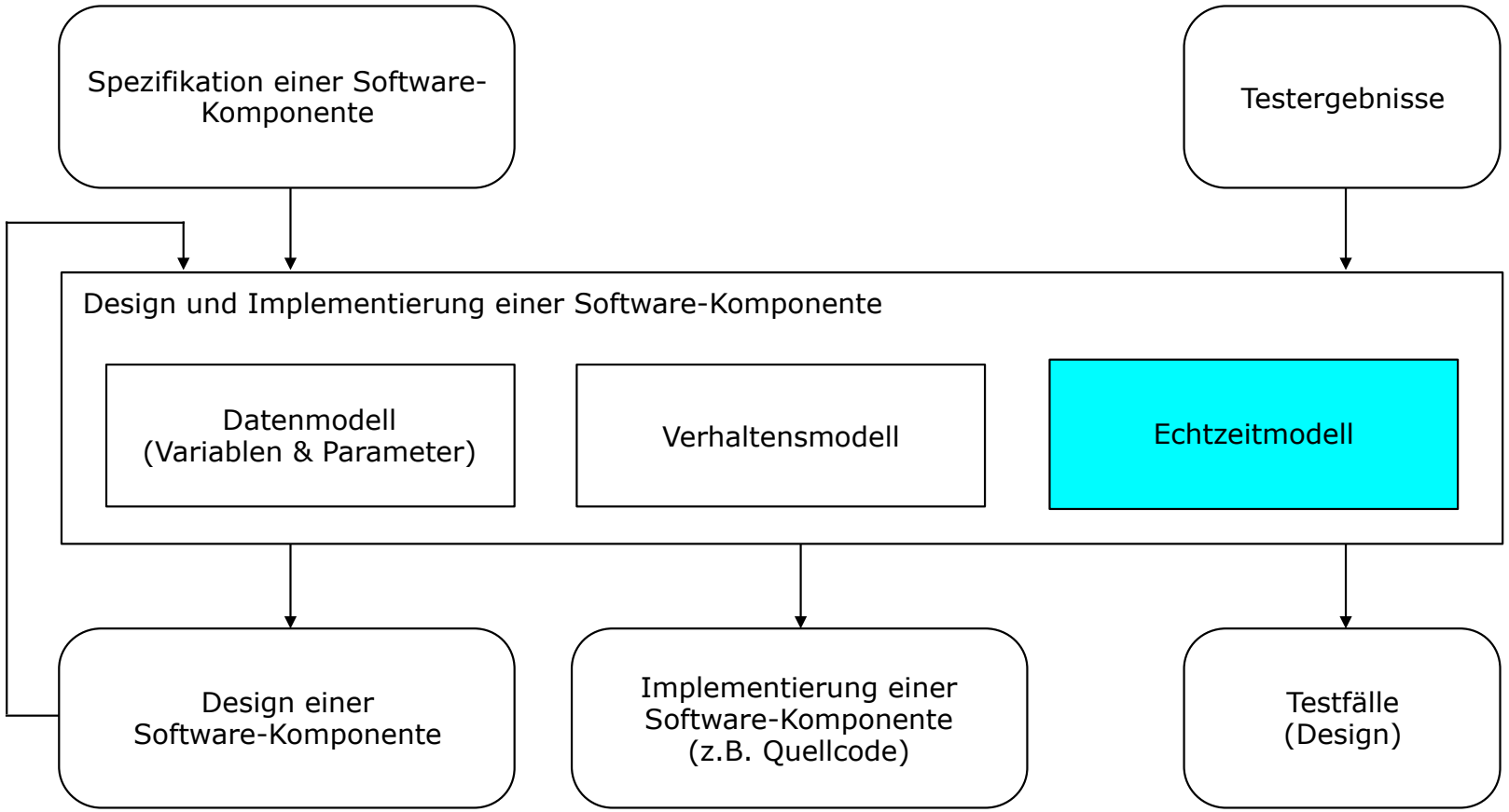
- Kontrollfluss
  - Sequenz
  - Selektion / Verzweigung
  - Iteration

  - Struktogramme



- Programmiersprache
  - if <Bedingung>
    then
    Anweisungsblock 1

Design und Implementierung

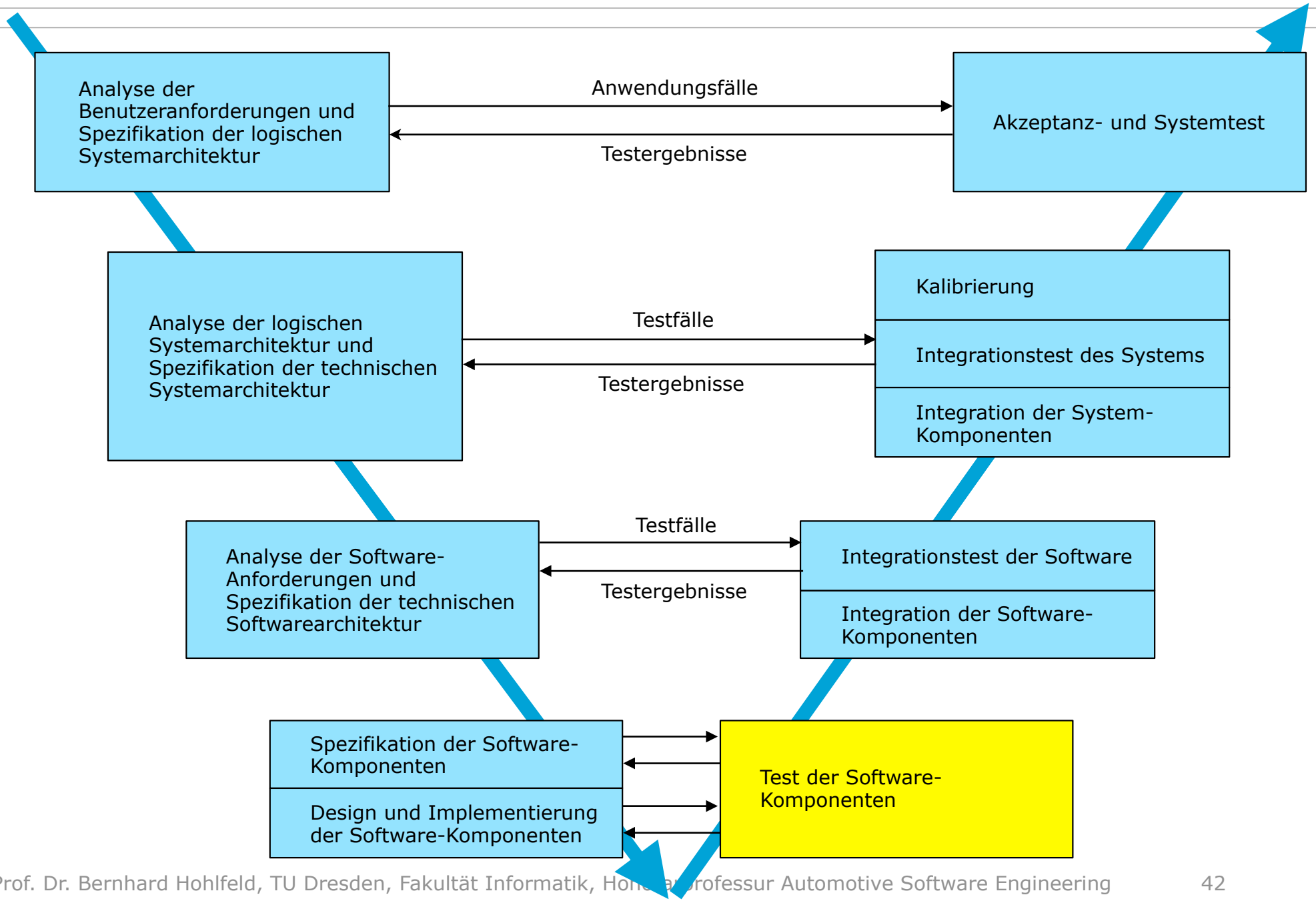


Spezifikation des Echtzeitmodells

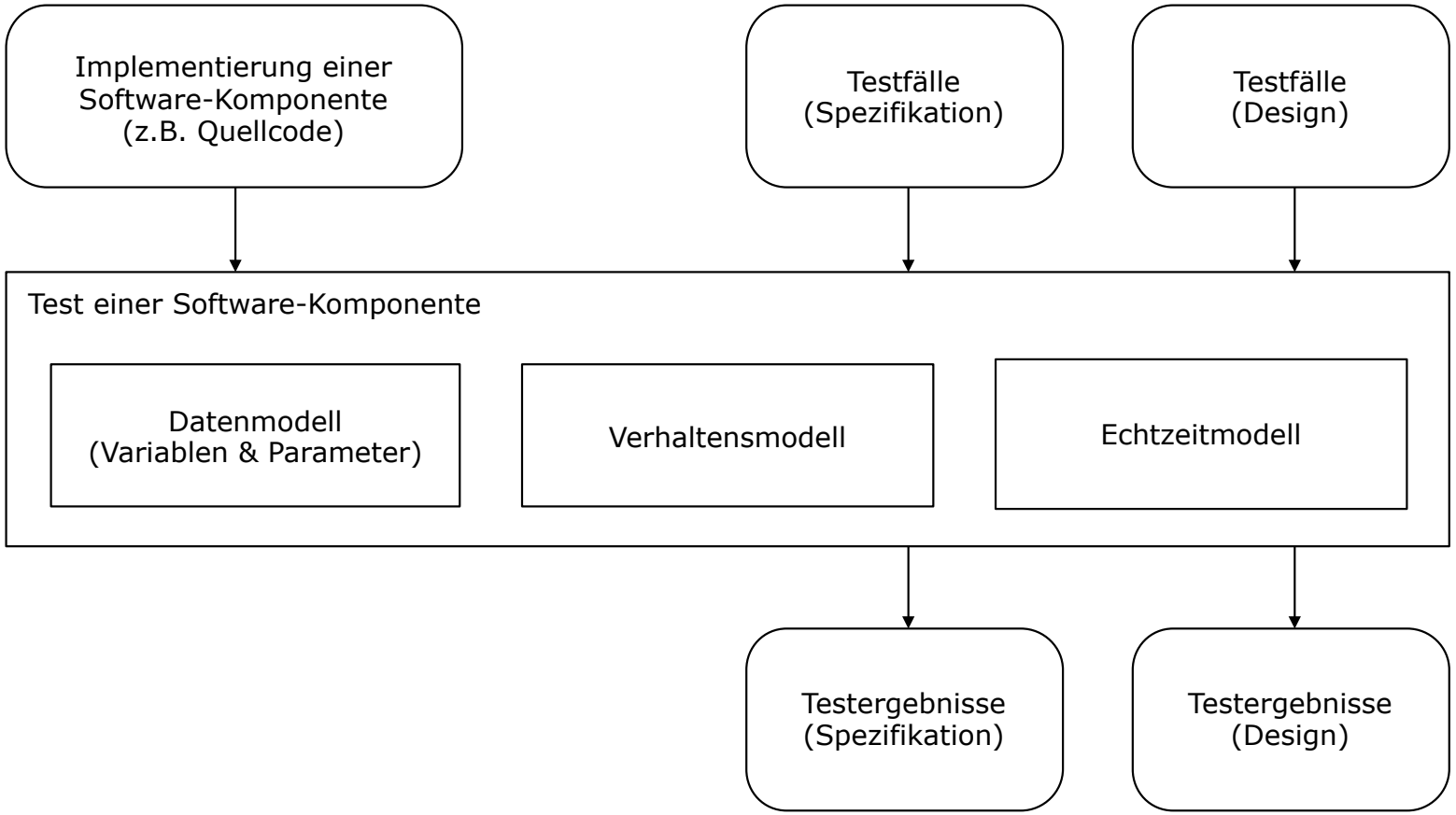
- Zuordnung der Anweisungen der Software-Komponente zu Prozessen
- Zuordnung der Prozesse zu Tasks
- Festlegung der Echtzeitanforderungen für die Tasks

6. SW-Entwicklung / 2. Kernprozess

1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. **Test der Software-Komponenten**
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. Kalibrierung
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

Test der Software-Komponenten



Maßnahmen zur Qualitätssicherung von Software

- Verifikation
  - Statische Techniken
    - Review
      Walkthrough, Fagan-Inspektion, Code-Inspektion, Peer-Review, …
    - Analyse
      Statische Analyse, Formale Prüfung, Kontroll- und Datenfluss, …
  - Dynamischer Test
    Komponenten-/Integrationstest
    - Black-Box-Test
      Funktionale Leistungsfähigkeit, Stress, Grenzwert, Fehlererwartung, ...
    - White-Box-Test
      Struktur, Pfad, Zweig, Bedingung, Abdeckung, …
- Validation
  - Animation, Formale Spezifikation, Modellierung, Simulation, Rapid Prototyping, …
  - Systemtest/Akzeptanztest
    Funktionale Leistungsfähigkeit, Stresstests, Grenzwerttests, Fehlererwartungstests, Ursache-Wirkungs-Graph, Äquivalenz-klassentests, …

- siehe auch http://campar.in.tum.de/twiki/pub/Chair/TeachingWs04MedInform/Softwaresicherheit.pdf

Verifikation/verifizierung und Validation/Validierung

Verifikation:

- „Does the system things right?“ / „Erfüllt das System seine Aufgabe richtig?“
- Prüfung z.B. gegen Technische Anforderungen
- Beispiel: Analyse des Zeitverhaltens durch Untersuchung der Worst Case Execution Time (WCET)

Validation:

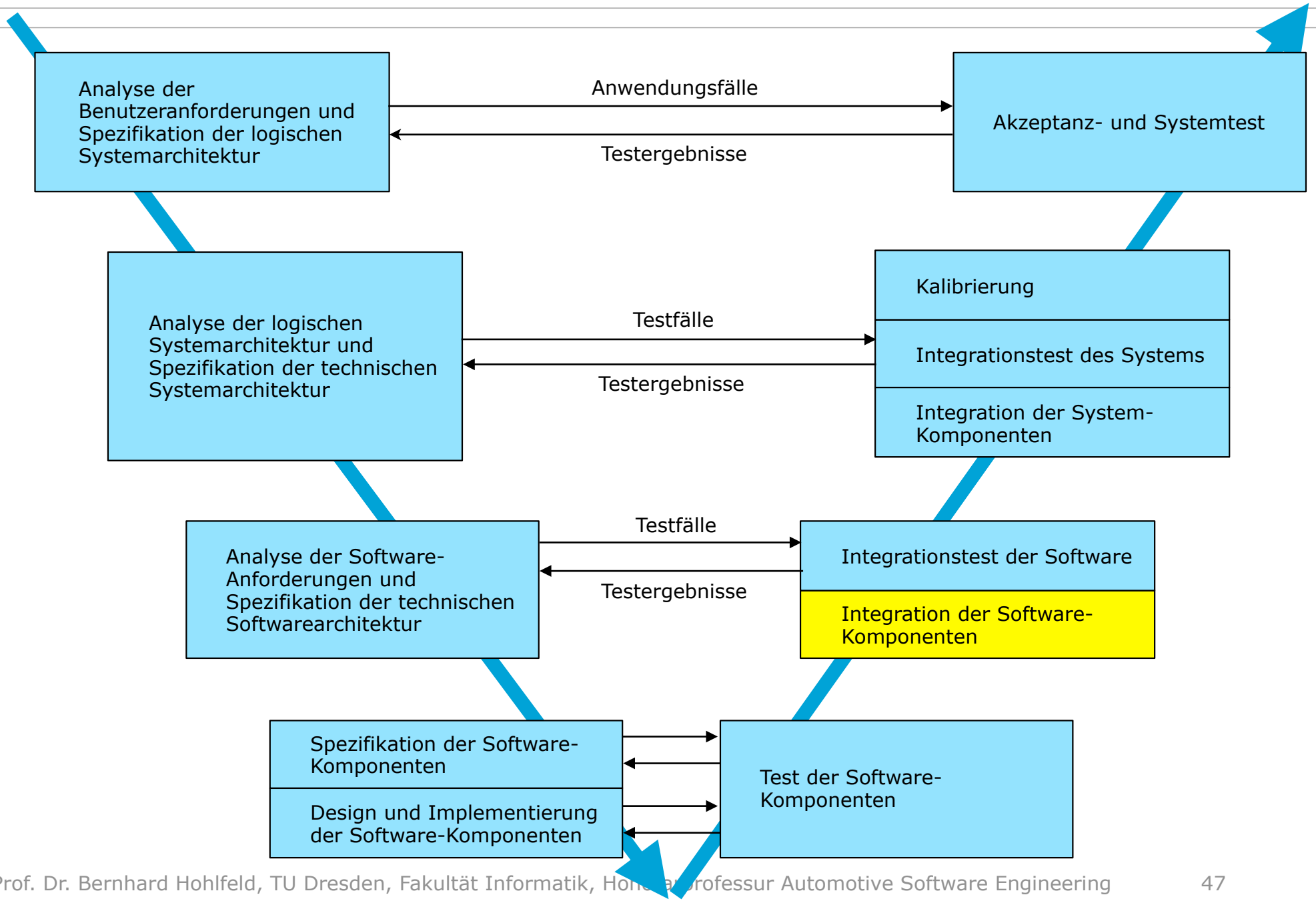
- „Does the system the right things?“ / „Erfüllt das System die richtige Aufgabe?“
- Prüfung z.B. gegen Benutzeranforderungen
- Beispiel: Simulation der Benutzeroberfläche

Ähnlich:

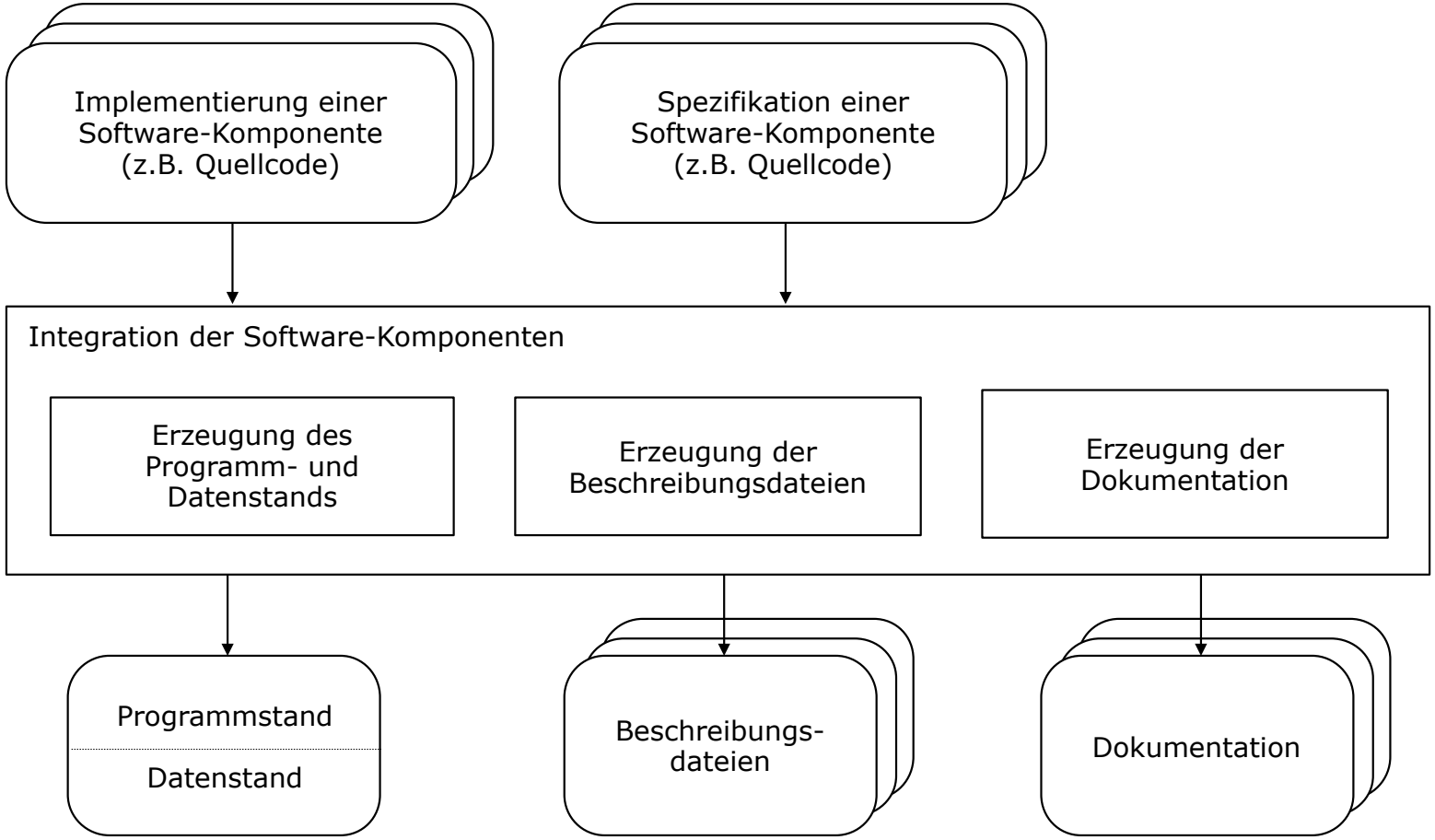
- Effektiv: Die richtigen Dinge tun.
- Effizient: Die Dinge richtig tun.

6. SW-Entwicklung / 2. Kernprozess

1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. **Integration der Software-Komponenten**
10. Integrationstest der Software
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. Kalibrierung
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

Integration der Software-Komponenten



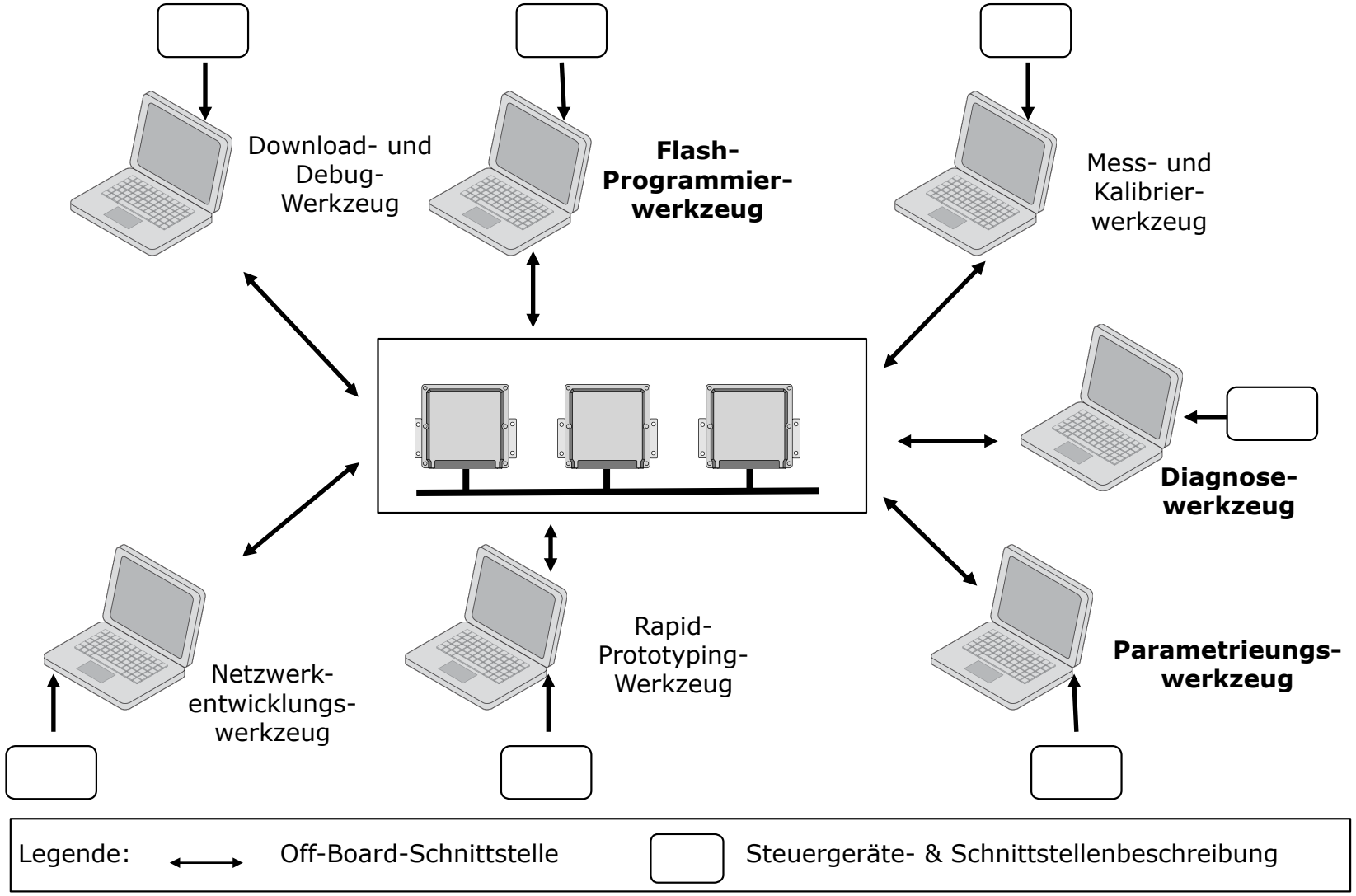
Integration der Software-Komponenten

- siehe auch 5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur

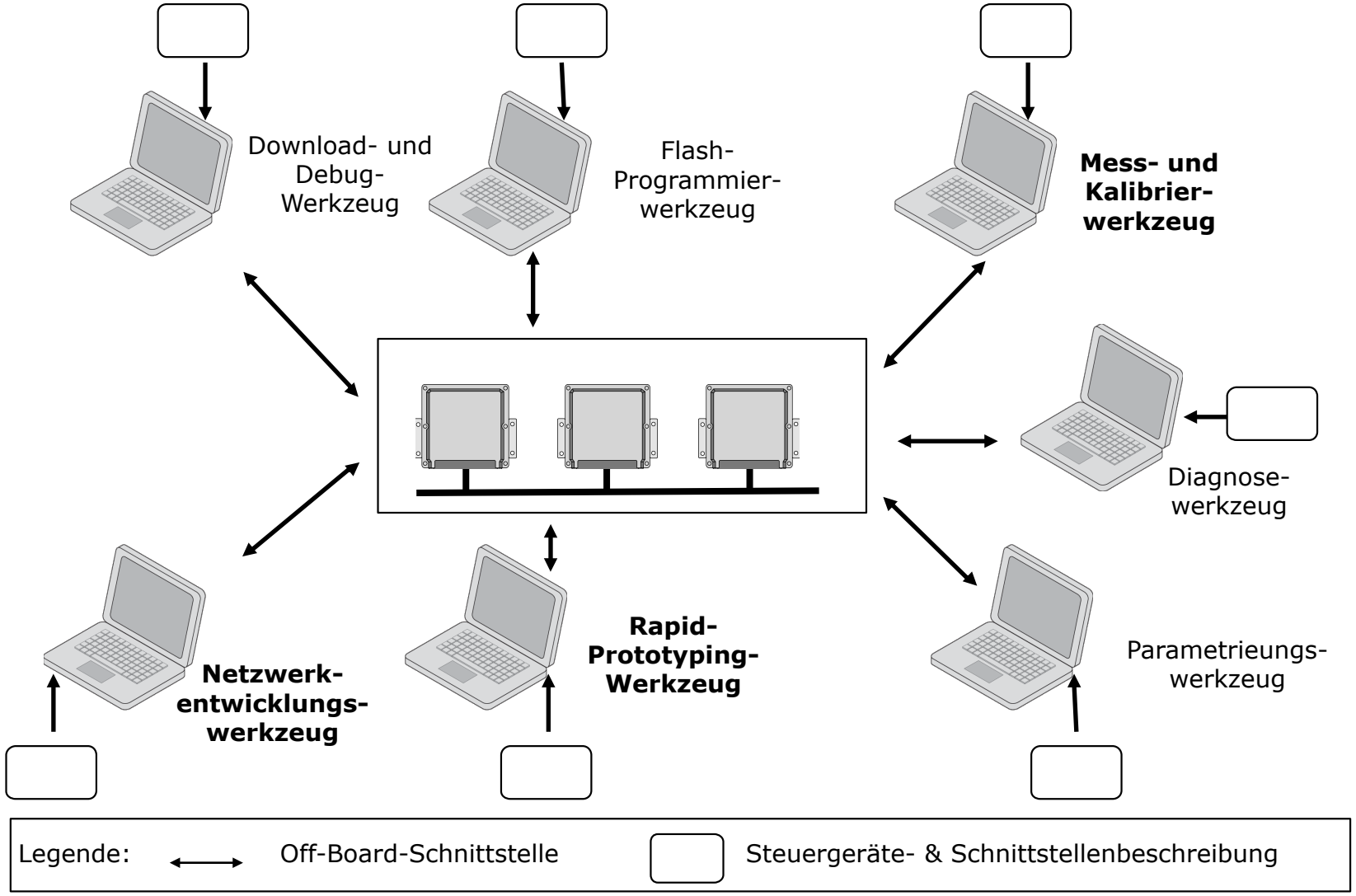
- Software-Stand für ein Seriensteuergerät
  - Programm- und Datenstand für alle Mikrocontroller des Steuergerätes
  - Dokumentation
  - Beschreibungsdateien für Produktions- und Servicewerkzeuge
    - Diagnose-Werkzeuge
    - Software-Parametrisierungs-Werkzeuge
    - Flash-Programmier-Werkzeuge

- Beschreibungsdateien für Entwicklungssteuergeräte
  - Beschreibungsdateien für Mess- und Kalibrierungswerkzeuge
  - Beschreibungsdateien für Werkzeuge zur Netzwerkentwicklung (On-Board-Kommunikation)
  - Beschreibungsdateien für Bypass-Schnittstellen (Rapid -Prototyping)

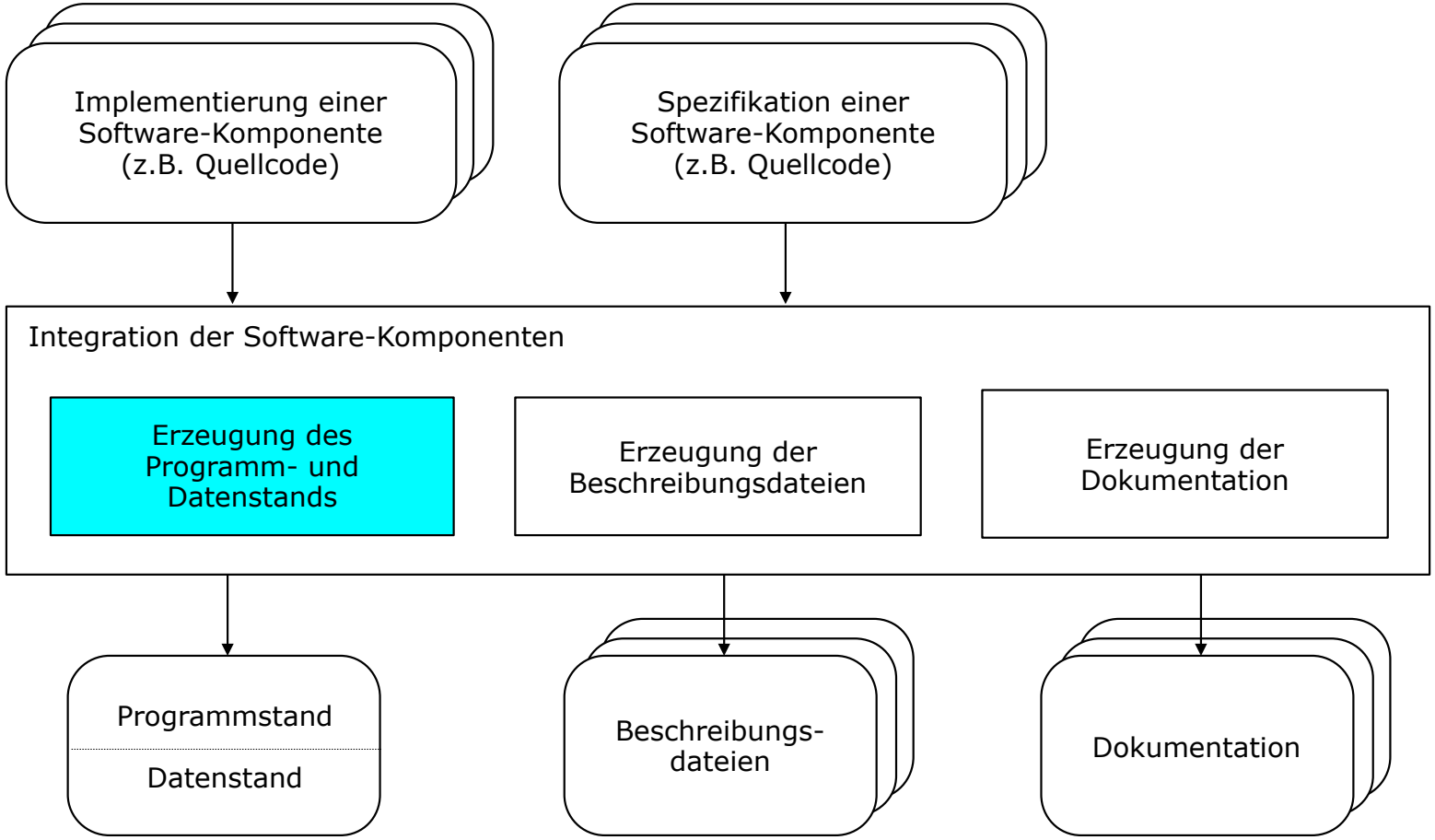
Seriensteuergerät



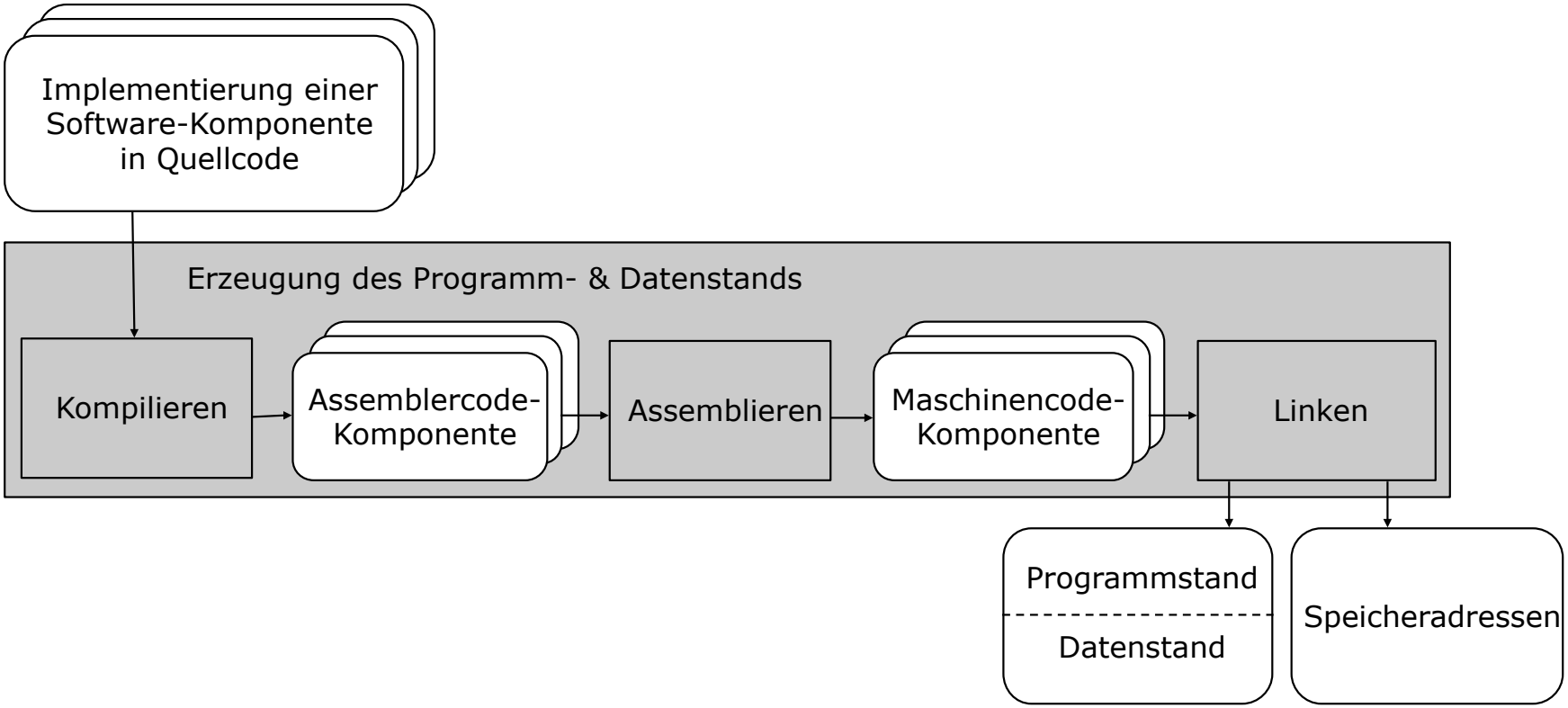
Entwicklungssteuergerät



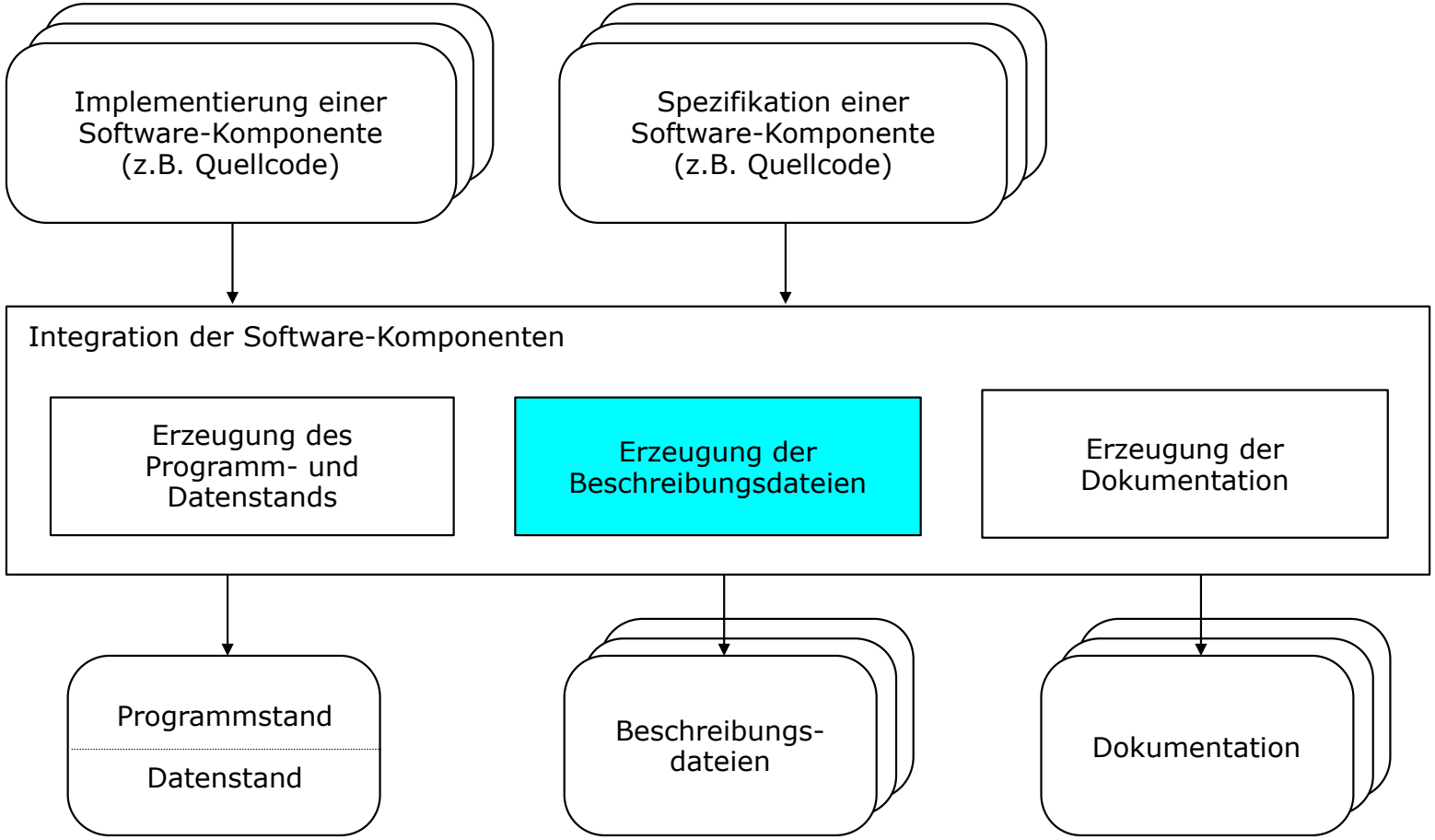
Integration der Software-Komponenten



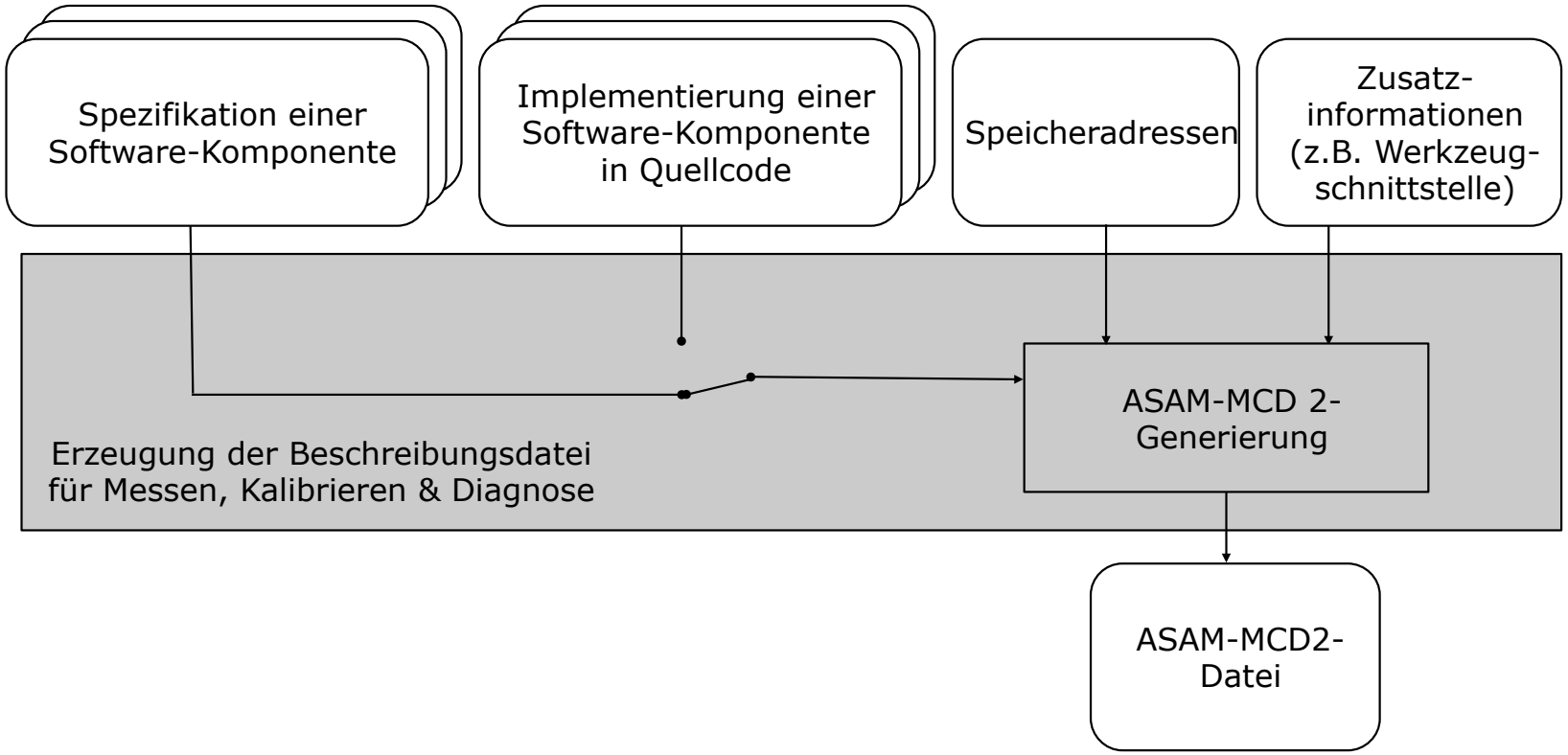
Erzeugung des Programm- und Datenstands



Integration der Software-Komponenten

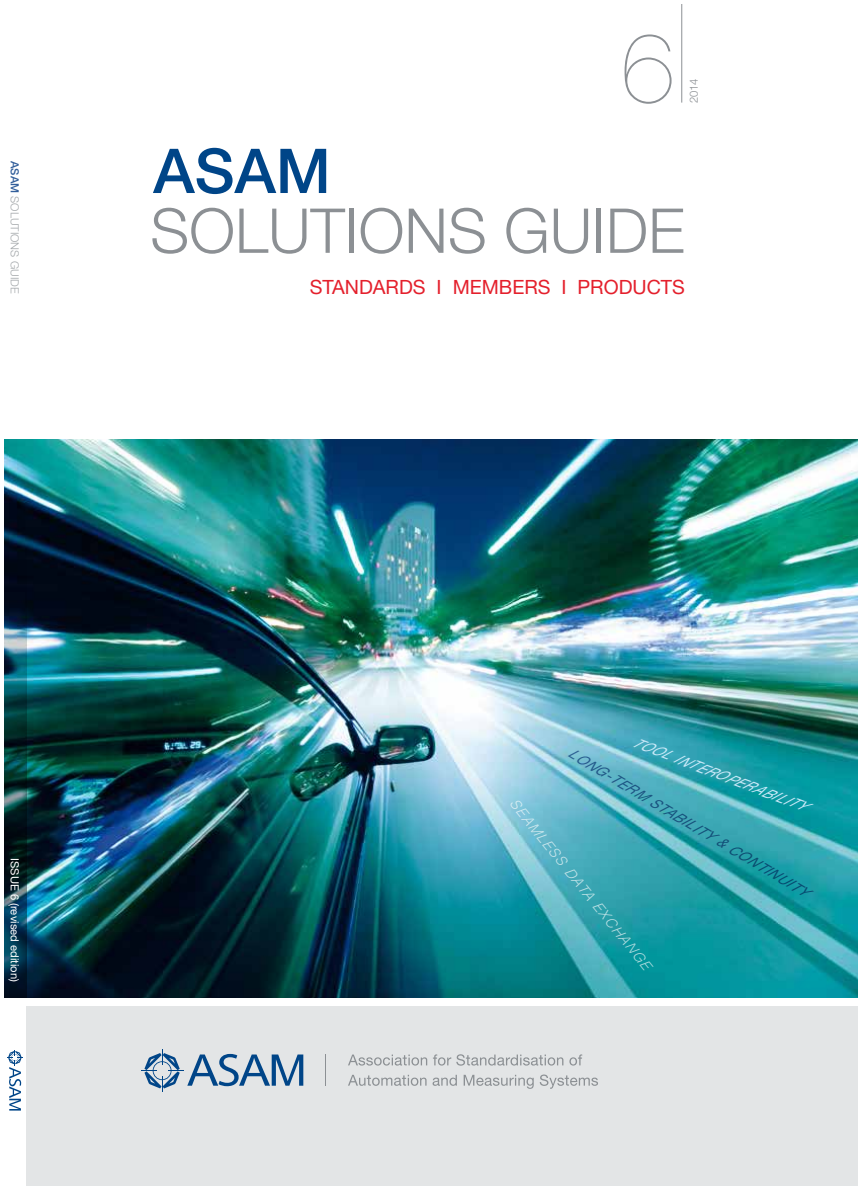


Erzeugung der Beschreibungsdateien



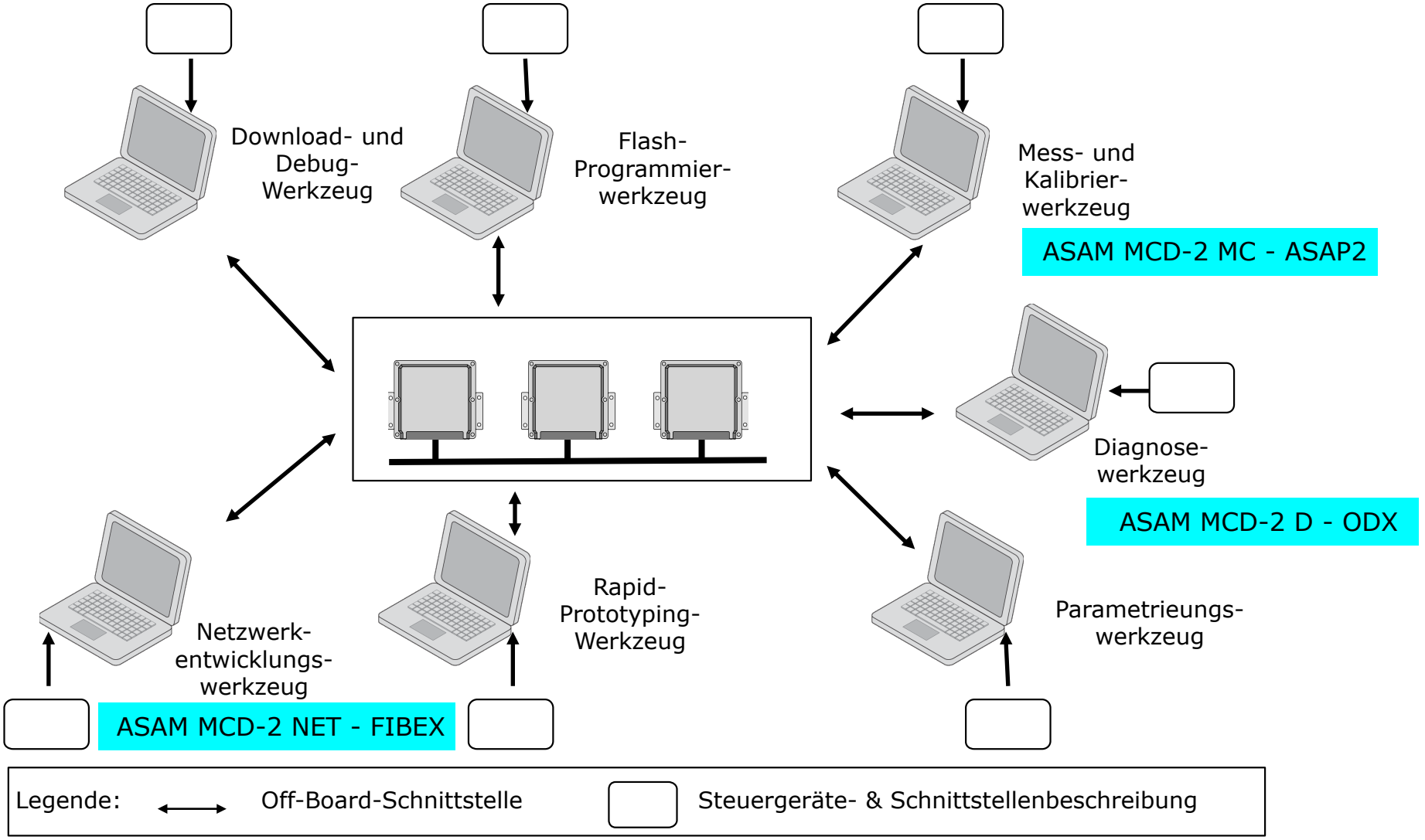
ASAM

- ASAM - Association for Standardisation of Automation and Measuring Systems
- http://www.asam.net

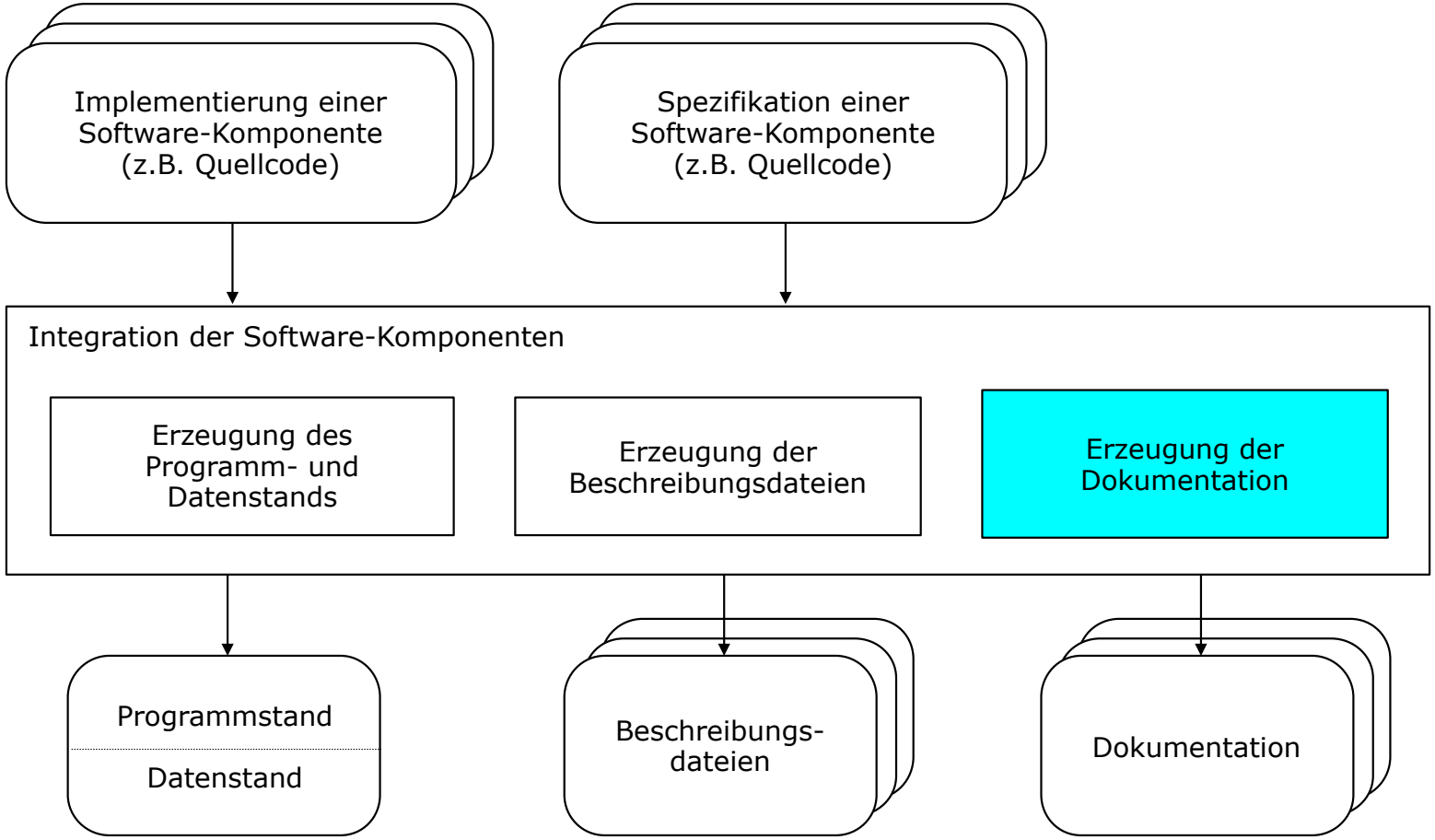


ASAM

- ASAM MCD-2
  The MCD-2 standard ‘family’ defines different symbolic description standards for ECU access or Network data. The standards summarized in the MCD-2 standard family are known in the market as ODX, FIBEX and ASAP2.
- ASAM MCD-2 D (market name: ODX) - Diagnose
  Data Model for ECU Diagnostics (also: Open Diagnostic Data Exchange Format)
  ODX defines a unique and open XML exchange format for transferring ECU diagnostic and programming data between system supplier, vehicle manufacturer and service dealerships and diagnostic tools of different vendors.
- ASAM MCD-2 NET (market name: FIBEX) - Netzwerk
  Data Model for ECU Network Systems (also: Field Bus Data Exchange Format)
  FIBEX describes an XML exchange format for data exchange between tools that deal with bus communication systems (tools for bus configuration, parameterization, design, monitoring, simulation, …).
- ASAM MCD-2 MC (market name: ASAP2) - Messen und Kalibrieren
  ECU Measurement and Calibration Data Exchange Format
  ASAP2 is a description format for calibration values and signals of an Electronic Control Unit (ECU), its communication methods and interfaces. In contrast to ODX which is service-based, ASAP2 contains address-based information of the ECU code.


Download- und Debug-Werkzeug
Flash-Programmier-werkzeug
Mess- und Kalibrier-werkzeug
ASAM MCD-2 MC - ASAP2
Diagnose-werkzeug
ASAM MCD-2 D - ODX
Netzwerk-entwicklungs-werkzeug
Rapid-Prototyping-Werkzeug
Parametrieungs-werkzeug
ASAM MCD-2 NET - FIBEX
Legende:
Off-Board-Schnittstelle
Steuergeräte- & Schnittstellenbeschreibung

Integration der Software-Komponenten

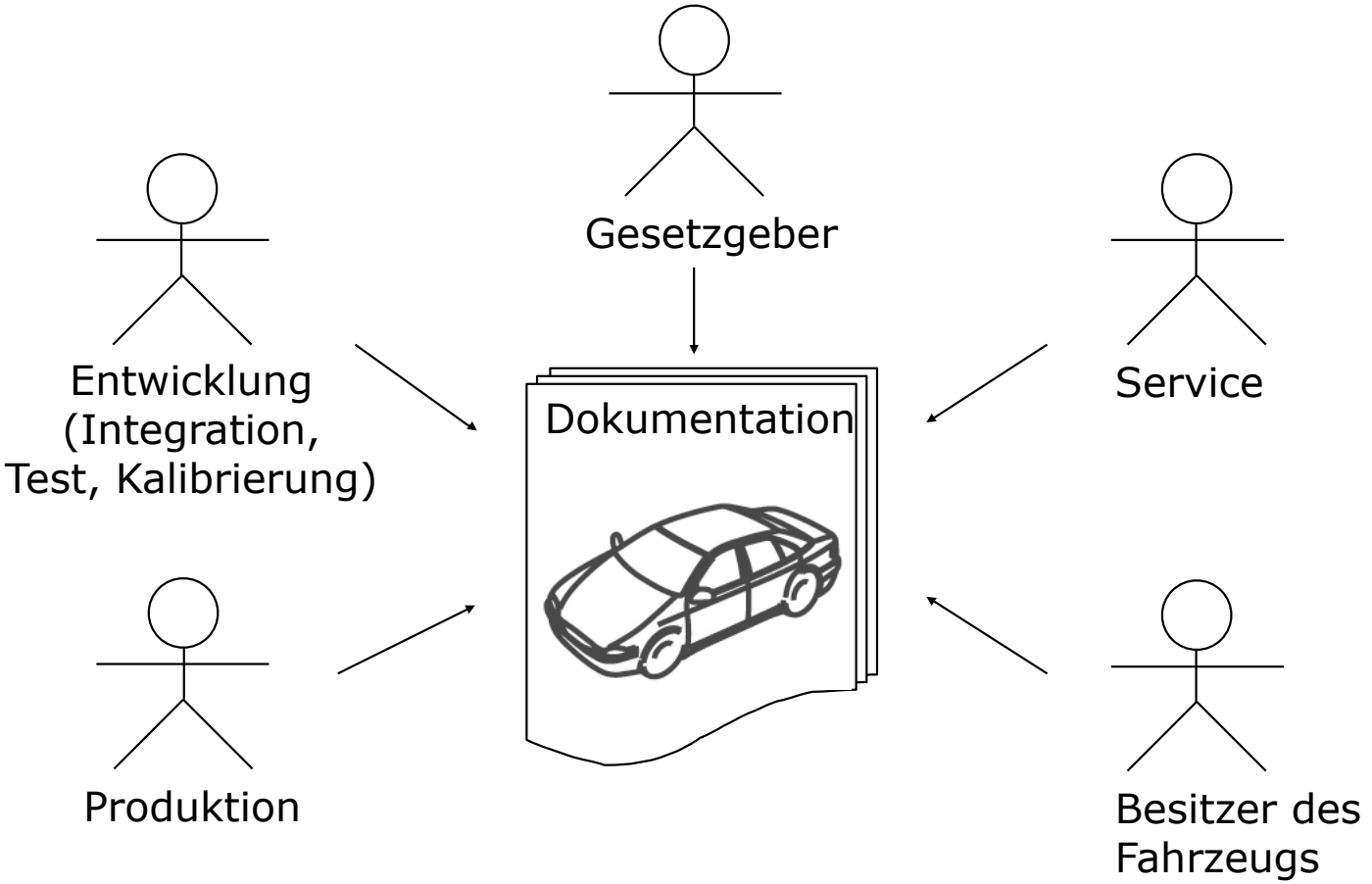


Dokumentation

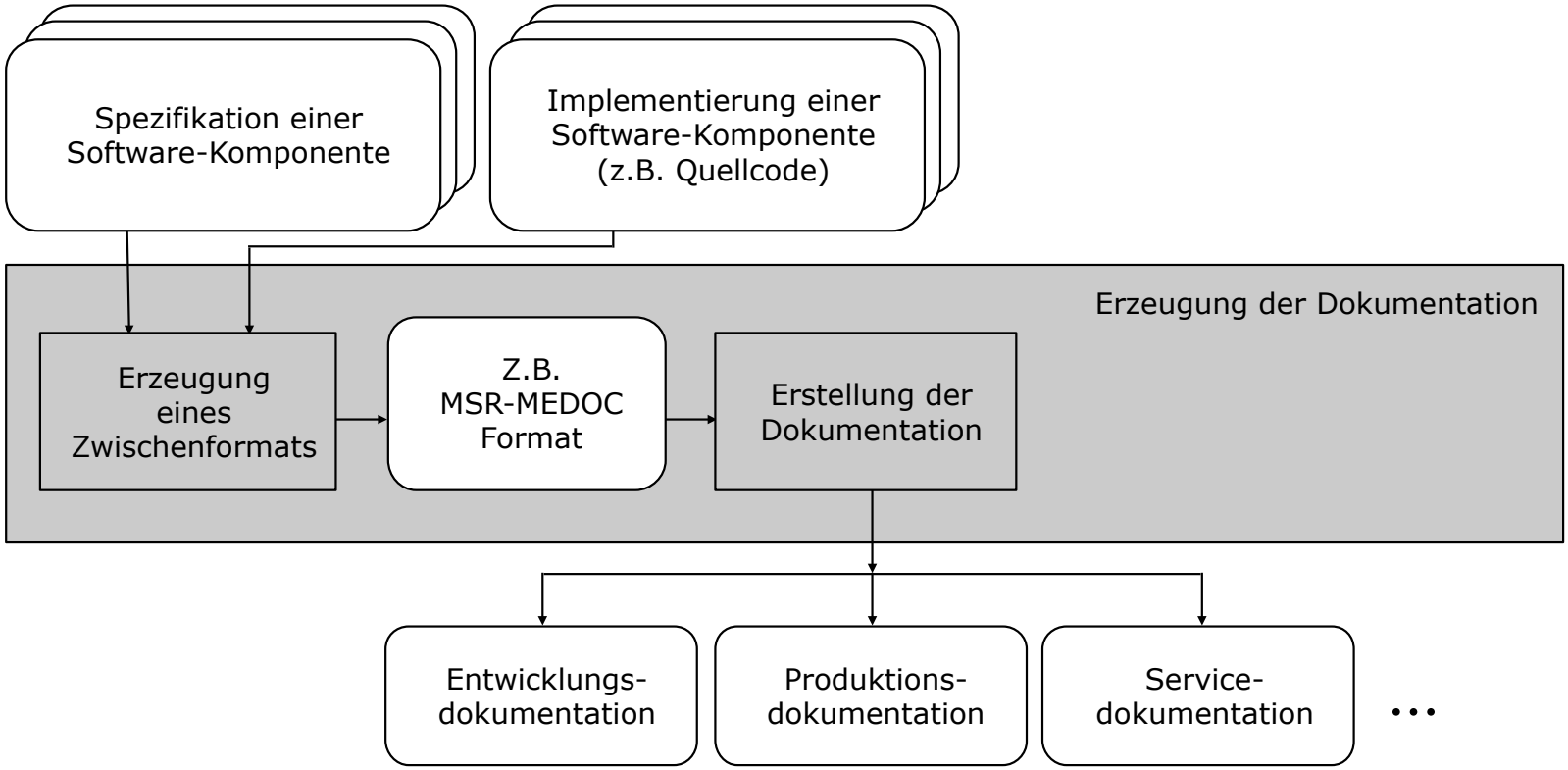
- Die Dokumentation stellt ein Artefakt dar, das für alle Unterstützungsprozesse in der Software-Entwicklung benötigt wird. Auch die ausgeprägte und meist firmenübergreifende Arbeitsteilung, die langen Produktlebenszyklen sowie die damit verbundene lange Wartungsphase für die Software erfordern eine ausführliche (und standardisierte) Dokumentation.
- Alle folgenden Entwicklungsschritte - wie Integration, Test und Kalibrierung des Systems erfordern eine Dokumentation.
- Die Dokumentation wird für Produktion und Service (weltweit) benötigt.
- Der Gesetzgeber fordert Dokumentation, z.B. bei der Beantragung der Zulassung eines Fahrzeugs zum Strassenverkehr.
- Normen fordern Dokumentation
- Beispiel ISO 26262 Road Vehicles - Functional Safety

- siehe auch
  6. SW-Entwicklung
     3. Unterstützungsprozesse

Benutzergruppen der Software-Dokumentation

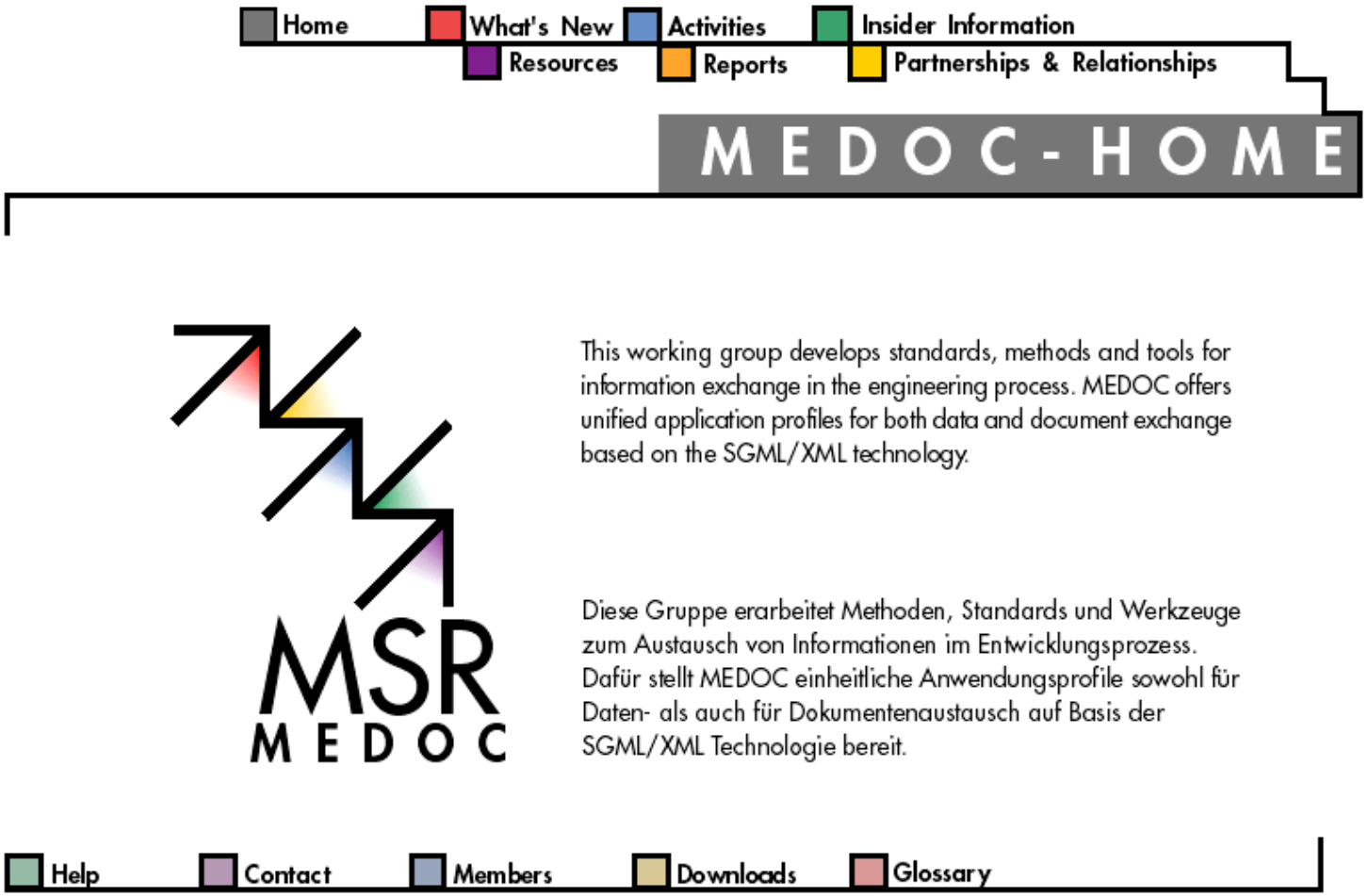


Erzeugung der Dokumentation



MEDOC
MSR Engineering Data Objects and Contents

http://www.msr-wg.de/medoc/



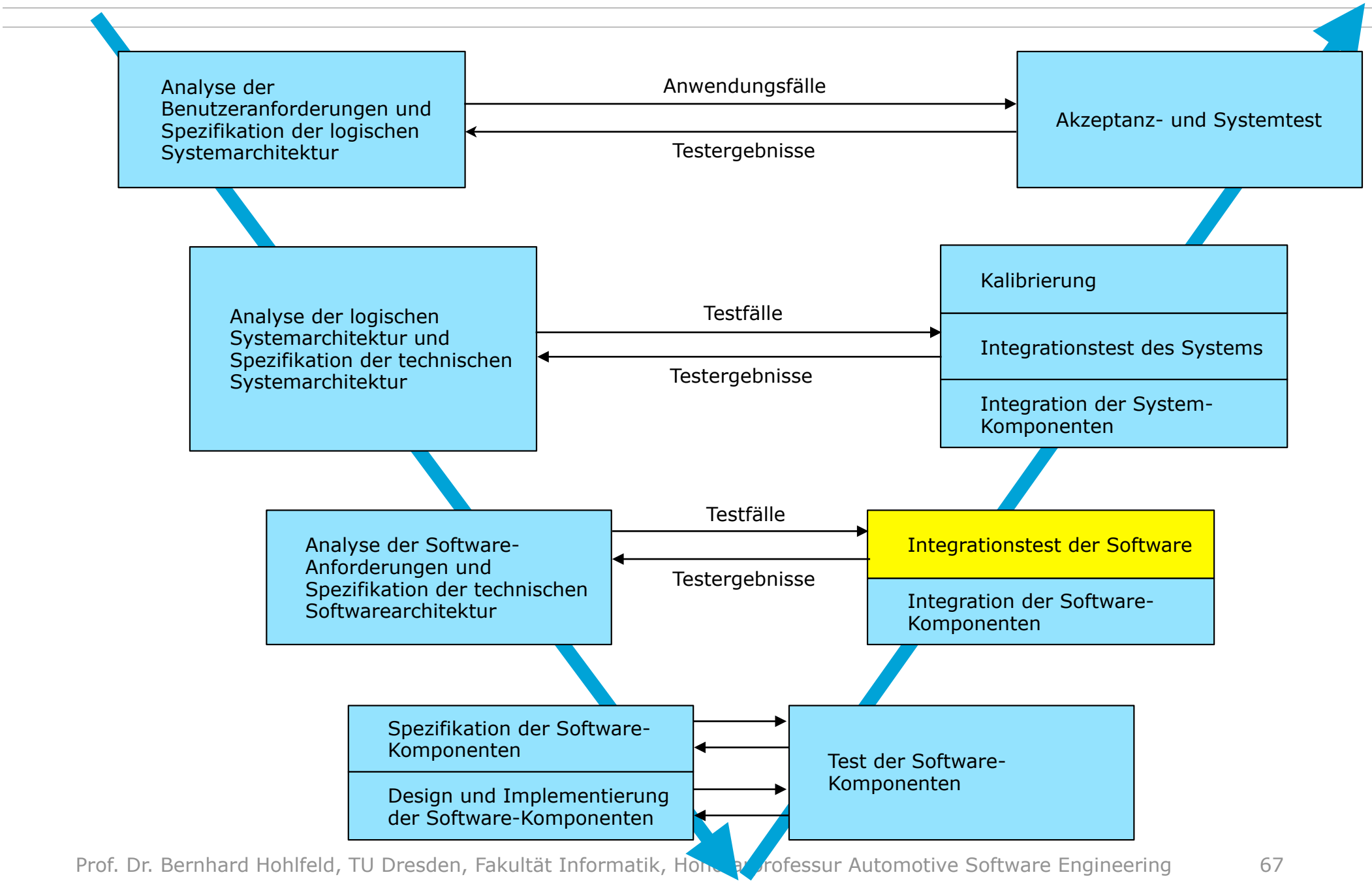
MSR/MEDOC

- What is MSR?
  - The MSR consortium (Manufacturer Supplier Relationship) is an initiative of the top managers of development (E-Leiter) of the German car makers. It is run as task 11 within strategy circle 4 (electric/electronical development).
  - The mission is the development of cost reduction potentials by cooperation in non competitive areas and the use of synergy potentials in a common projects.
  - MSR supports the joint development of car manufacturers and their electric/ electronic component and system suppliers by enabling process synchronization and proper information exchange.
- What is MSR/MEDOC?
  - MSR/MEDOC is a subtask of the MSR consortium. MEDOC is an acronym for MSR Engineering Data Objects and Contents. MEDOC develops methods standards and implementation for information exchange in the engineering process.

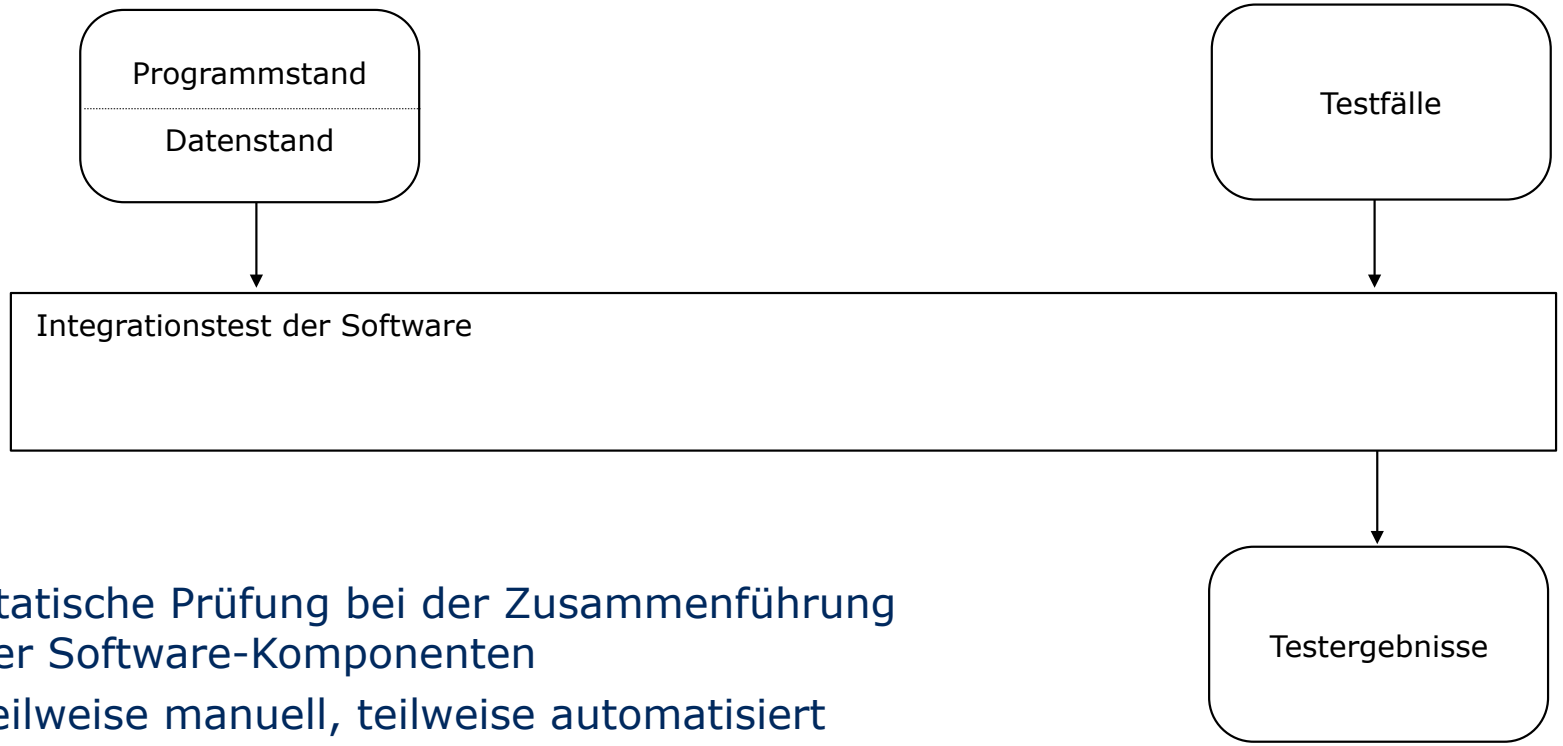
Quelle: http://www.msr-wg.de/medoc/

6. SW-Entwicklung / 2. Kernprozess

1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. **Integrationstest der Software**
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. Kalibrierung
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

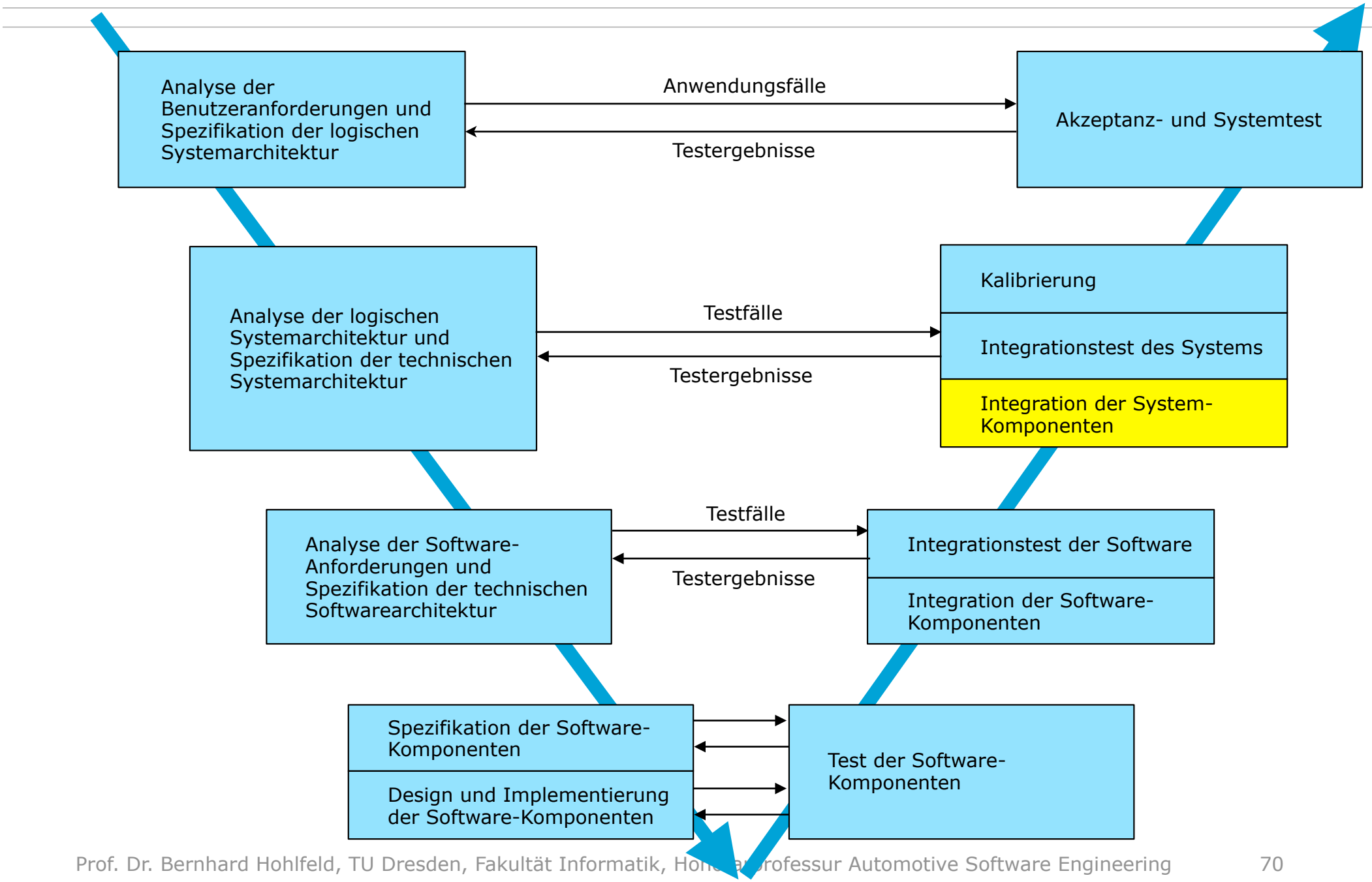
Integrationstest der Software



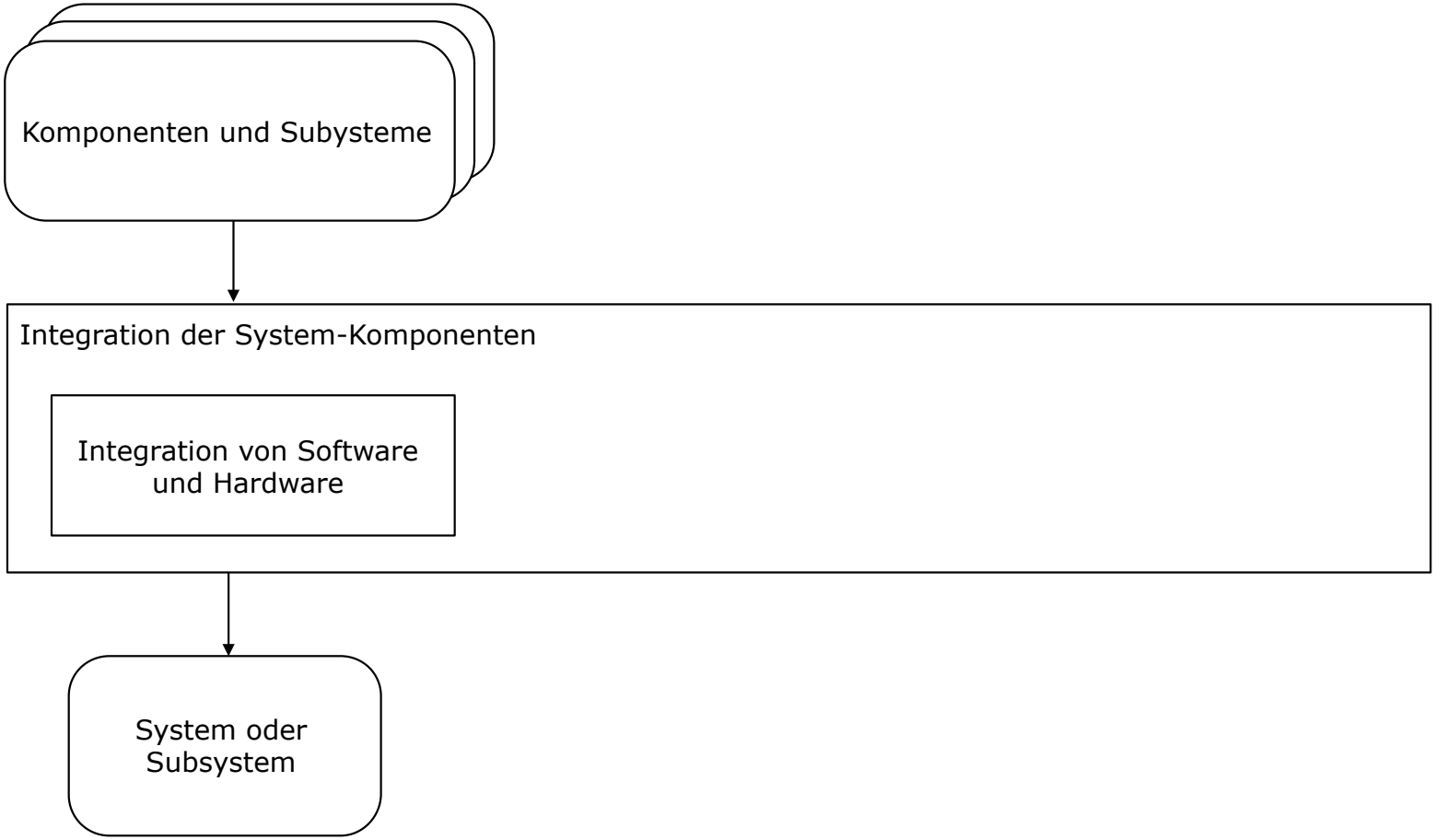
- Statische Prüfung bei der Zusammenführung der Software-Komponenten
- Teilweise manuell, teilweise automatisiert
- Schnittstellenspezifikation eingehalten?
- Namensraum für Variable eingehalten?
- Speicherlayout

6. SW-Entwicklung / 2. Kernprozess

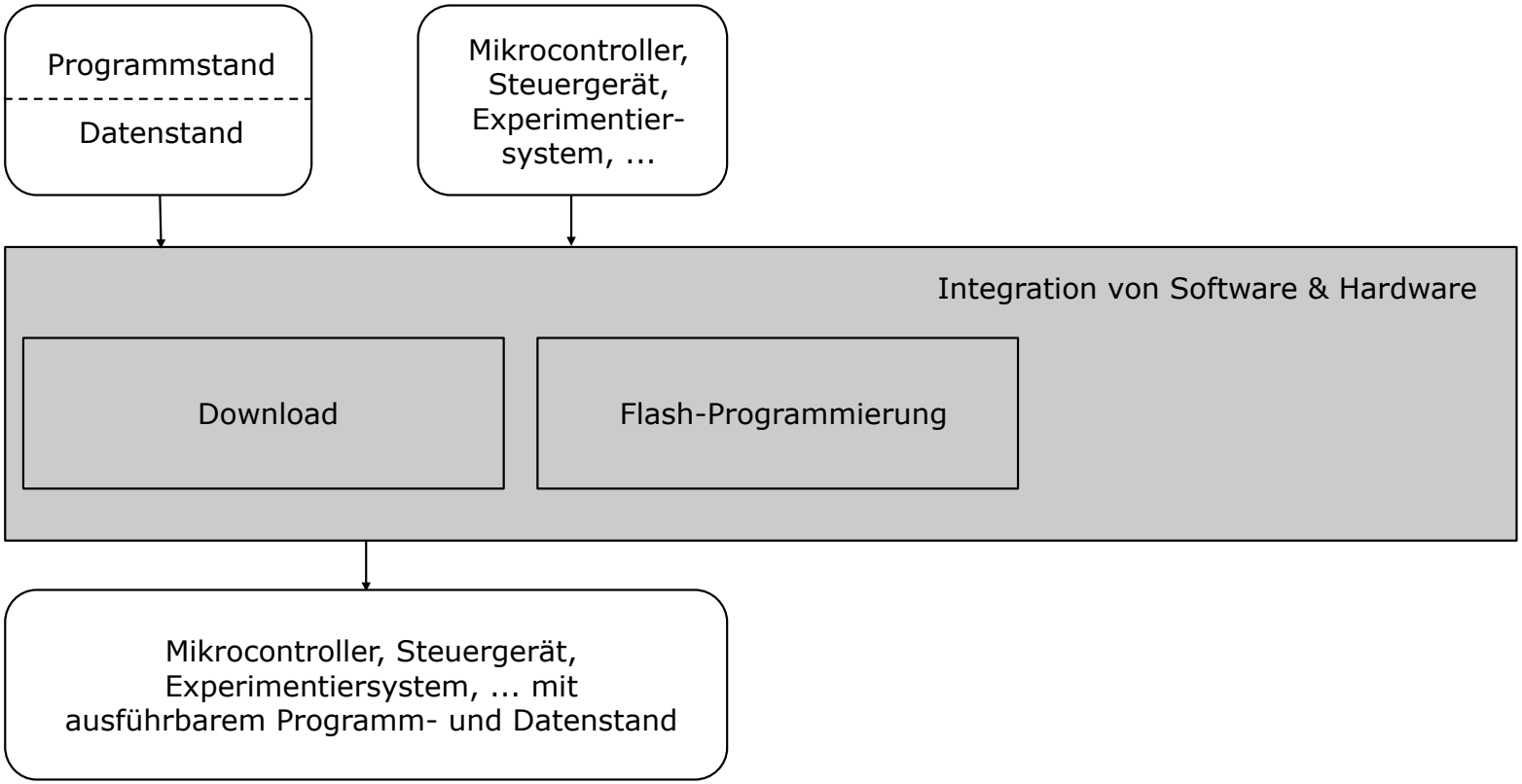
1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. **Integration der System-Komponenten**
12. Integrationstest des Systems
13. Kalibrierung
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

Integration der System-Komponenten



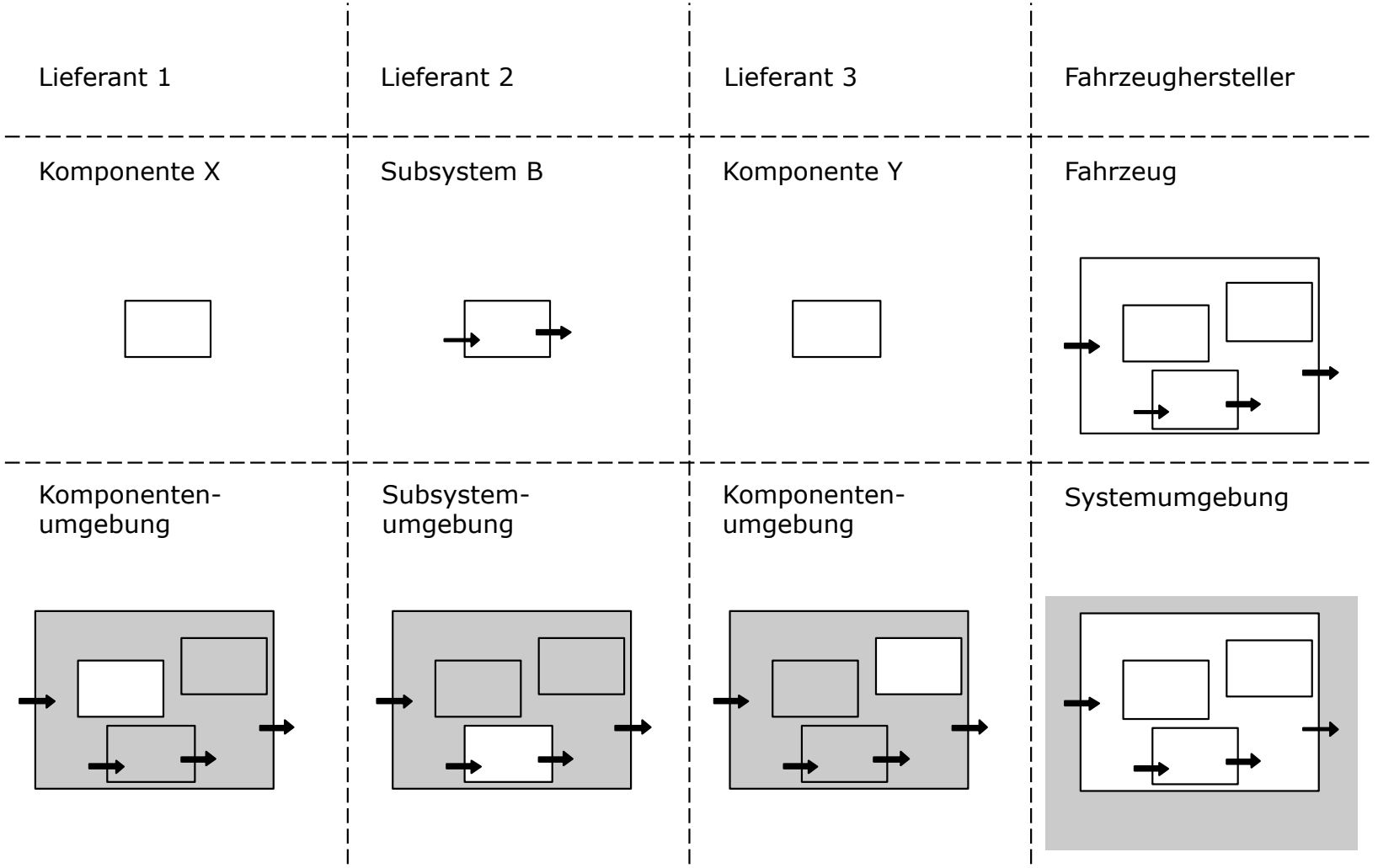
Integration von Software & Hardware



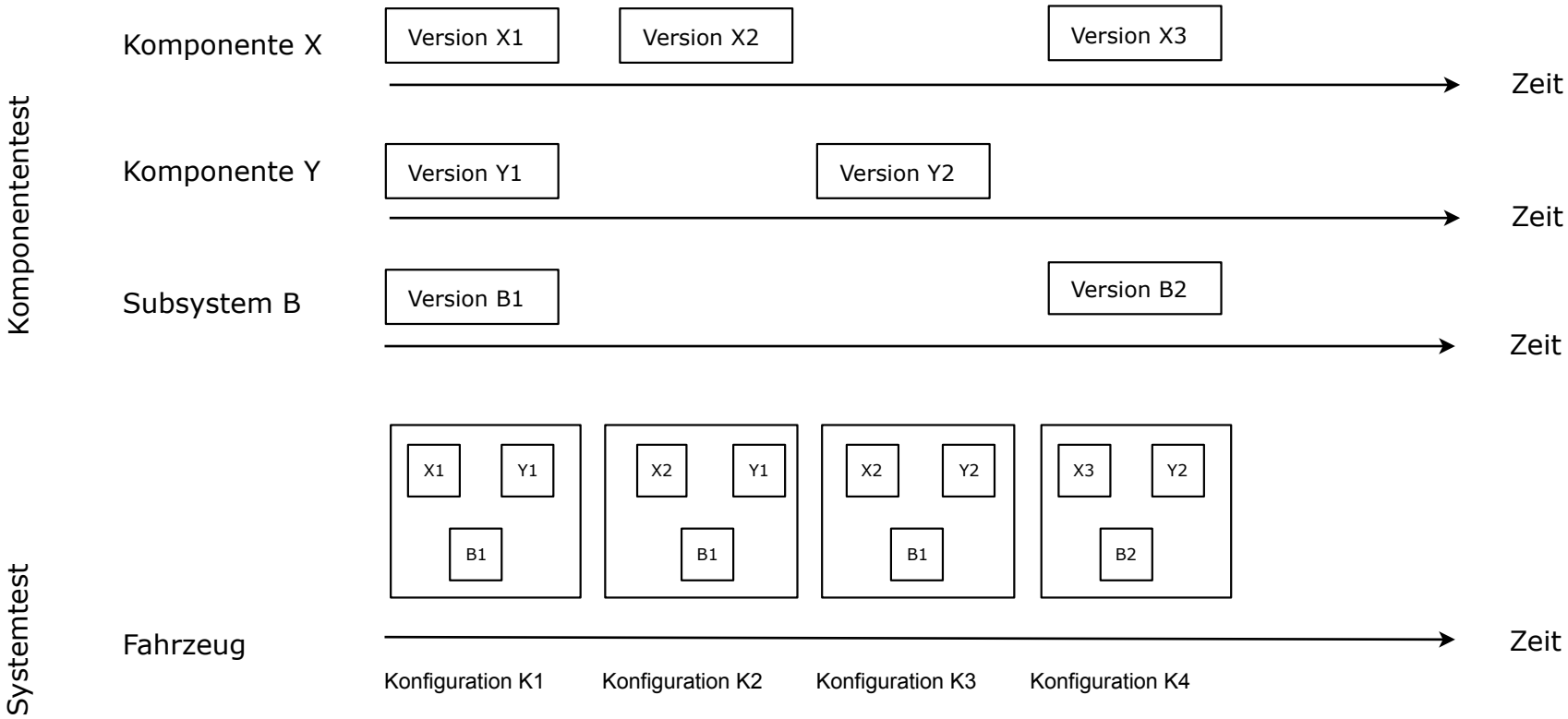
Integration von Software & Hardware

- Integration von Steuergeräten, Sollwergebern, Sensoren und Aktuatoren
  - Beim OEM Abnahmetest der vom Zulieferer gelieferten Komponenten und Subsysteme
  - Nur eingeschränkte Testmöglichkeiten beim Zulieferer
  - Komponentenintegration ist Synchronisationspunkt

Unterschiedliche Integrations-Umgebungen



Komponenten- und Systemtest

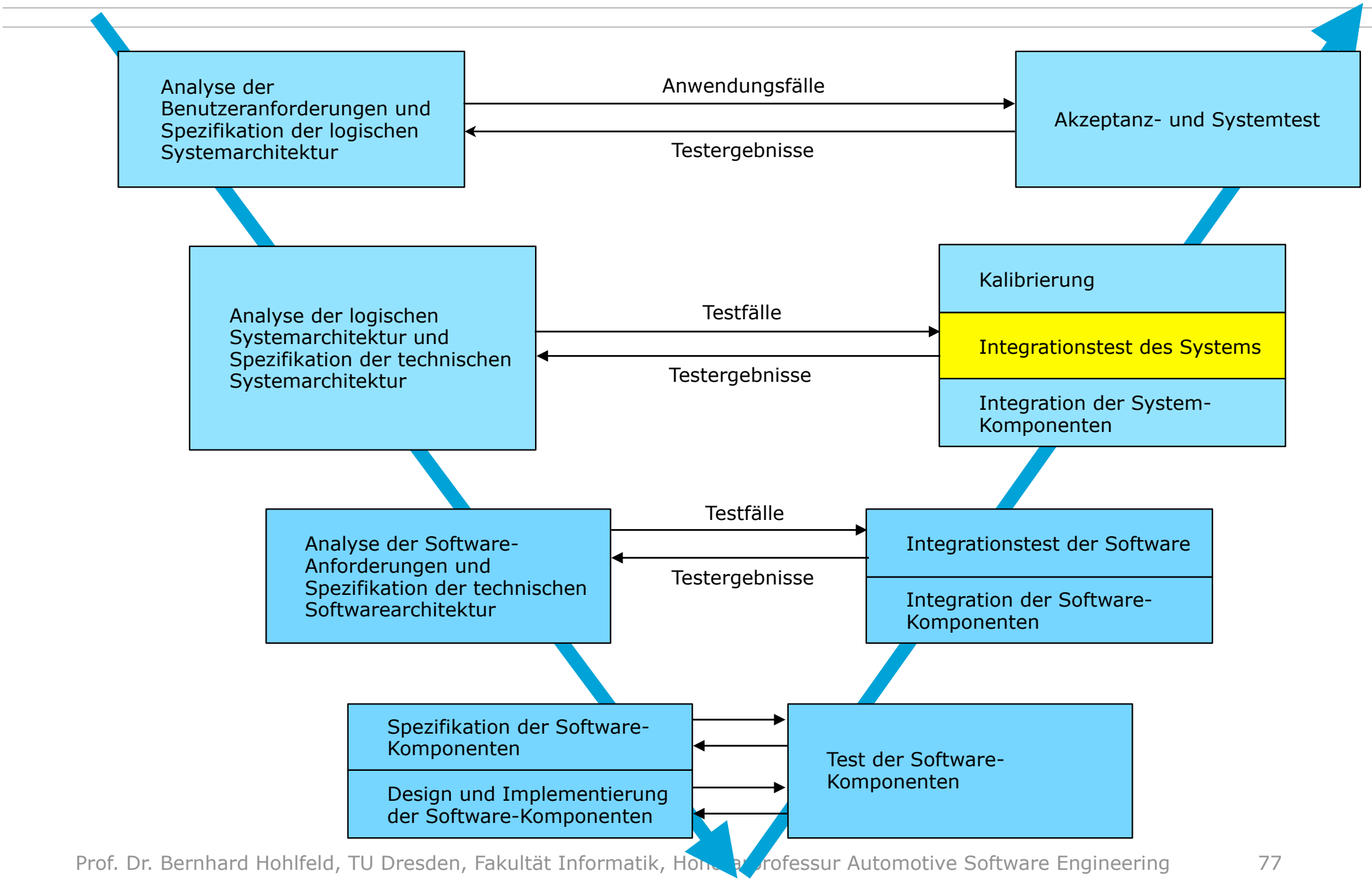


siehe auch 3. Unterstützungsprozesse

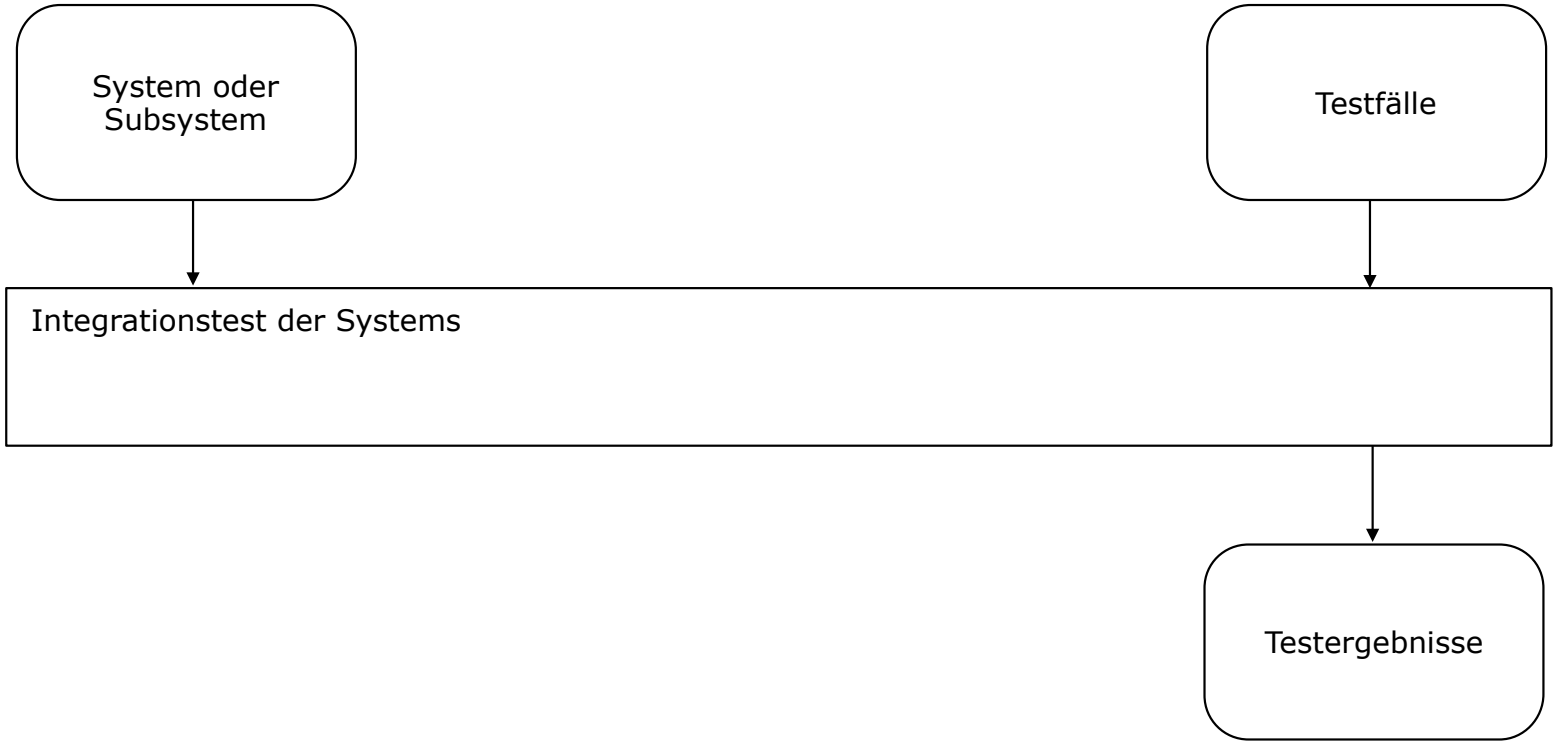
- 3. Konfigurationsmanangement
  - 3. Versionen und Konfigurationen

6. SW-Entwicklung / 2. Kernprozess

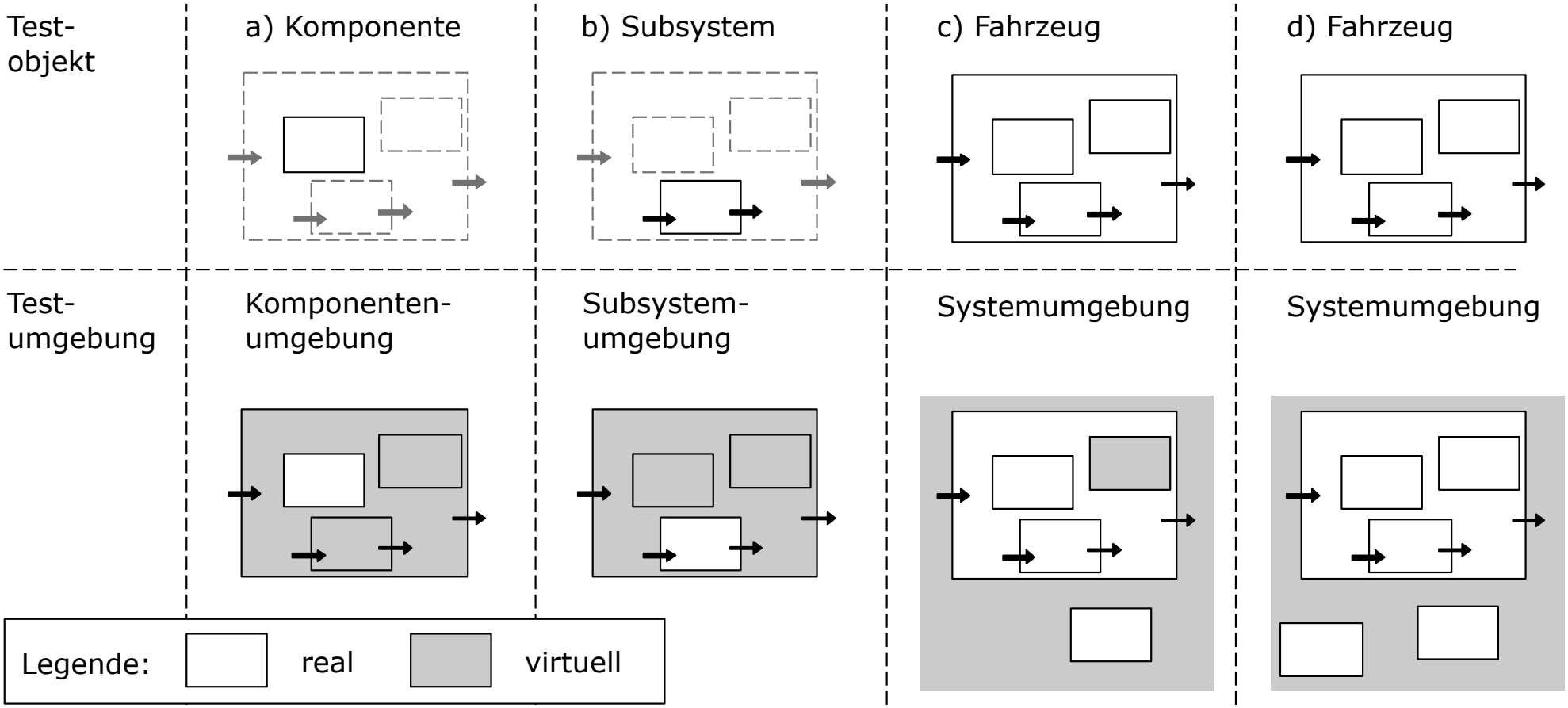
1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. Integration der System-Komponenten
12. **Integrationstest des Systems**
13. Kalibrierung
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

Integrationstest des Systems

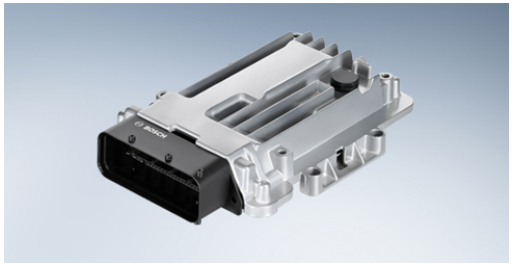


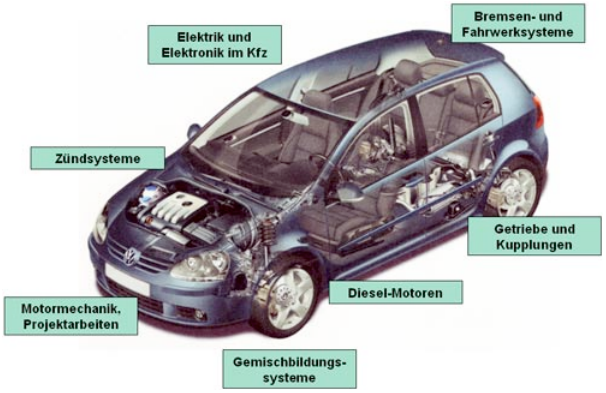
Testobjekt und Testumgebung



Begriffsdefinitionen

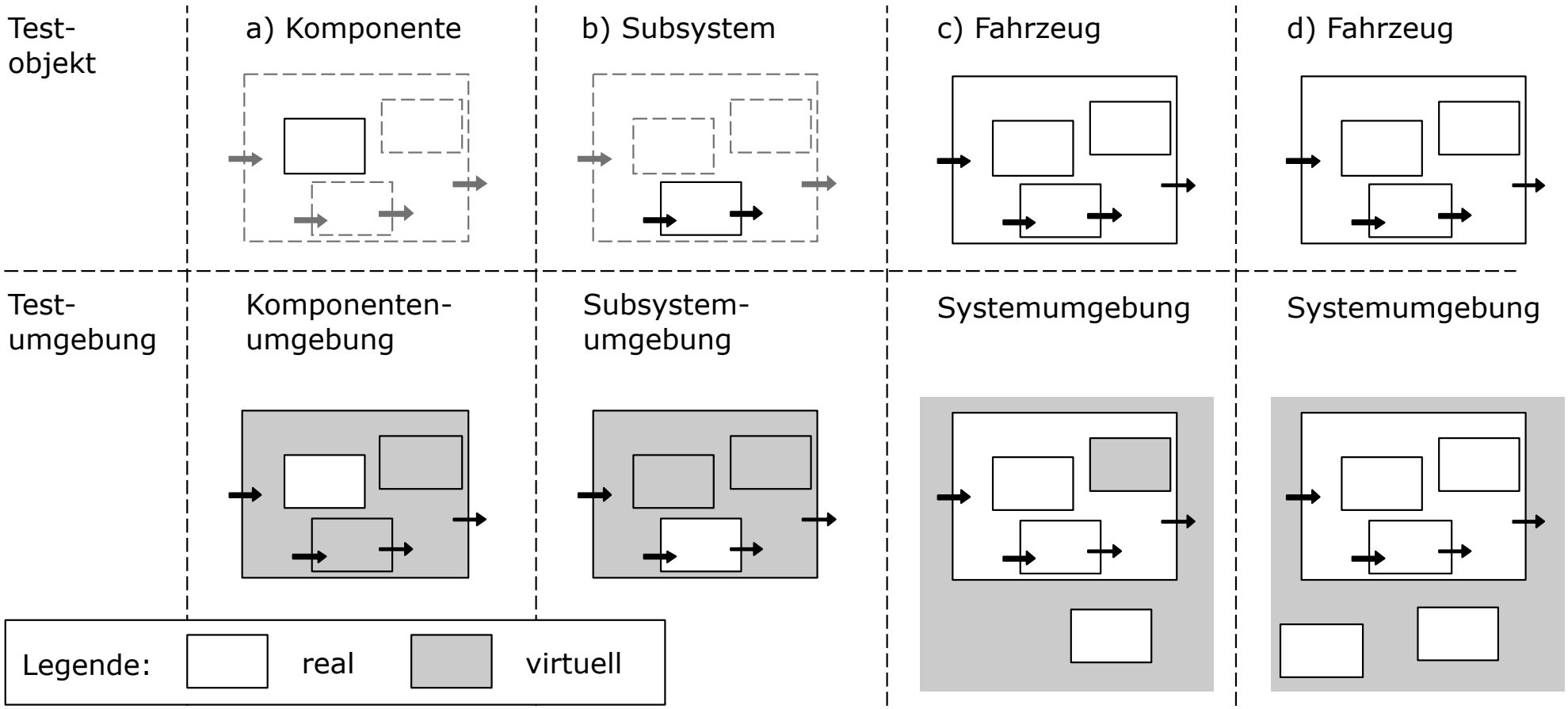
- MIL Model in the Loop, SIL Software in the Loop
  - Modell / SG-Software läuft auf simuliertem SG, das von simulierter Fahrzeugumgebung (rechnergenerierten Sensor- und Bussignalen) gespeist wird.
- HIL Hardware in the Loop
  - Reale SG-Hardware wird von simulierter Fahrzeugumgebung gespeist
- Prototyp
  - Simuliertes Steuergerät im Fahrzeug (“PC im Kofferraum”)
- Prüfstand, Fahrversuch
  - Steuergerät im Fahrzeug unter Laborbedingungen bzw. auf der Strasse (Testgelände, öffentliches Strassennetz)



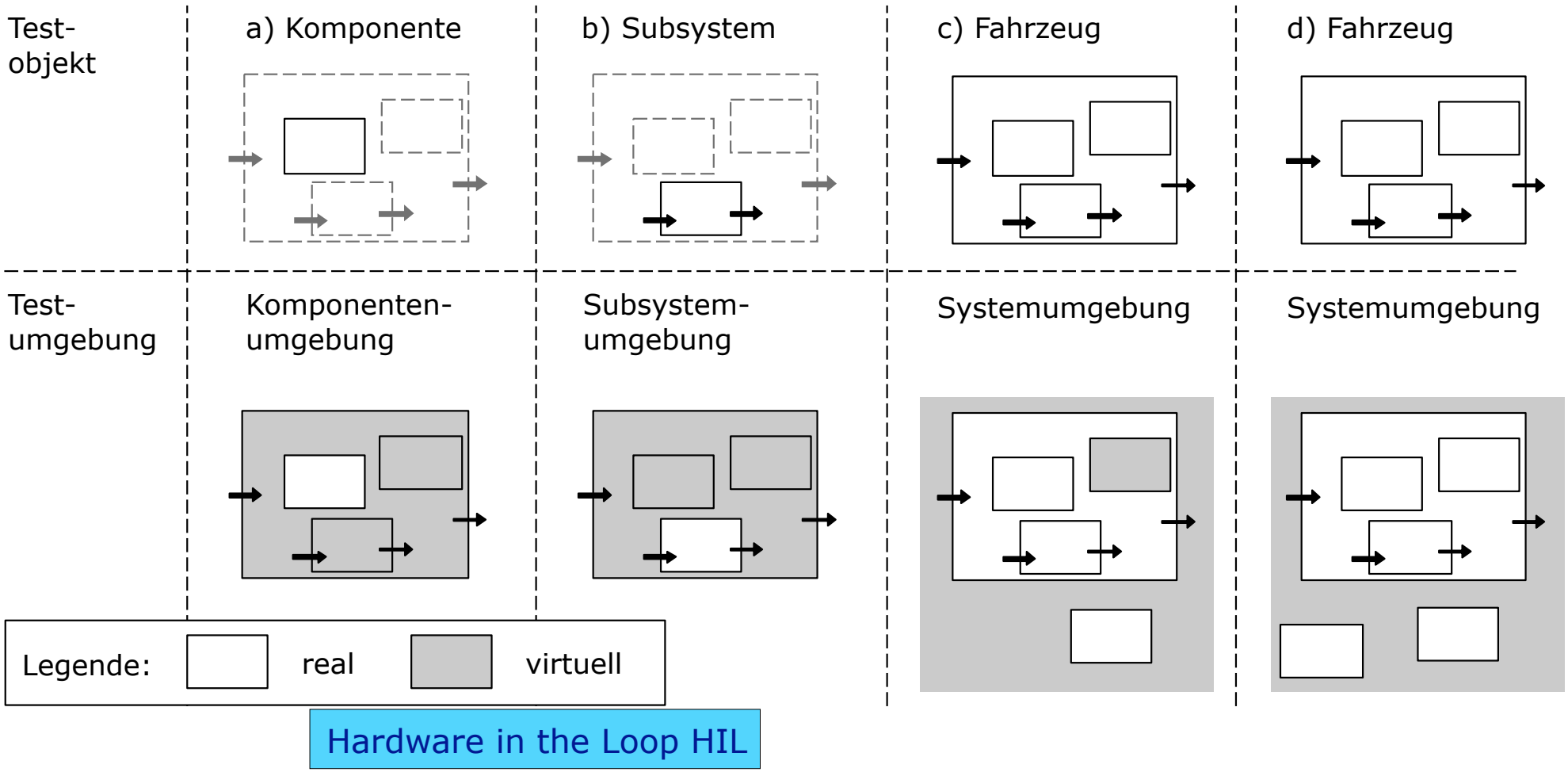


	Steuergerät	
Umgebung, Fahrzeug	simuliert	real
simuliert	MIL, SIL	HIL
real	Prototyp	Prüfstand, Fahrversuch

Testobjekt und Testumgebung

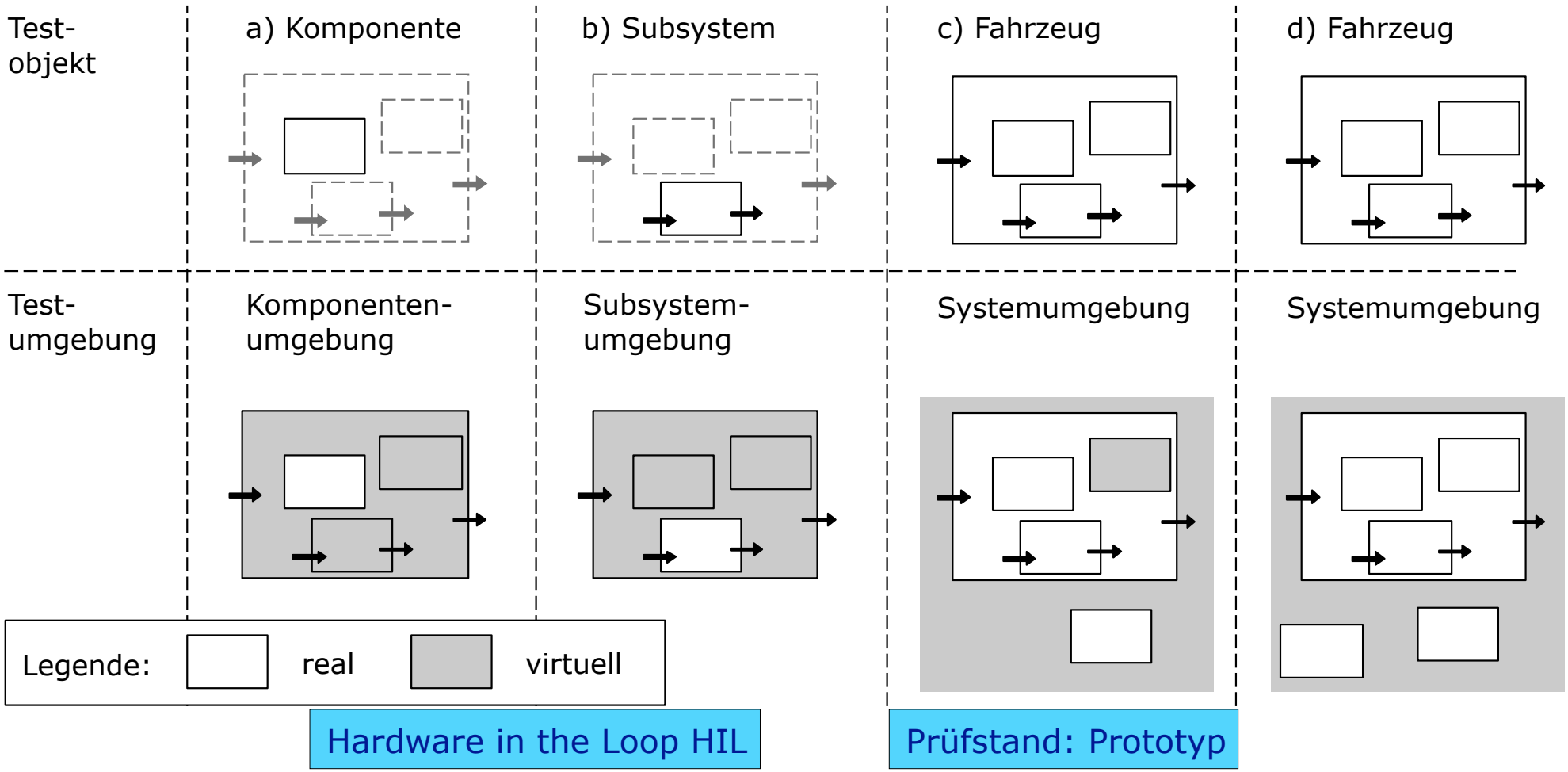


Testobjekt und Testumgebung

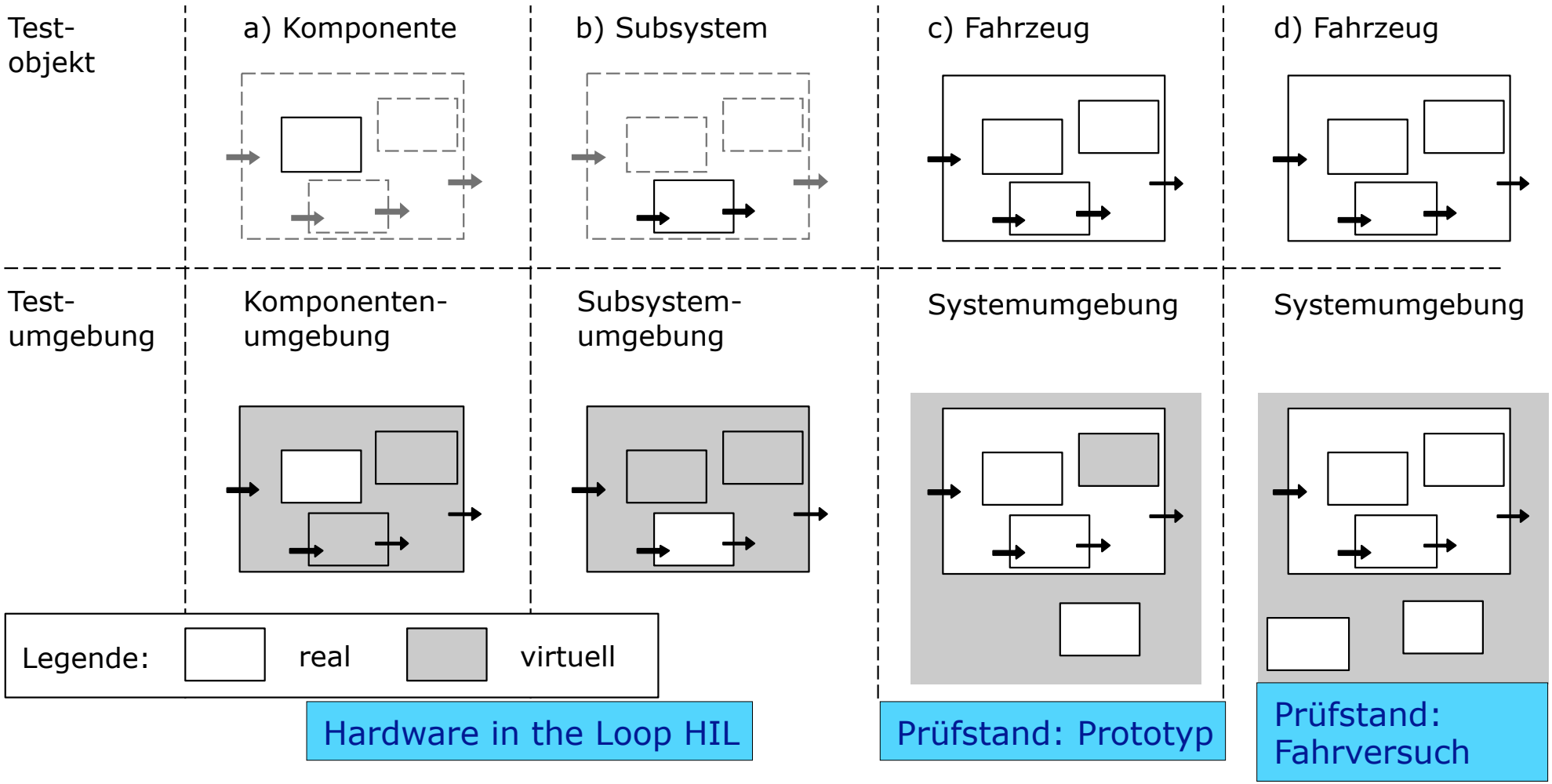


Hardware in the Loop HIL

Testobjekt und Testumgebung

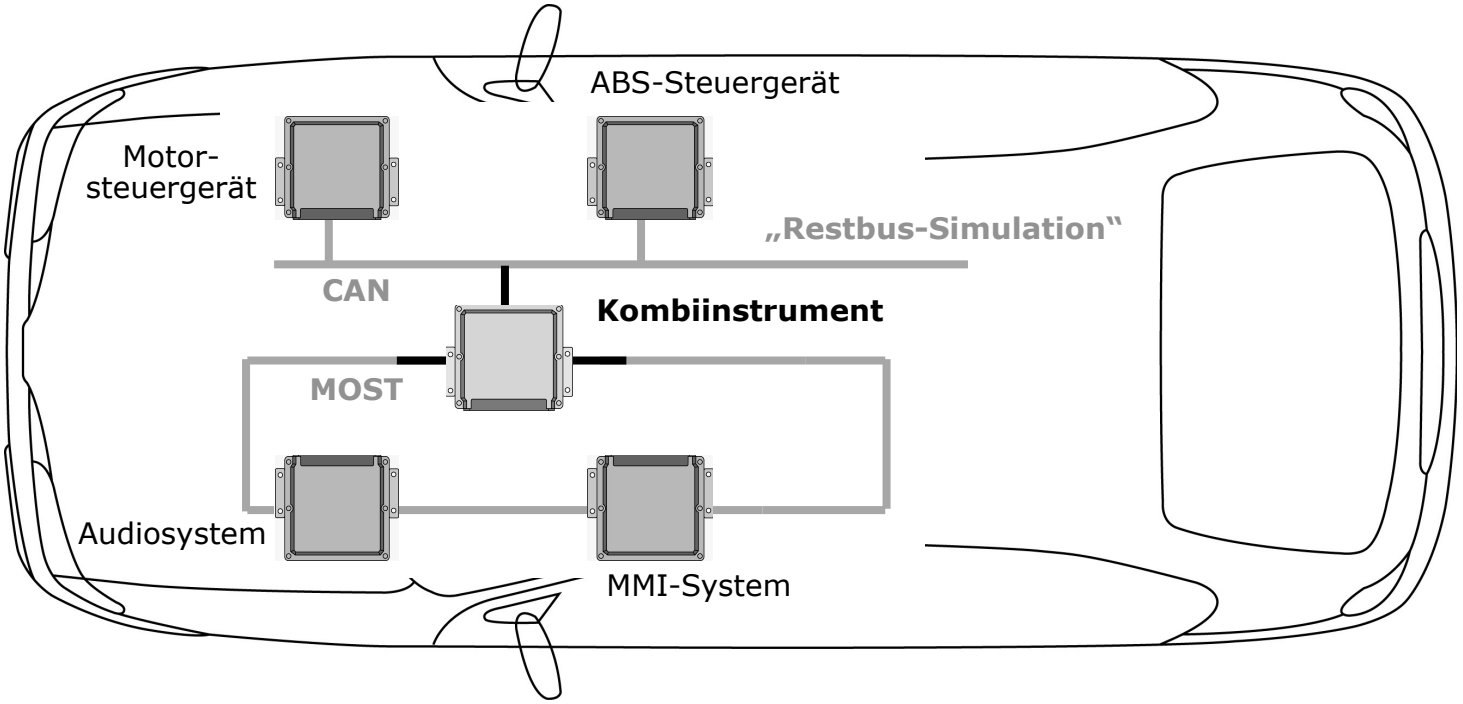


Testobjekt und Testumgebung

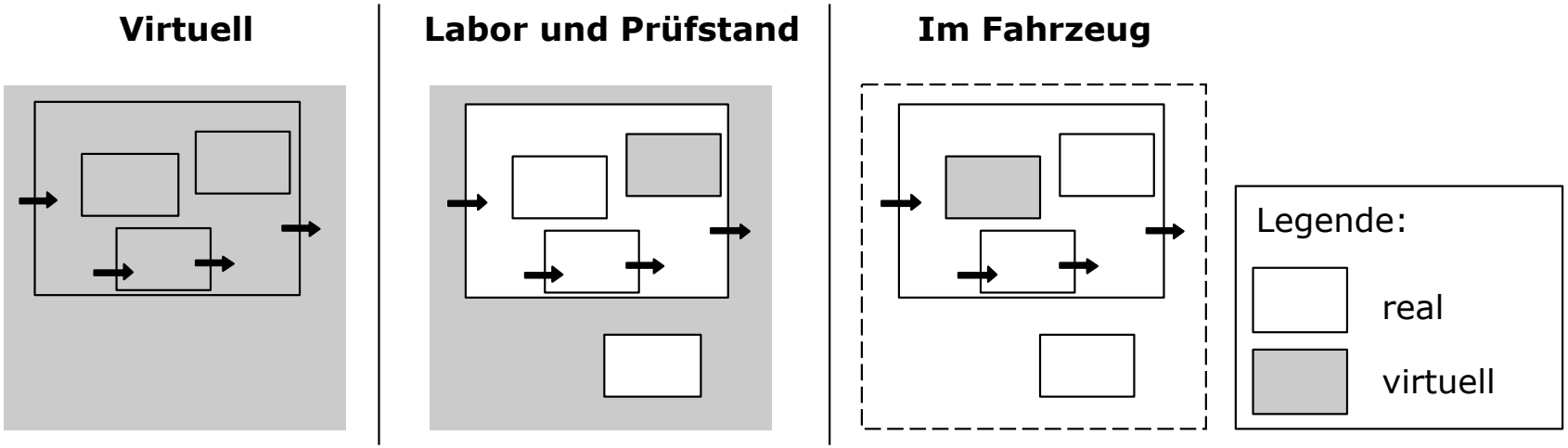


Virtuelle Netzwerkumgebung für Kombiinstrument

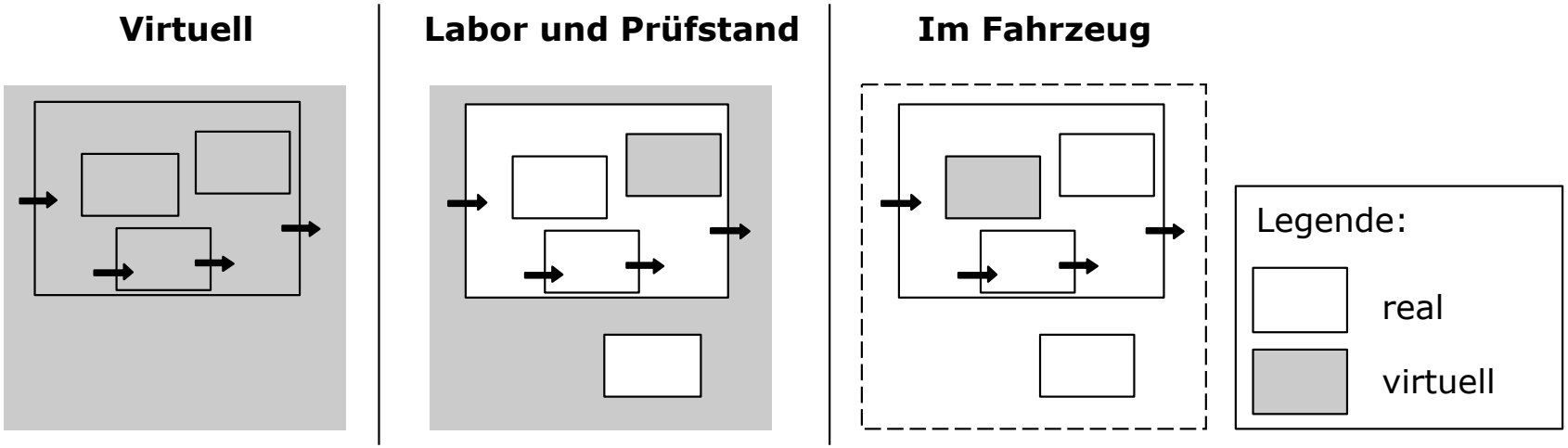
- Vorteile virtueller Testumgebungen
  - Prüfschritte im Labor oder am Prüfstand statt im Fahrzeug
  - Reproduzierbarkeit, Automatisierung
  - Extremsituationen ohne Gefährdung von Testfahrern oder Prototypen.



Durchgängiger Integrations- und Testprozess

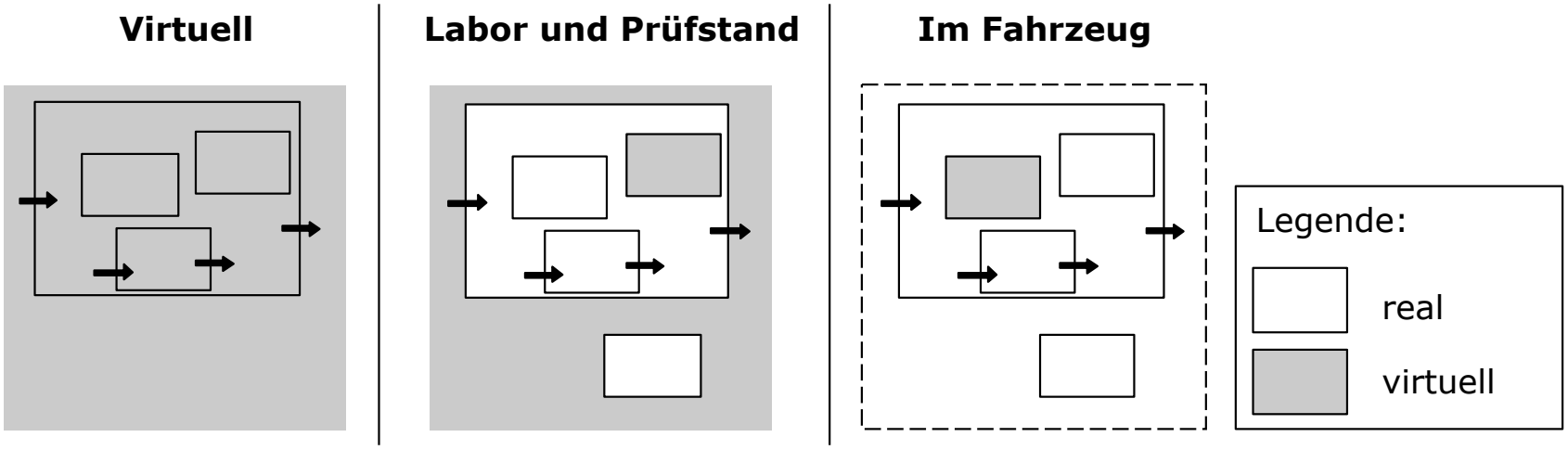


Durchgängiger Integrations- und Testprozess



Software in the Loop SIL

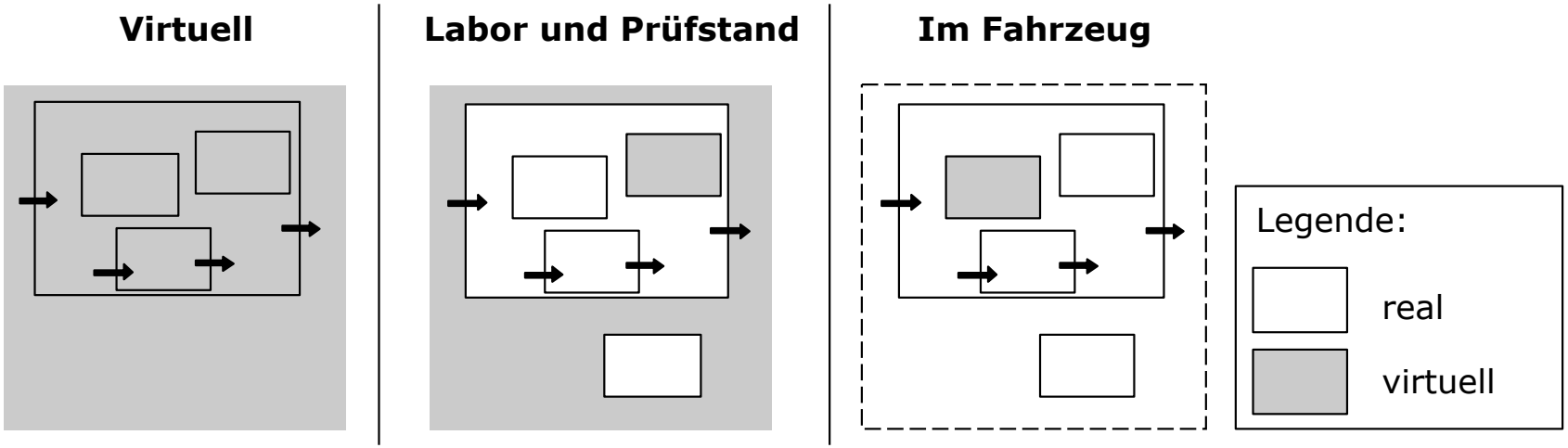
Durchgängiger Integrations- und Testprozess



Software in the Loop SIL

Prüfstand: Prototyp

Durchgängiger Integrations- und Testprozess



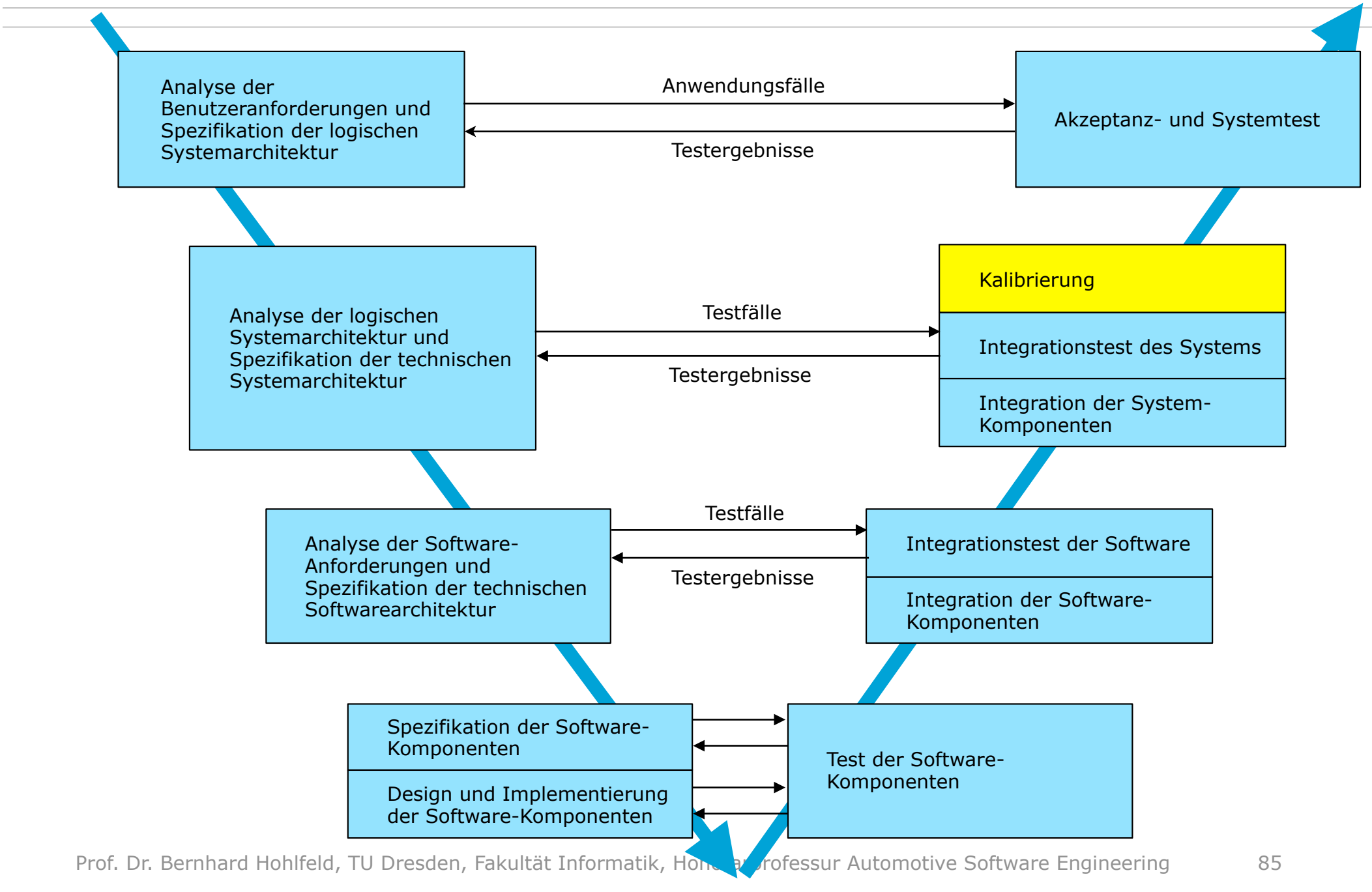
Software in the Loop SIL

Prüfstand: Prototyp

Prototyp im Fahrversuch

6. SW-Entwicklung / 2. Kernprozess

1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. **Kalibrierung**
14. Akzeptanz- und Systemtest


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

Kalibrierung

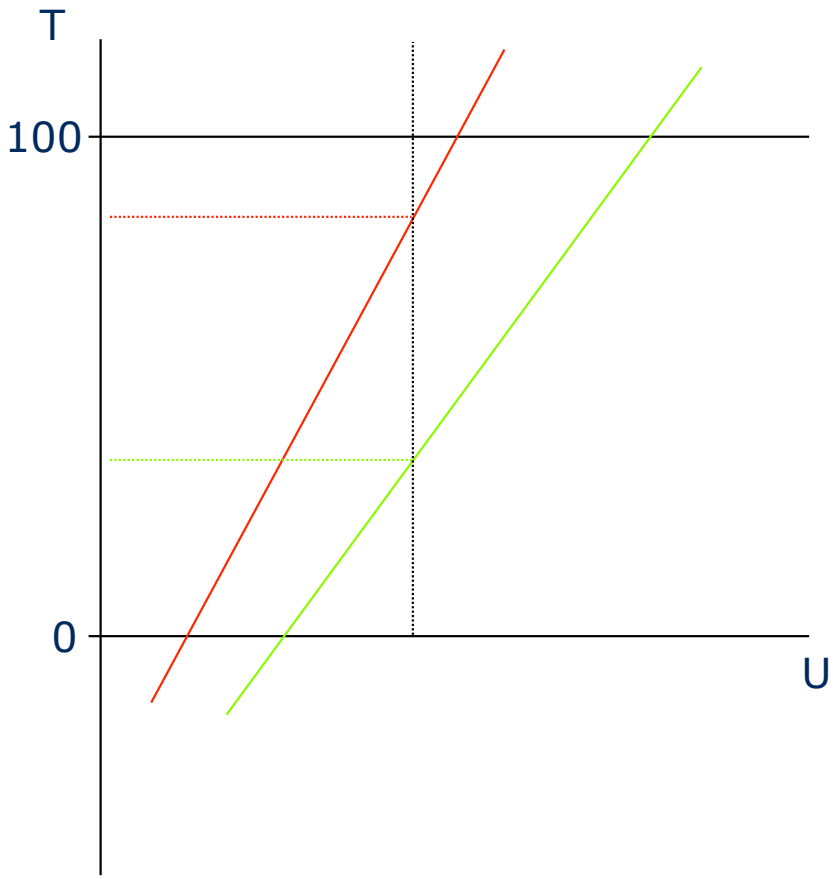
- Kalibrierung in der Messtechnik ist ein Messprozess zur Feststellung und Dokumentation der Abweichung eines Messgerätes oder einer Maßverkörperung zu einem anderen Gerät oder Maßverkörperung, das in diesem Fall als Normal bezeichnet wird. In der Definition des VIM von JCGM 2008 [1] kommt zwingend ein zweiter Schritt zur Definition der Kalibrierung hinzu, nämlich die Berücksichtigung der ermittelten Abweichung bei der anschließenden Benutzung des Messgerätes zur Korrektur der abgelesenen Werte.
- Quelle: Wikipedia

- Messen der Abweichung vom „Normal"wert
- Korrektur

- „Anpassen an die Physik"

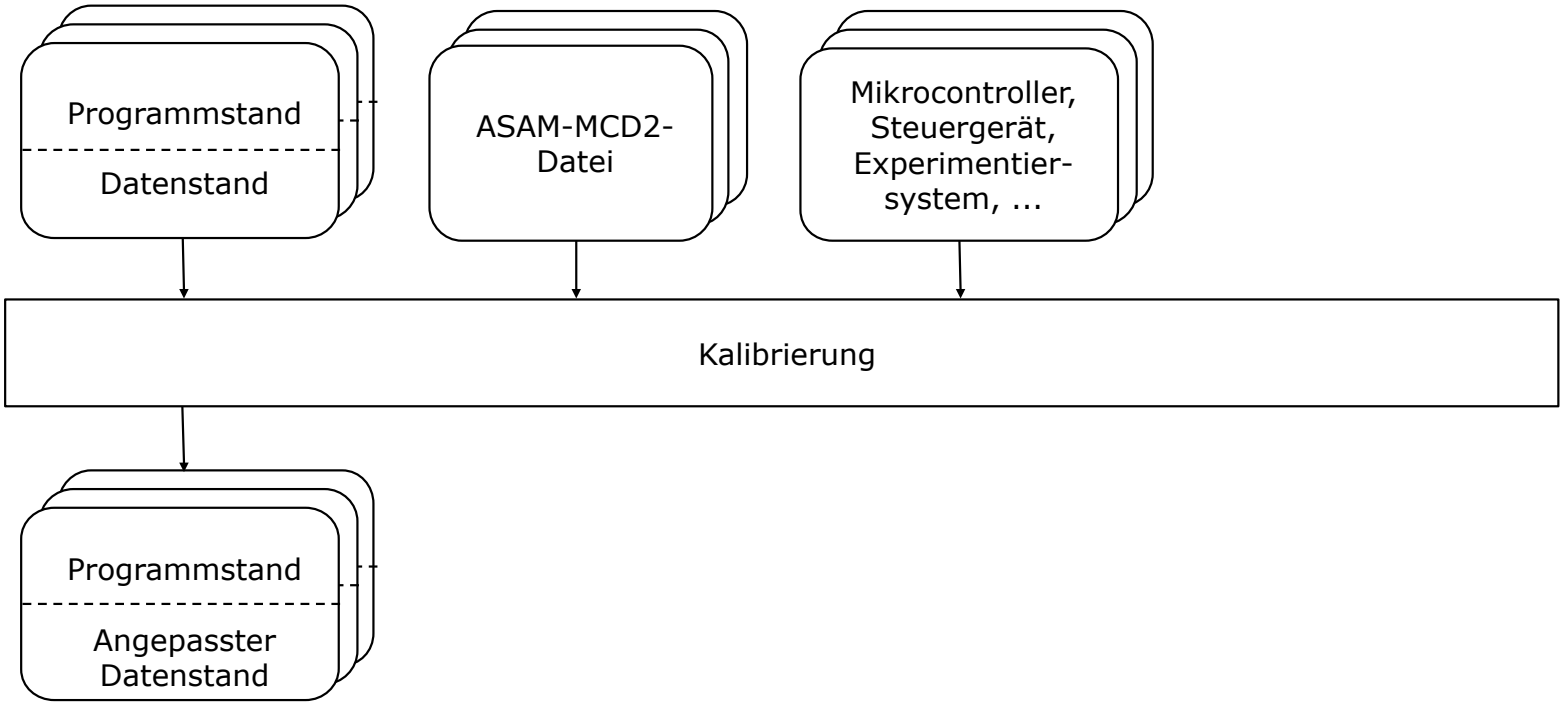
Kalibrierung - Ein einfaches Beispiel

- Ein Sensor erfasst Temperaturwerte
  - Kühlmitteltemperatur
  - Aussentemperatur
  - …
- Die Temperaturwerte werden an das Kombiinstrument gesendet
- Annahmen
  - Temperatur induziert Spannung im Sensor
  - Lineare Abhängigkeit:
    $U = f(T) + c$
- Sensorrauschen
- Intelligenter Sensor



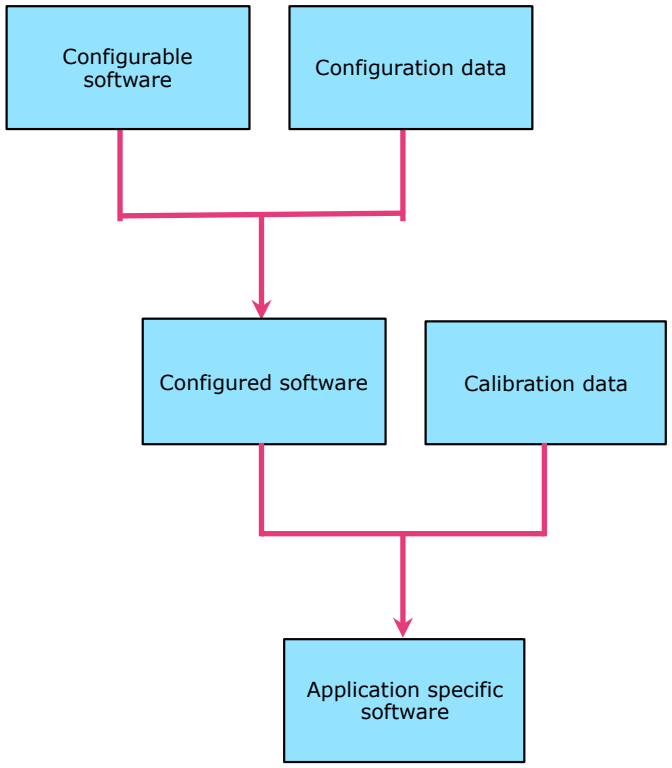
Kalibrierung

- Fahrzeugindividuelle Einstellung der Parameter der Software-Funktionen
- Häufig direkt im Fahrzeug bei laufenden Systemen
- Kalibriersystem: Steuergerät mit Off-Board-Schnittstelle zu Mess- und Kalibrierwerkzeugen
- Angepasste Datenstände in Festwertspeicher (ROM, EEPROM, Flash)



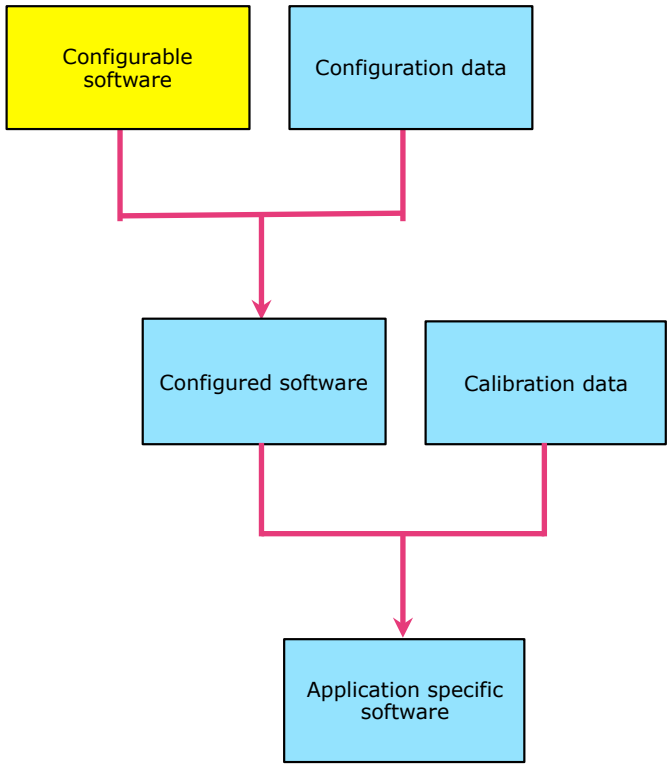
ISO 26262 Road vehicles — Functional safety

- Part 6: Product development: software level
- The objective of software configuration is to enable controlled changes in the behaviour of the software for different applications.



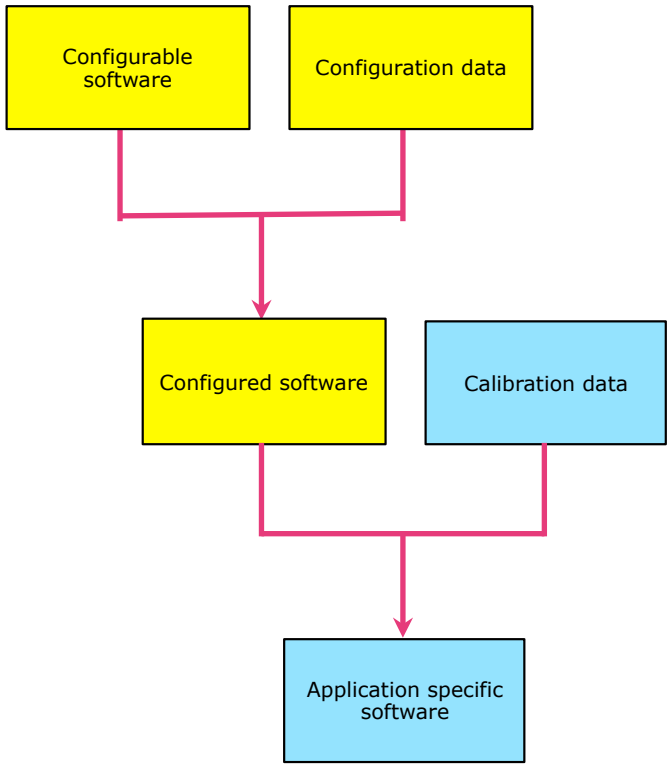
Annex C Software configuration - Verifikation

- Schritt 1:
  Verifikation der konfigurierbaren Software



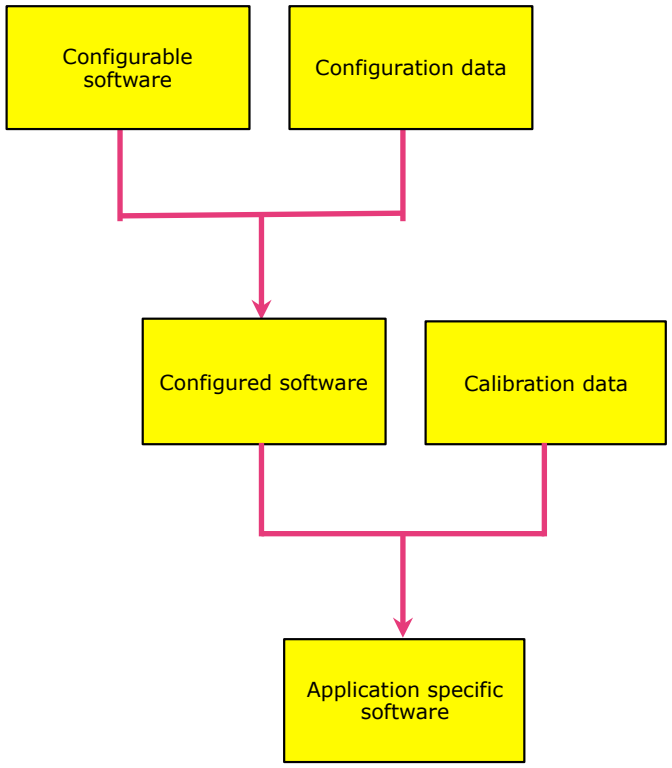
Annex C Software configuration - Verifikation

- Schritt 2:
  Verifikation der konfigurierten Software unter Verwendung der verifizierten konfigurierbaren Software und der Konfigurationsdaten
- Die konfigurierbare Software wird für beliebige (gültige) Konfigurationsdaten nur einmal verifiziert



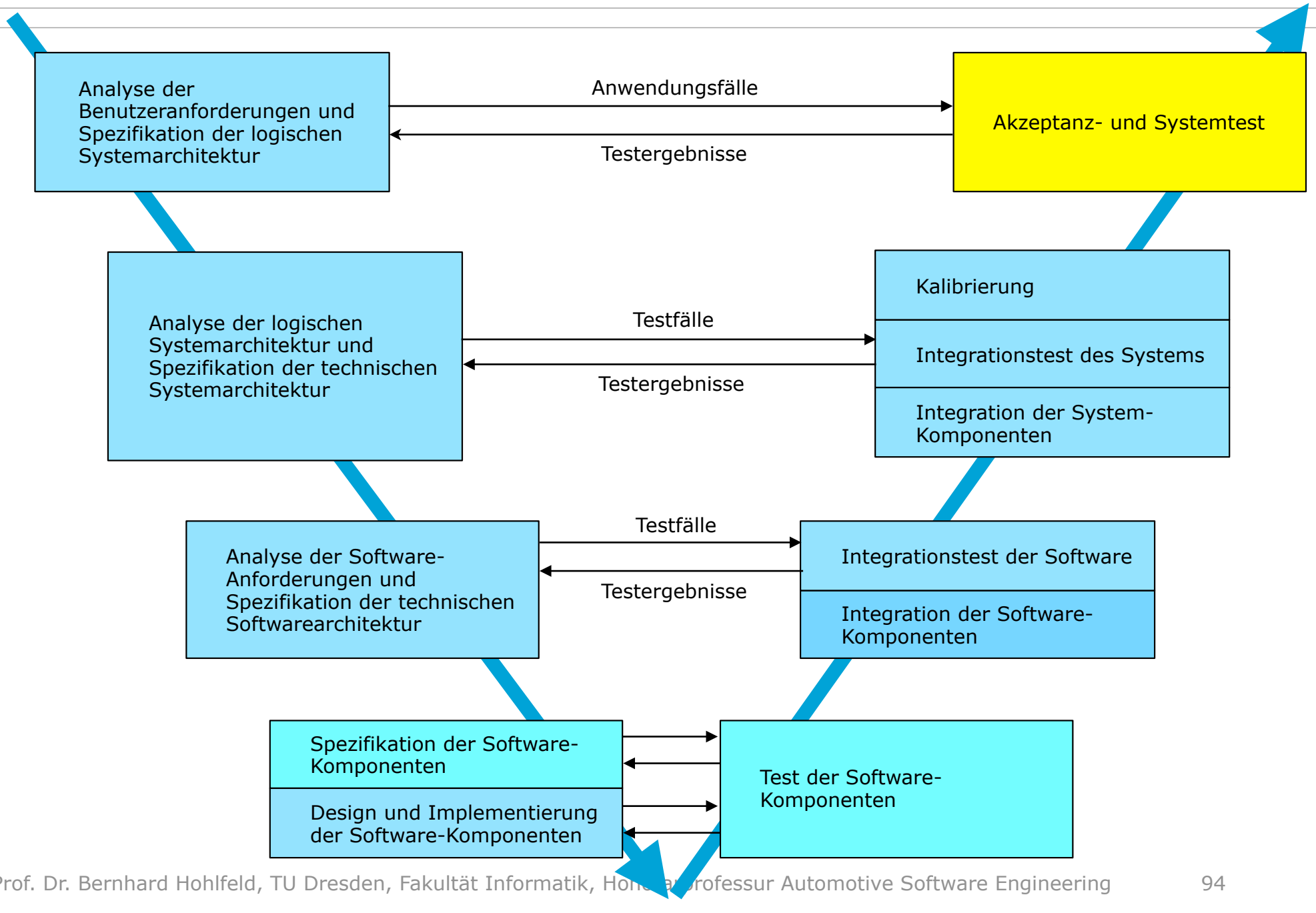
Annex C Software configuration - Verifikation

- Schritt 3:
  Verifikation der anwendungsspezifischen Software unter Verwendung der verifizierten konfigurierten Software und der Kalibrierungsdaten
- Die konfigurierte Software wird für beliebige (gültige) Kalibrierungsdaten nur einmal verifiziert



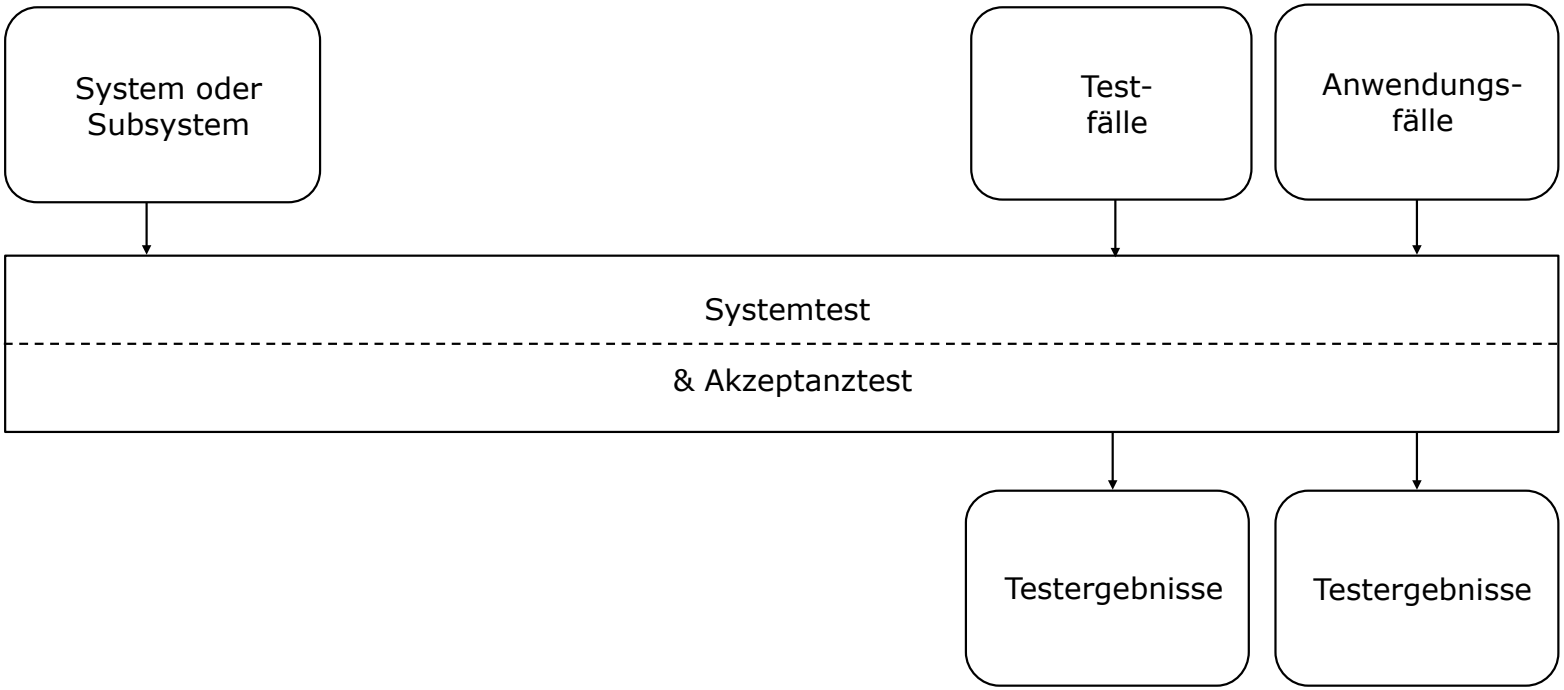
6. SW-Entwicklung / 2. Kernprozess

1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
6. Spezifikation der Software-Komponenten
7. Design und Implementierung der Software-Komponenten
8. Test der Software-Komponenten
9. Integration der Software-Komponenten
10. Integrationstest der Software
11. Integration der System-Komponenten
12. Integrationstest des Systems
13. Kalibrierung
14. **Akzeptanz- und Systemtest**


Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
Anwendungsfälle
Testergebnisse
Akzeptanz- und Systemtest
Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
Testfälle
Testergebnisse
Kalibrierung
Integrationstest des Systems
Integration der System-Komponenten
Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur
Testfälle
Testergebnisse
Integrationstest der Software
Integration der Software-Komponenten
Spezifikation der Software-Komponenten
Design und Implementierung der Software-Komponenten
Test der Software-Komponenten

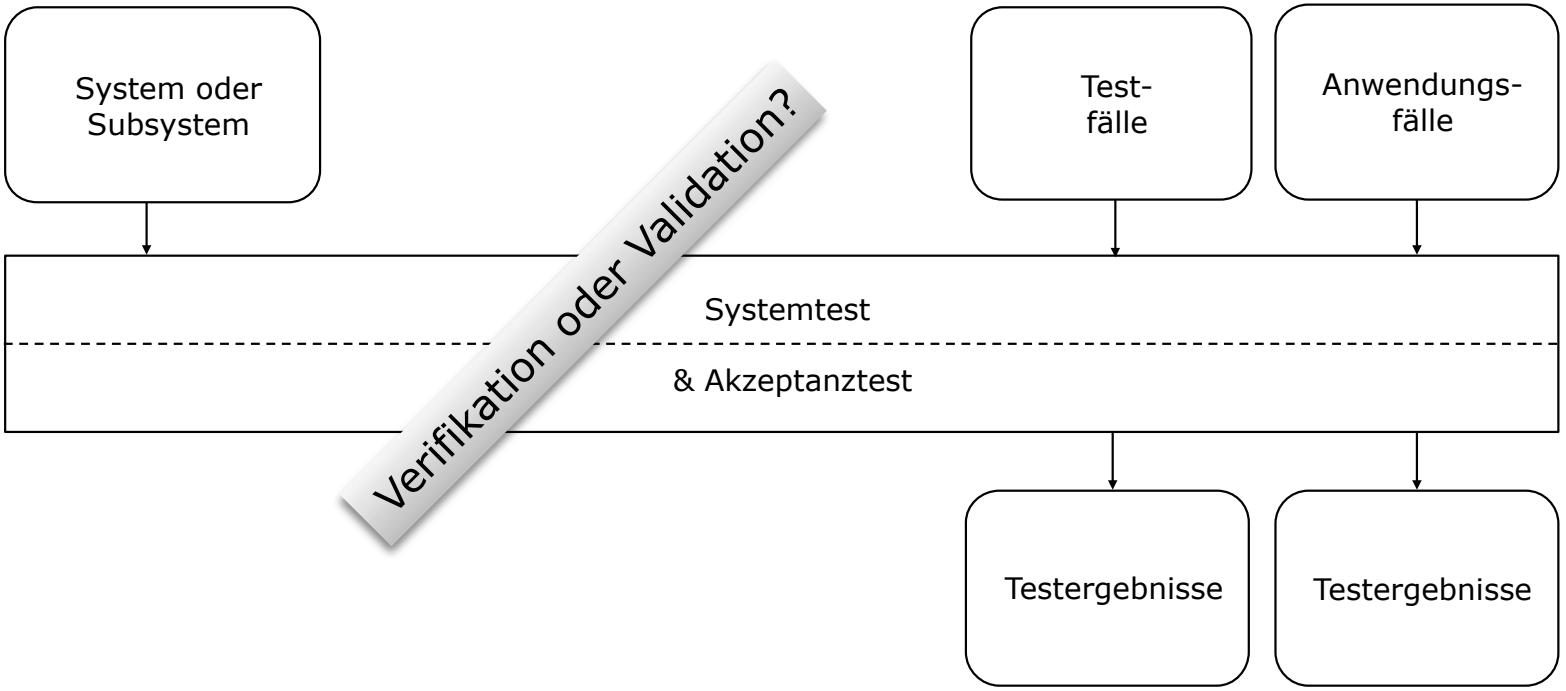
Akzeptanz- und Systemtest: Prüfung im Fahrzeug

- Restrisiko nach Simulation
- Systemtest im Fahrversuch: Akzeptanztest in der realen Betriebsumgebung
- Fahrzeugtauglicher Zugang zu Steuergeräten und Netzwerken
- Mobile Messtechnik

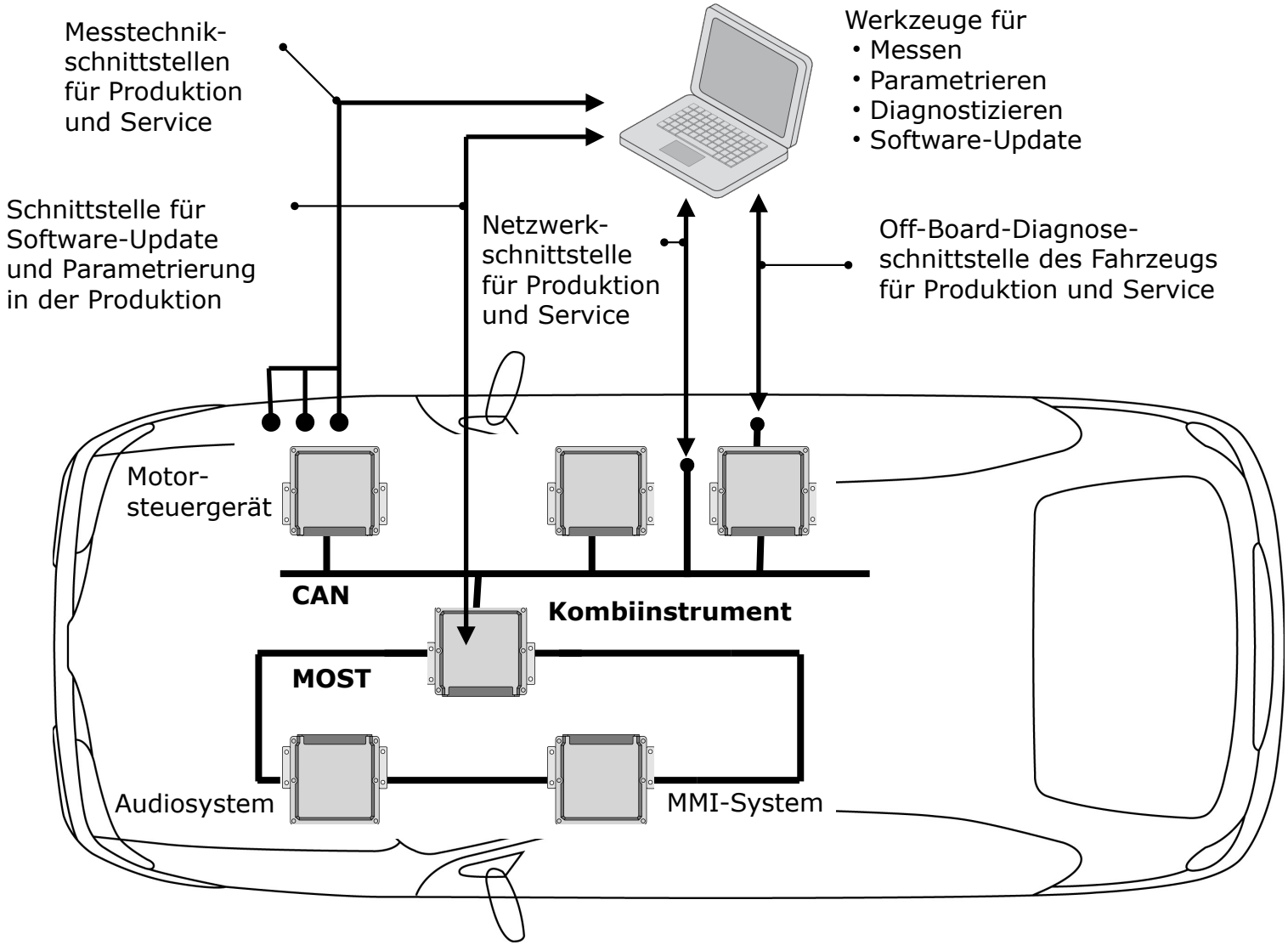


Akzeptanz- und Systemtest: Prüfung im Fahrzeug

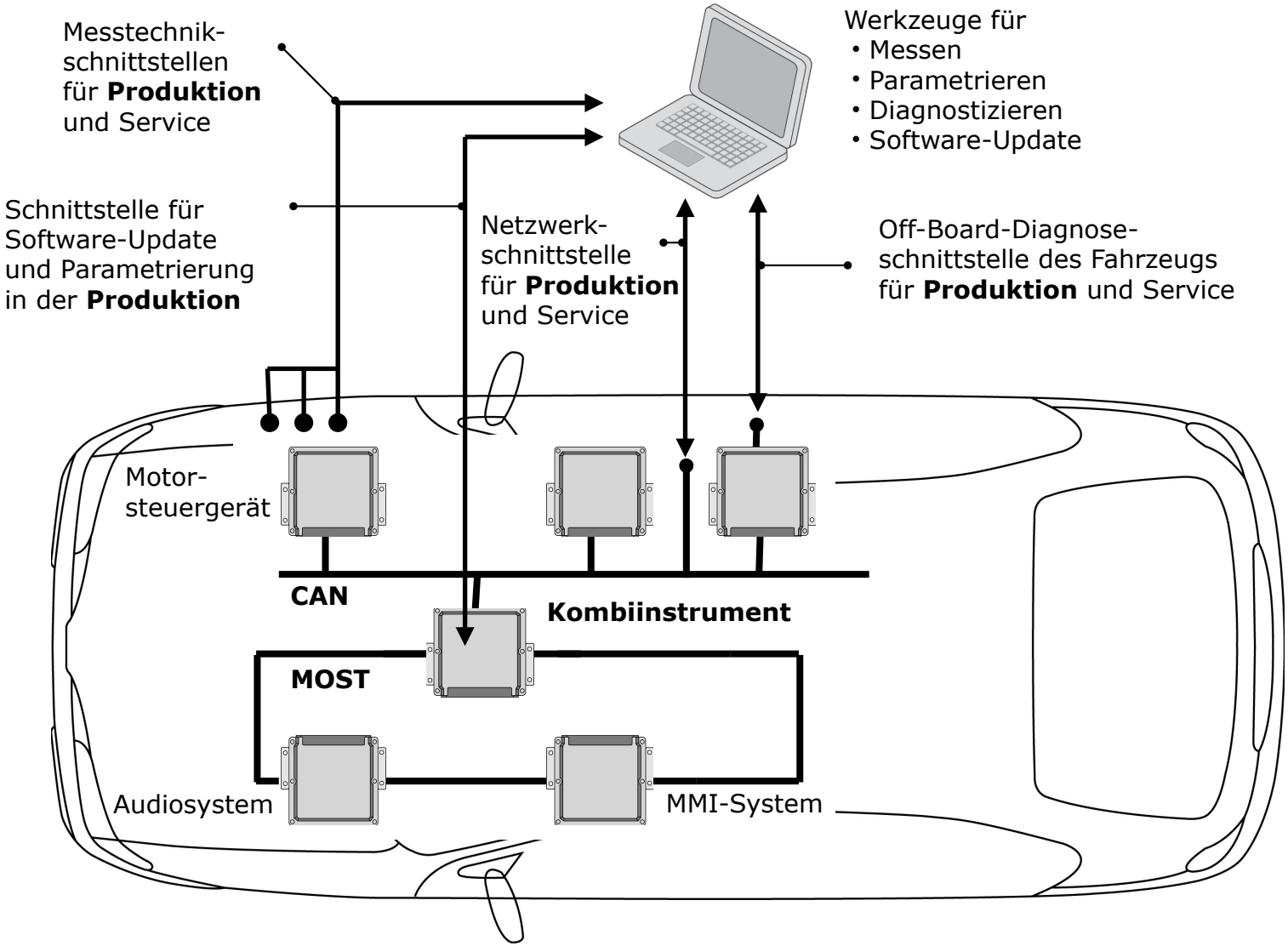
- Restrisiko nach Simulation
- Systemtest im Fahrversuch: Akzeptanztest in der realen Betriebsumgebung
- Fahrzeugtauglicher Zugang zu Steuergeräten und Netzwerken
- Mobile Messtechnik



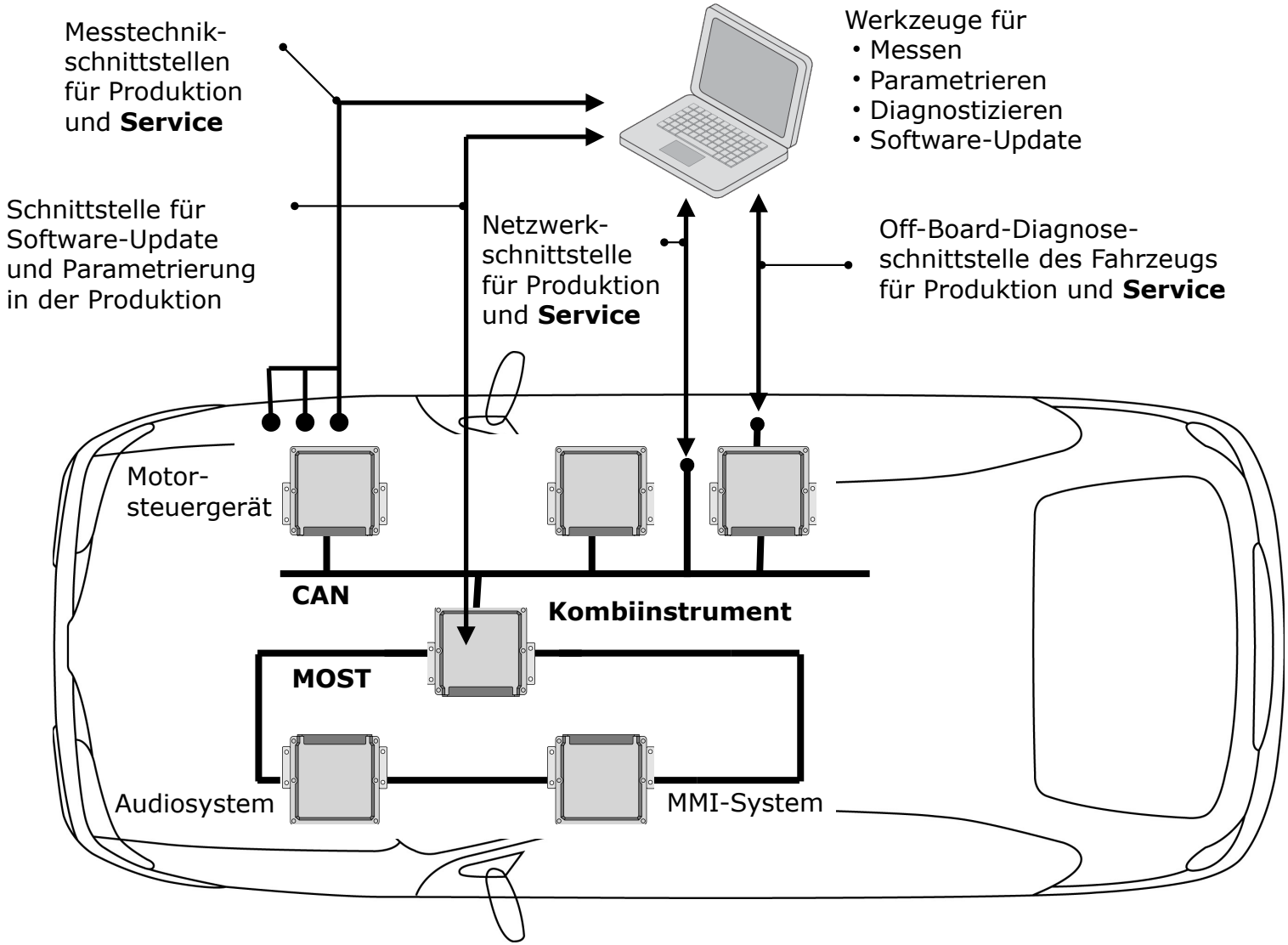
Schnittstellen



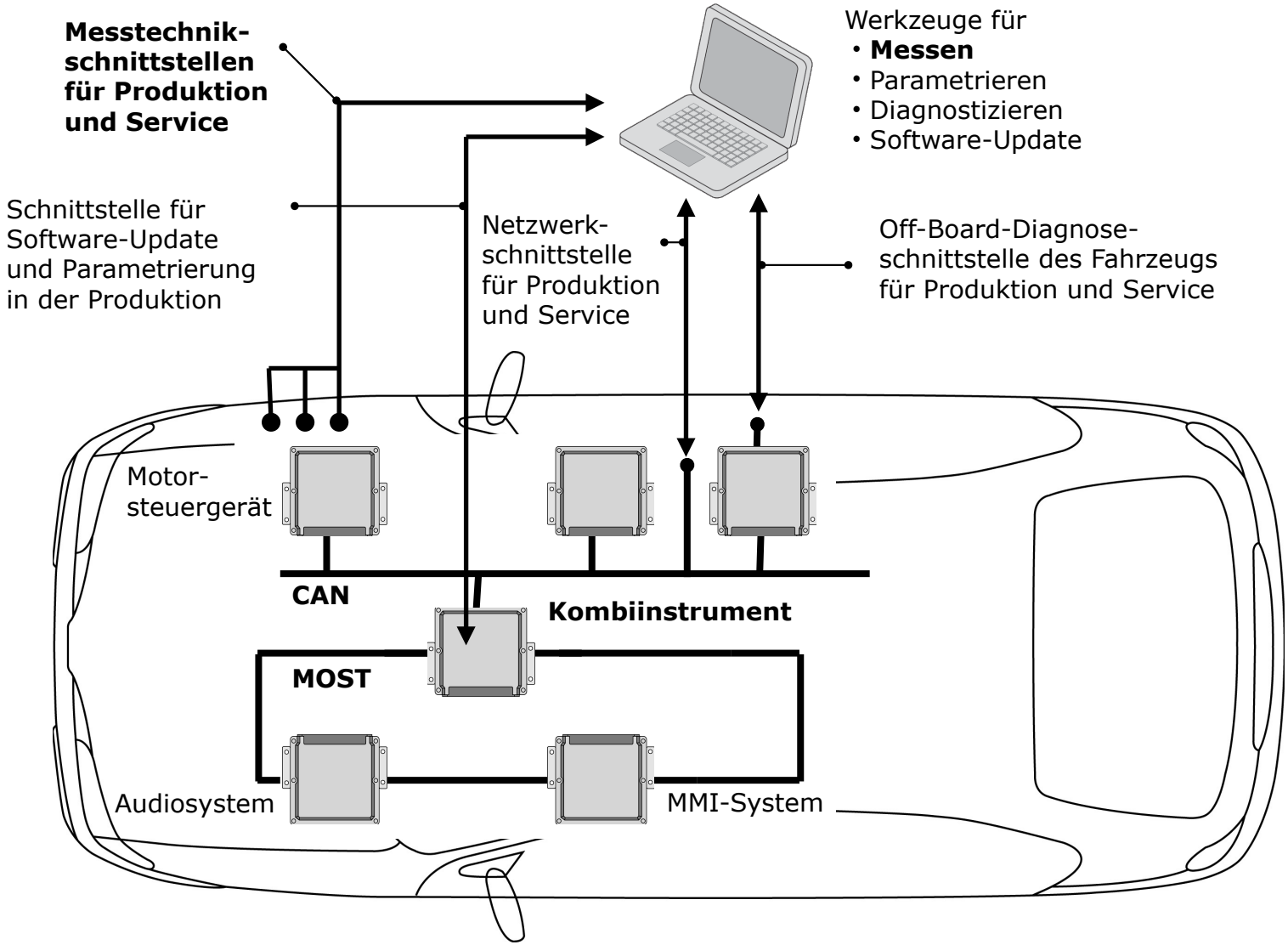
Schnittstellen



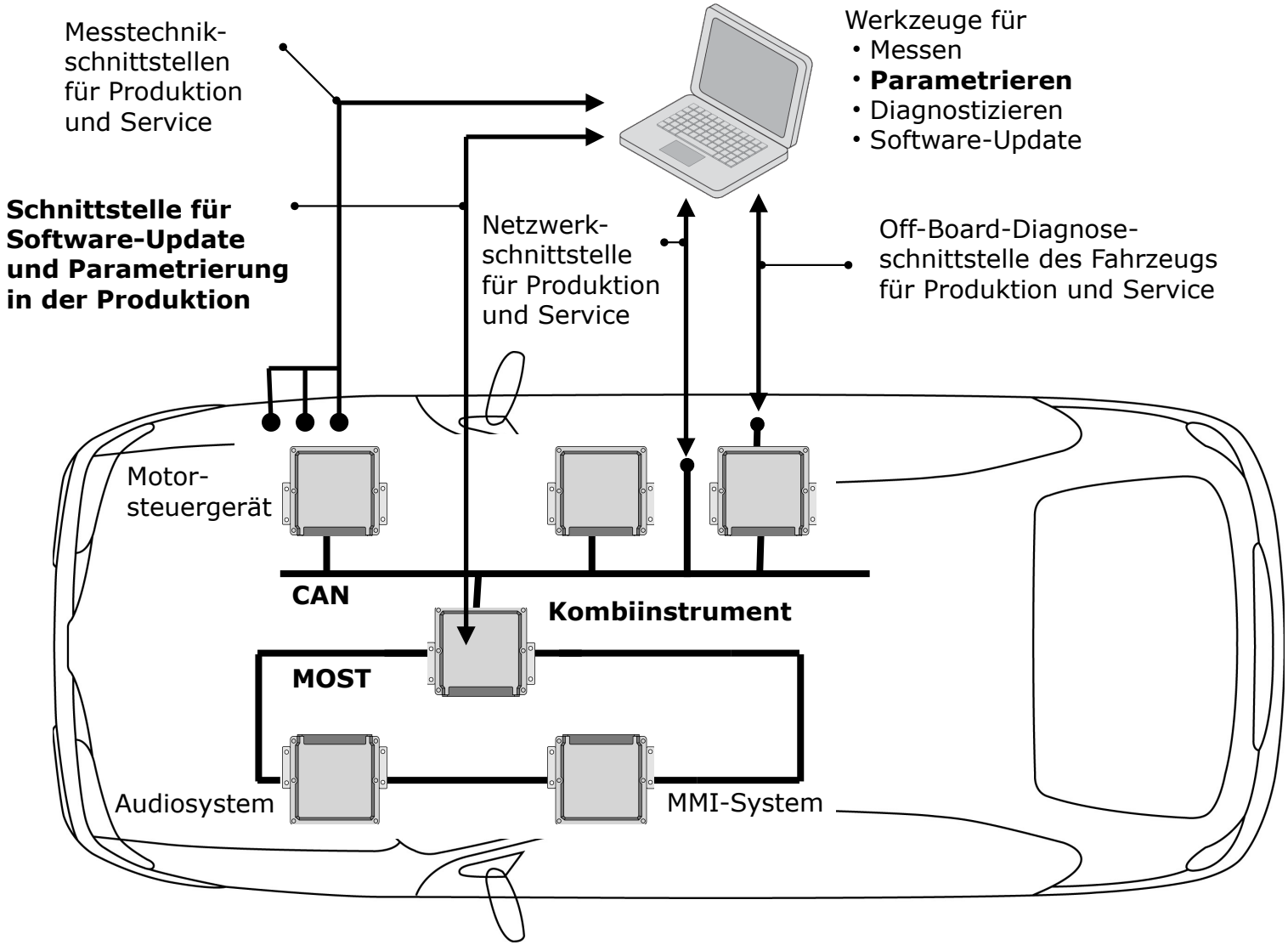
Schnittstellen



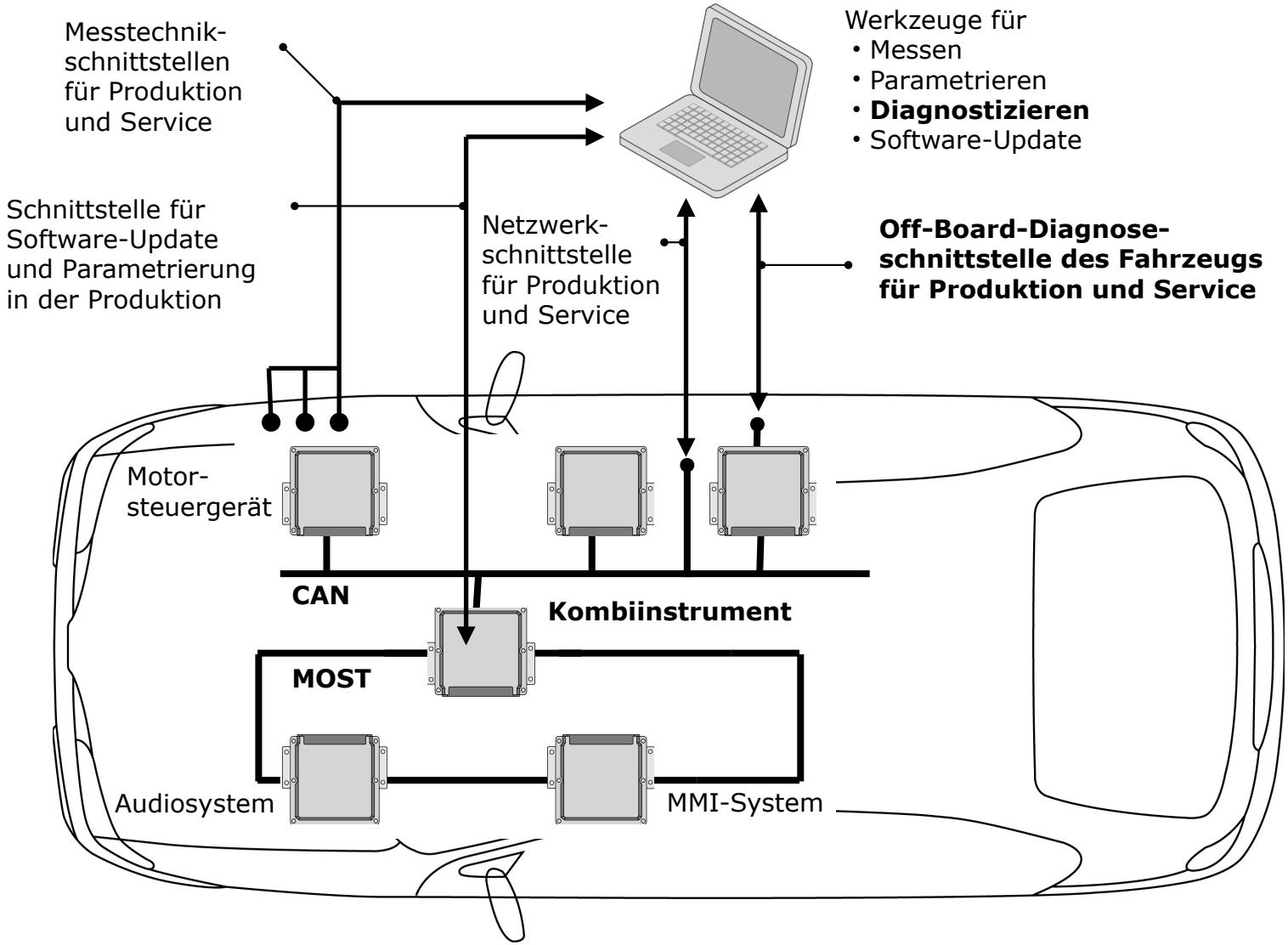
Schnittstellen



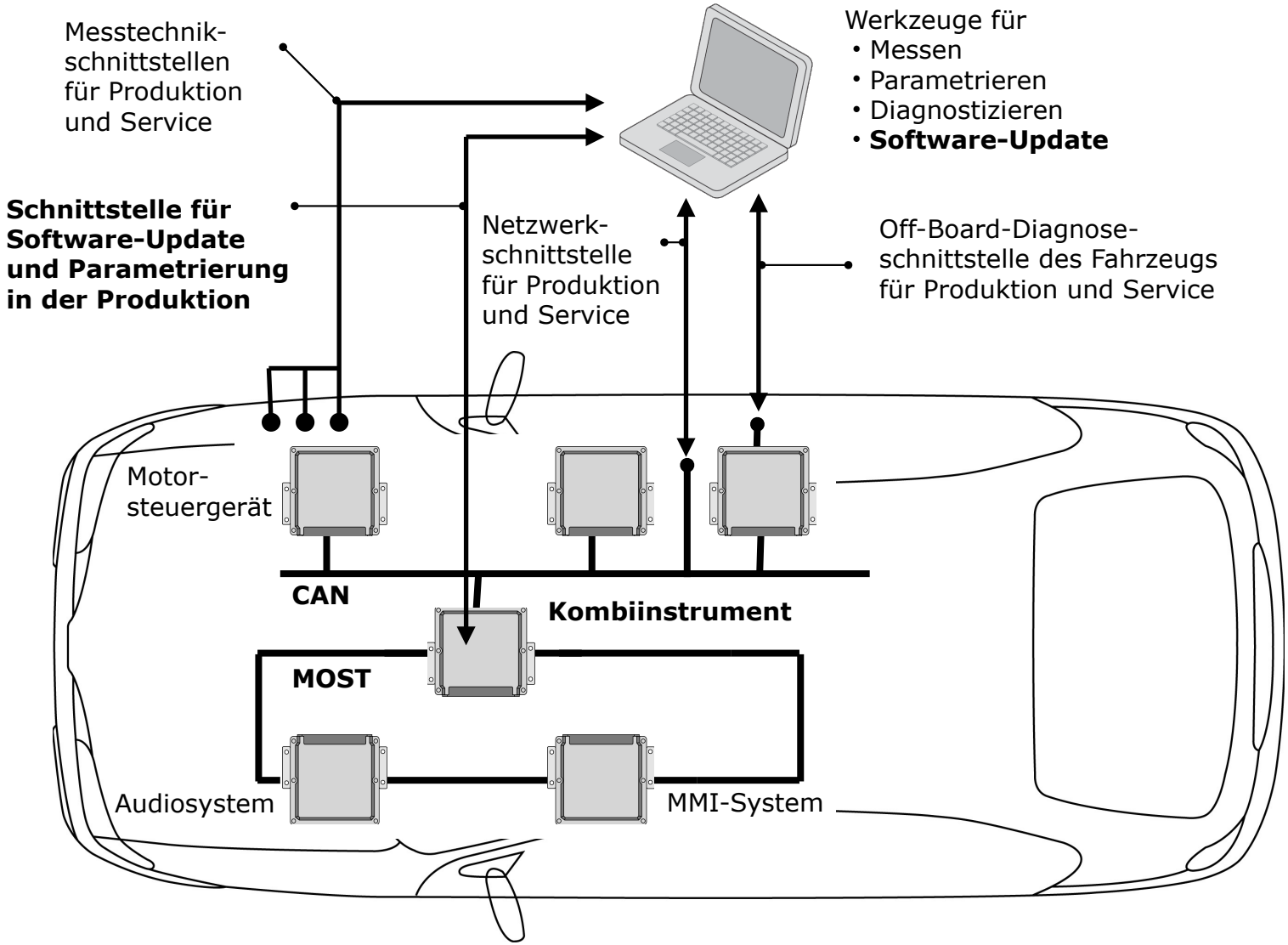
Schnittstellen



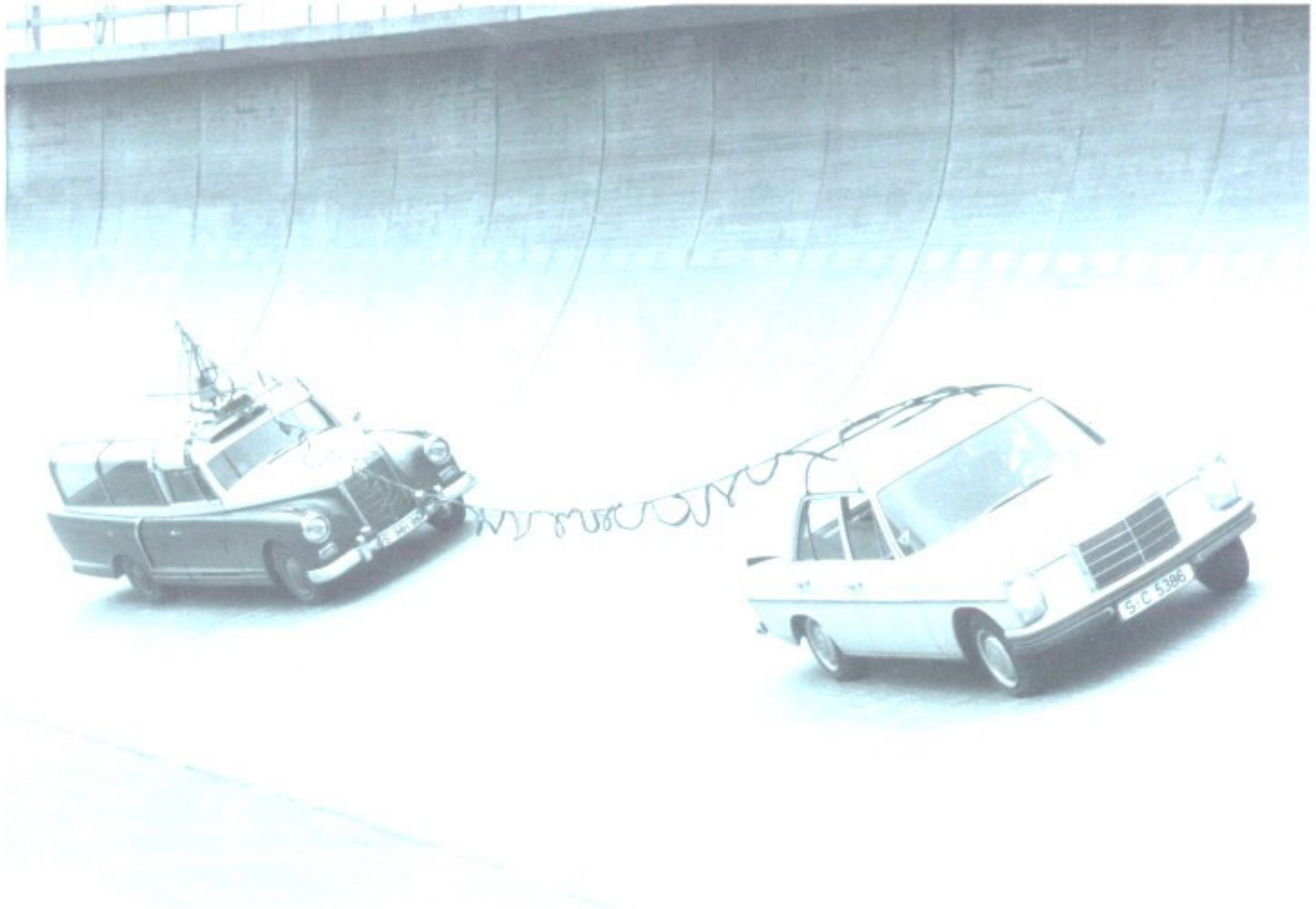
Schnittstellen



Schnittstellen



Mobile Messtechnik



Mobile Messtechnik

Kernprozess - Zusammenfassung

- Kernprozess zur Entwicklung von elektronischen Systemen und Software
- Kein Anspruch auf Vollständigkeit, sondern Darstellung der Vorgehensweise mit Beispielen
- Anpassung des Kernprozesses für konkrete Projekte nötig
  - In der Praxis weniger Prozessschritte im V-Modell
  - Beispiele
    - Motorsteuergeräte
      - Kalibrierung wichtig
      - Vielzahl verschiedener Funktionen
    - Karosserieelektronik
      - Kalibrierung untergeordnet
      - Verteilte und vernetzte Systeme