INFORMATIK ▪ CONSULTING ▪ SYSTEMS AG

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

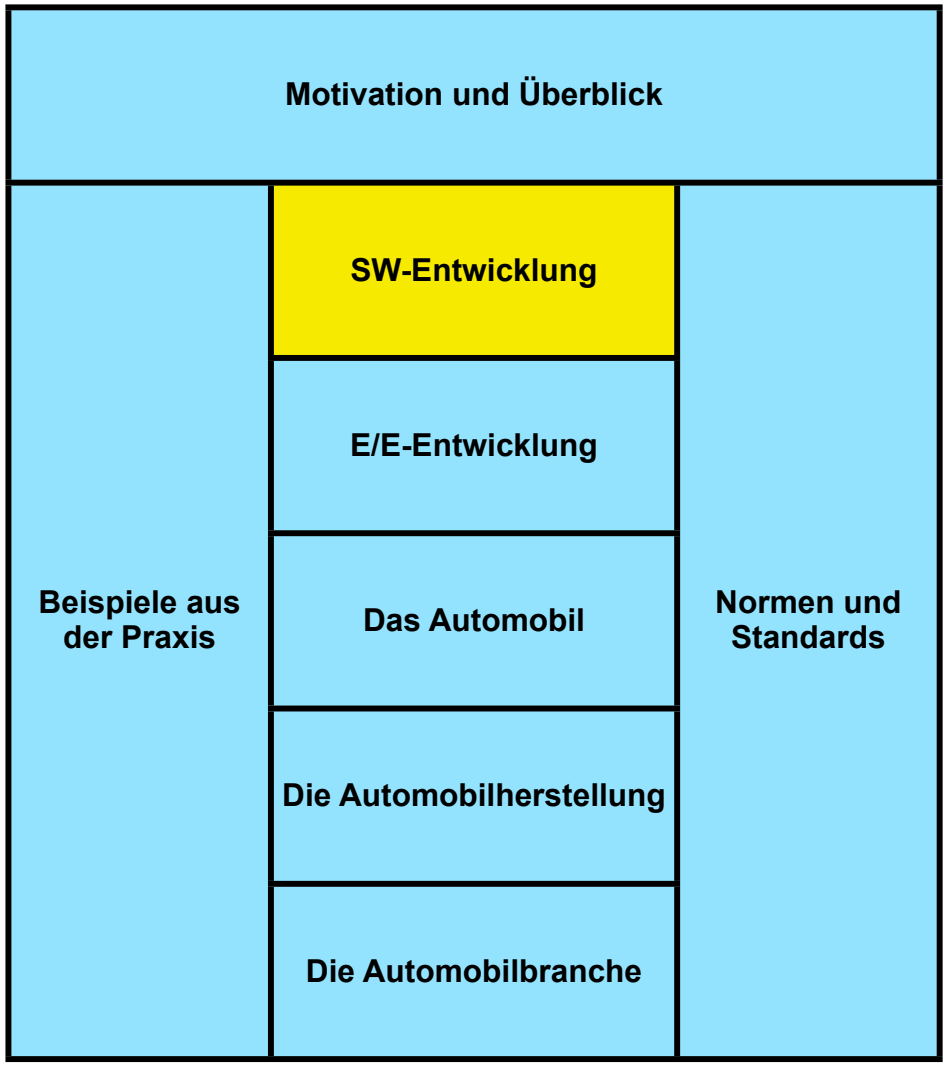
TU Dresden, Fakultät Informatik

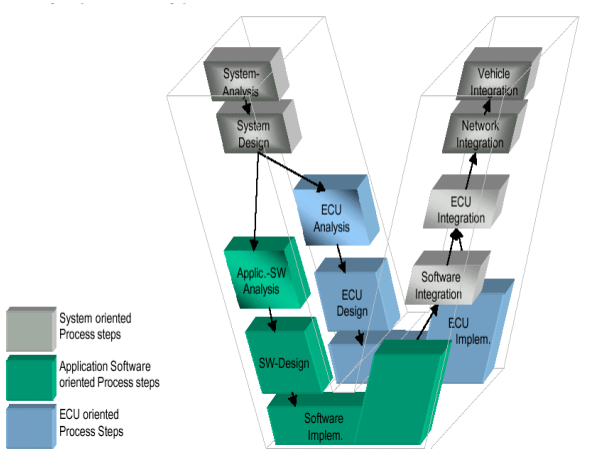
Sommersemester 2012

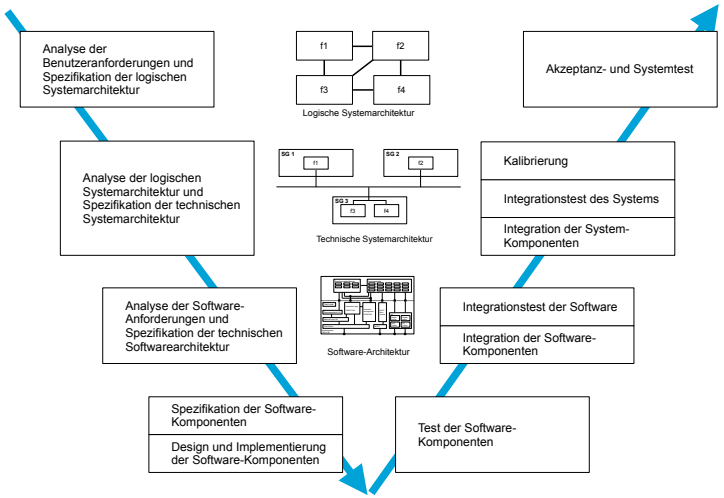
Prof. Dr. rer. nat. Bernhard Hohlfeld

bernhard.hohlfeld@daad-alumni.de

Vorlesung Automotive Software Engineering



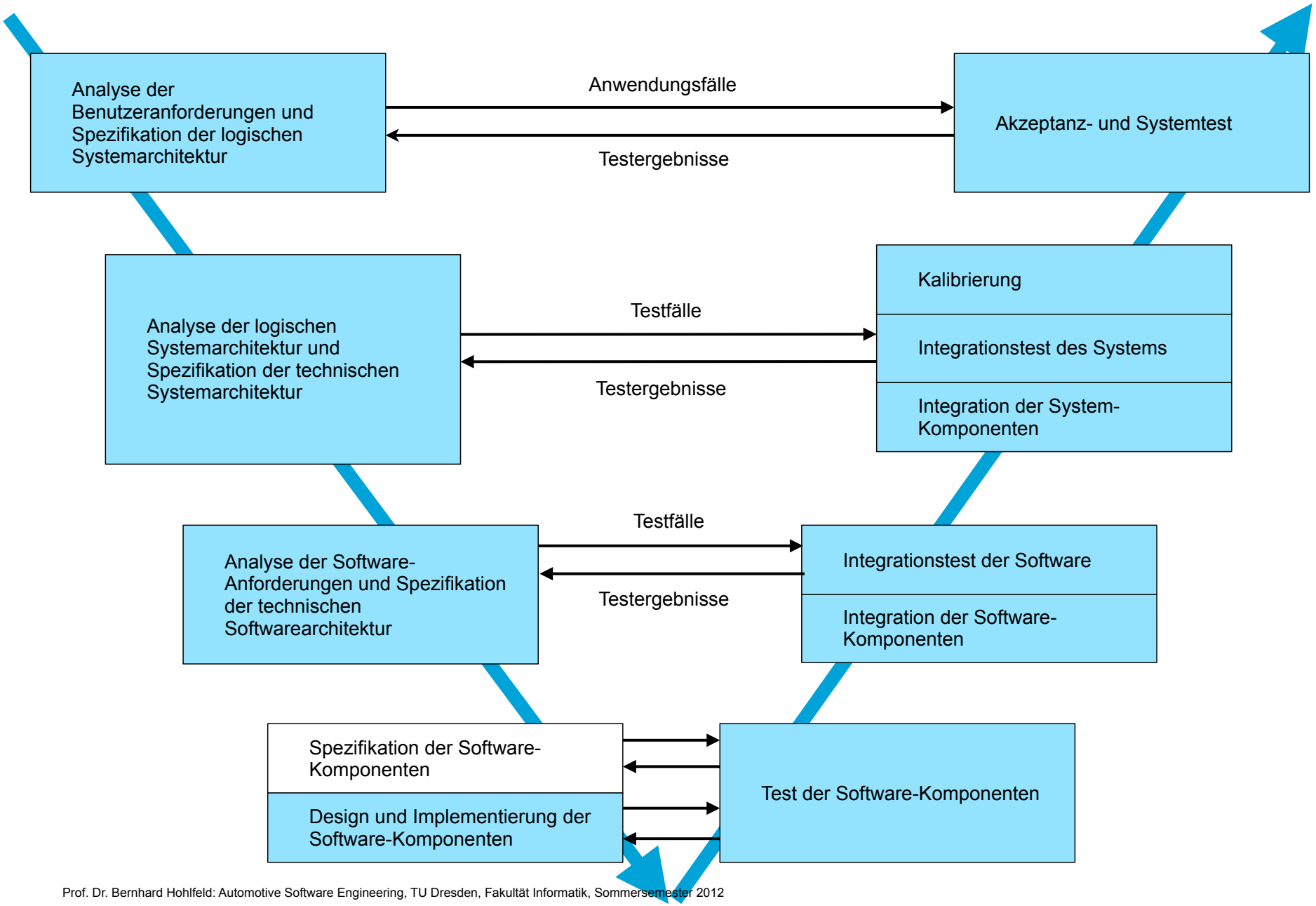




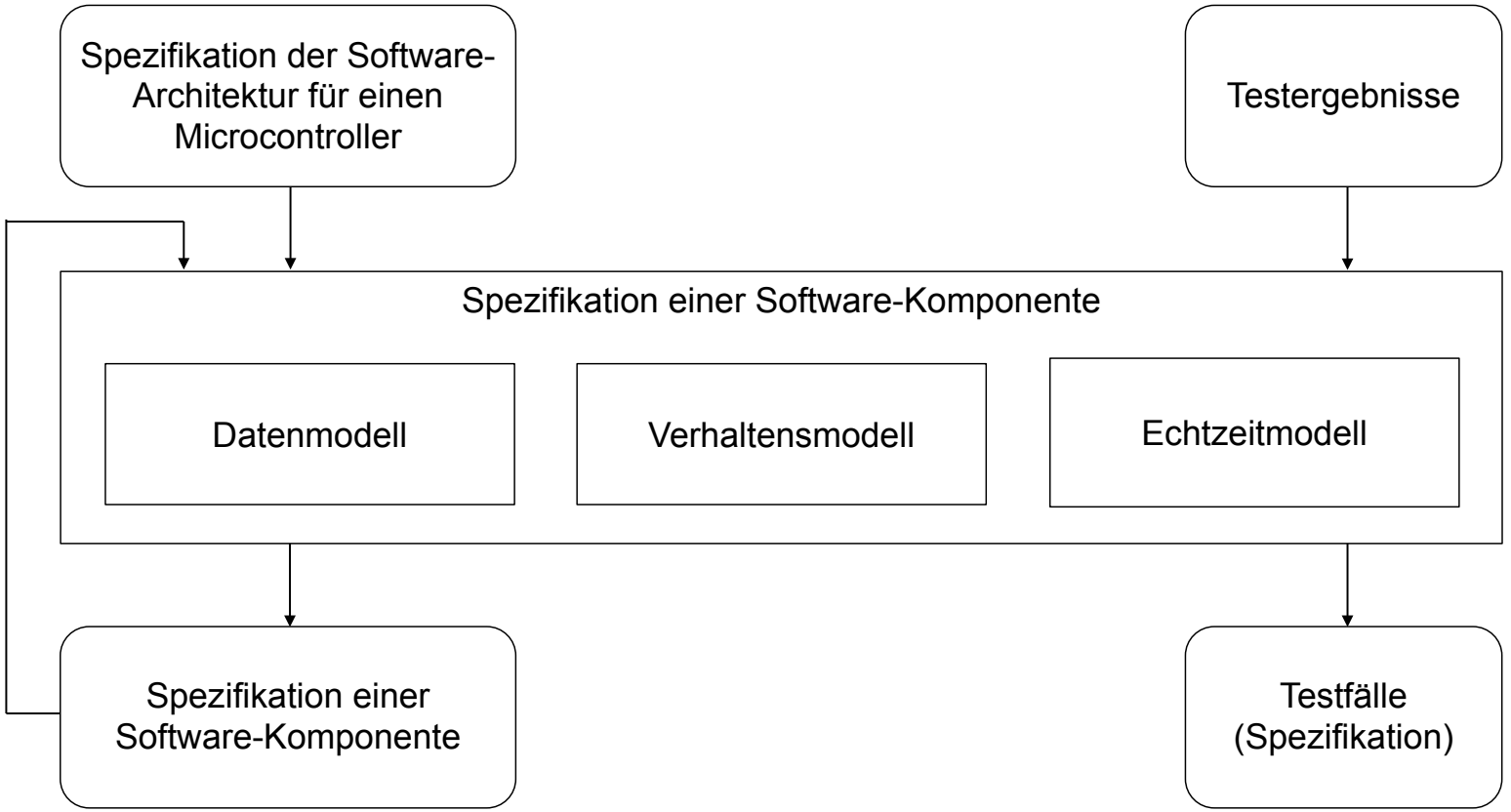
6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

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

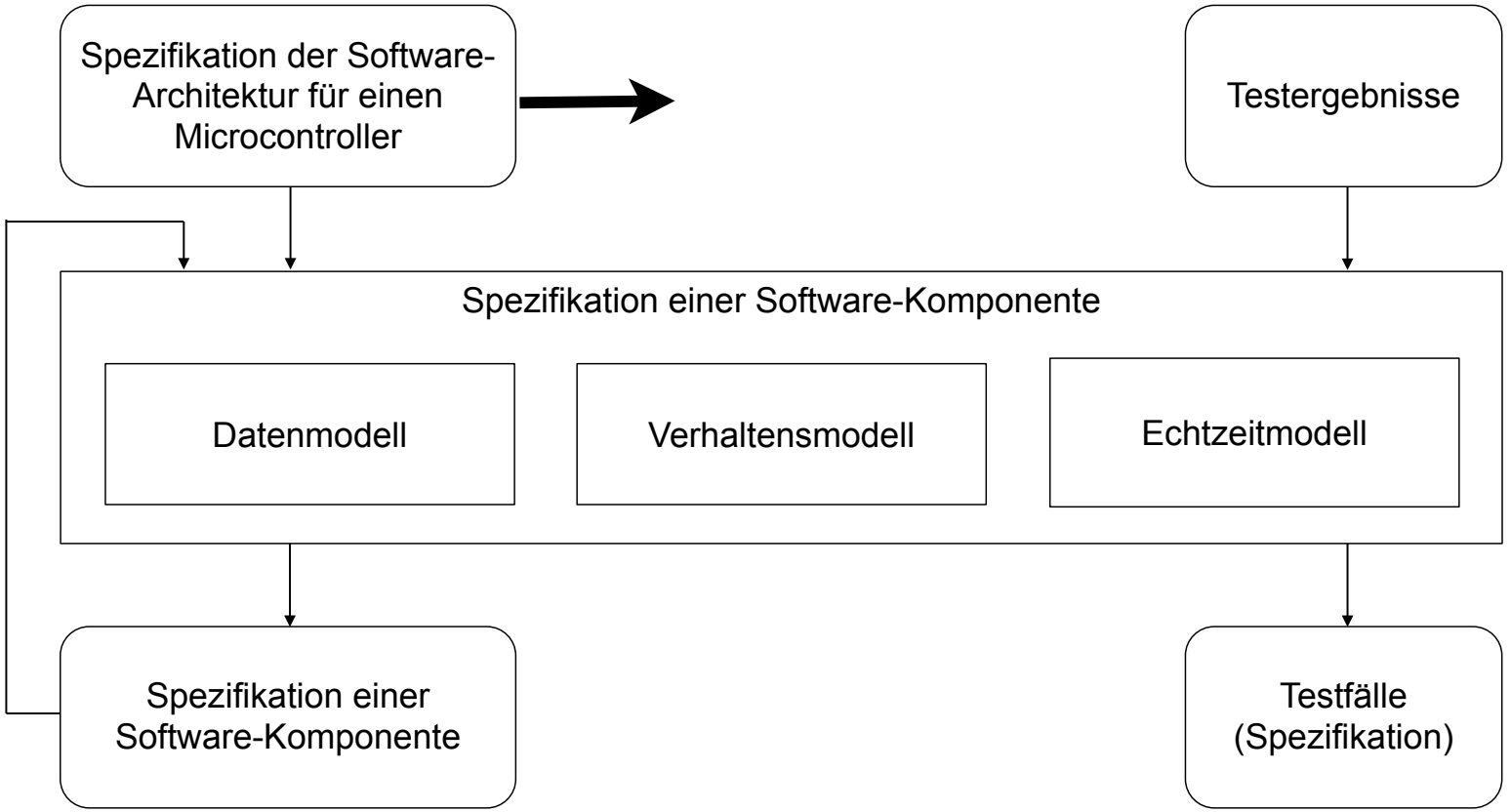
Spezifikation der Software-Komponenten



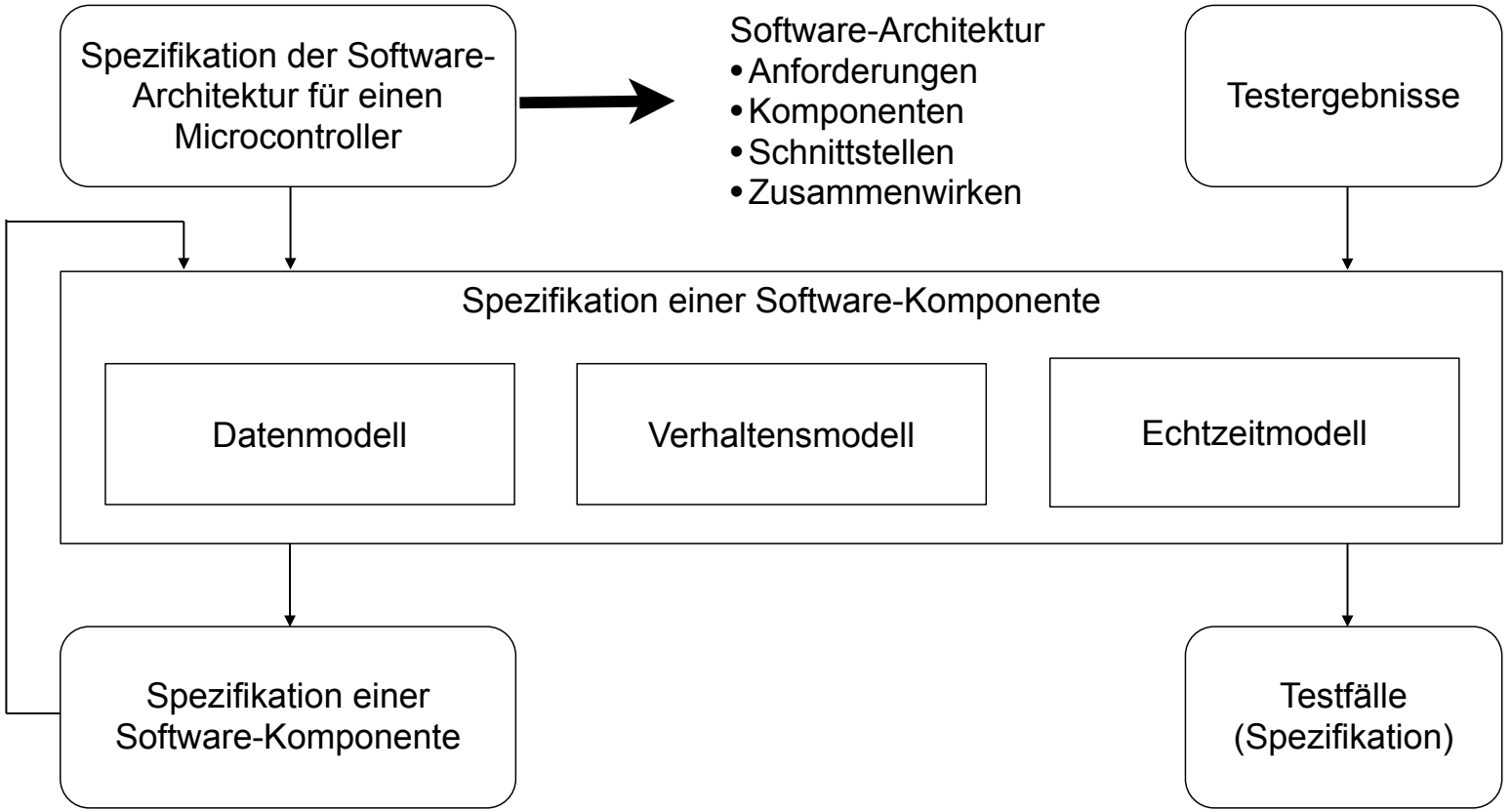
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



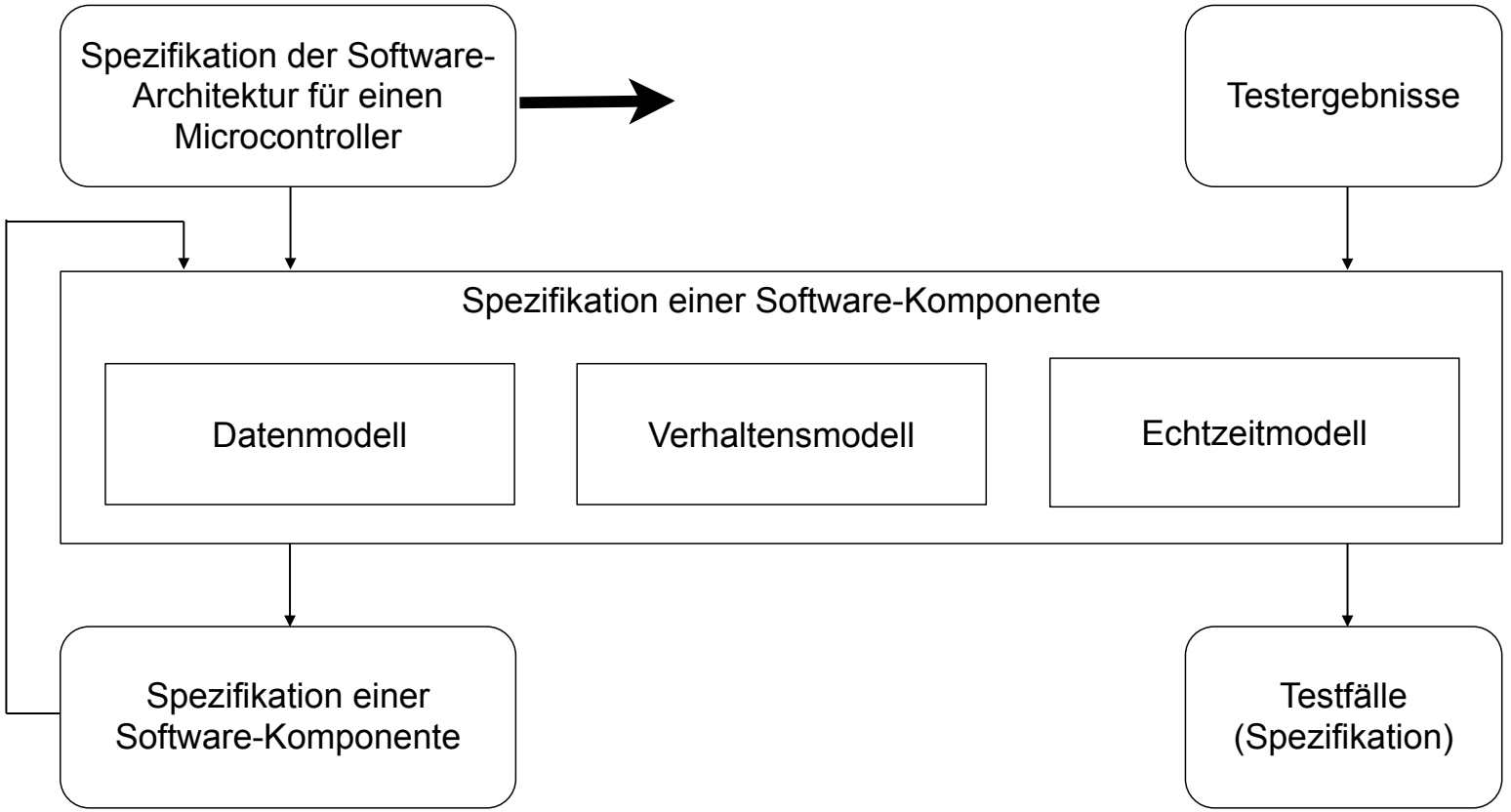
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



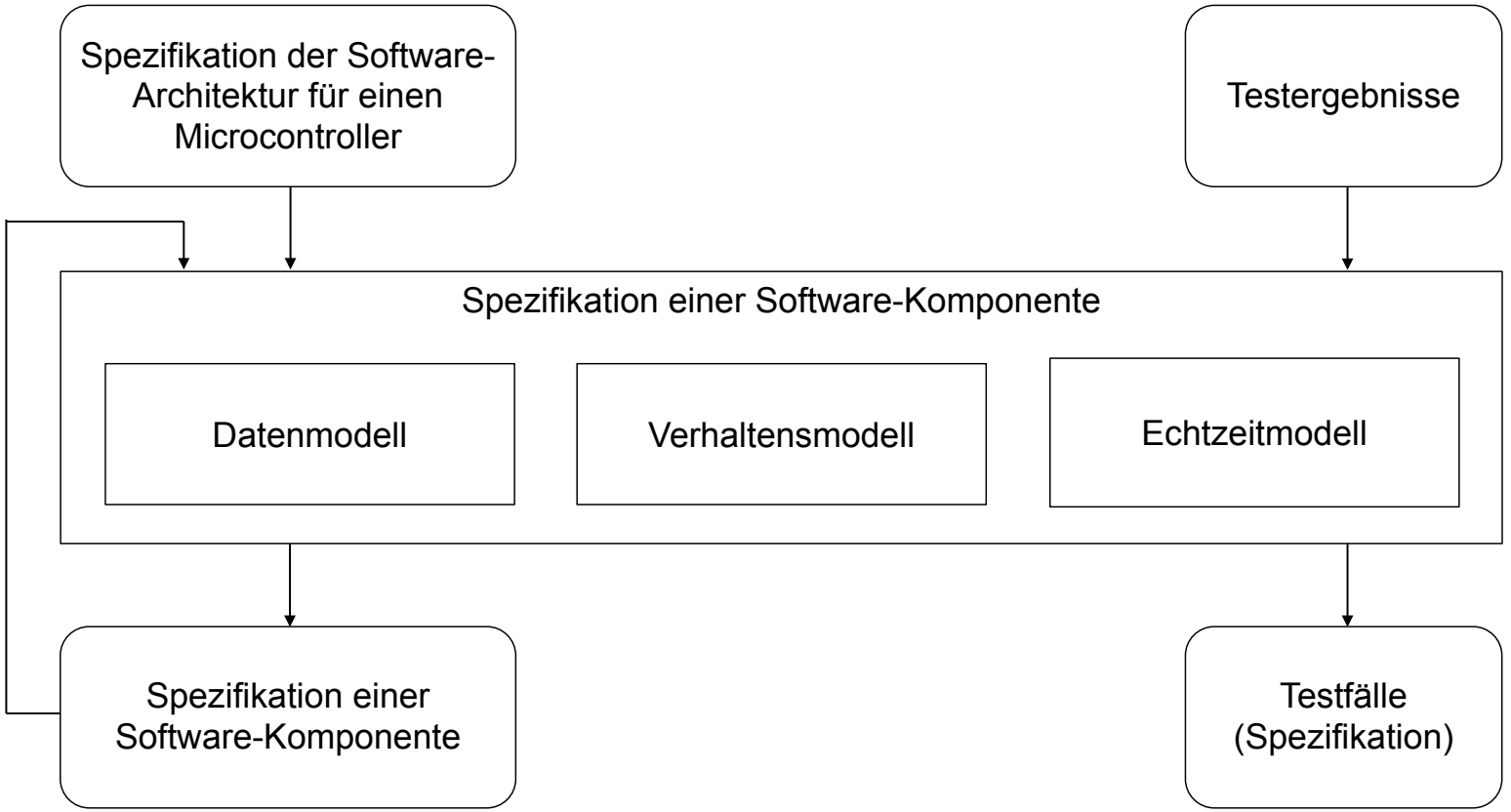
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



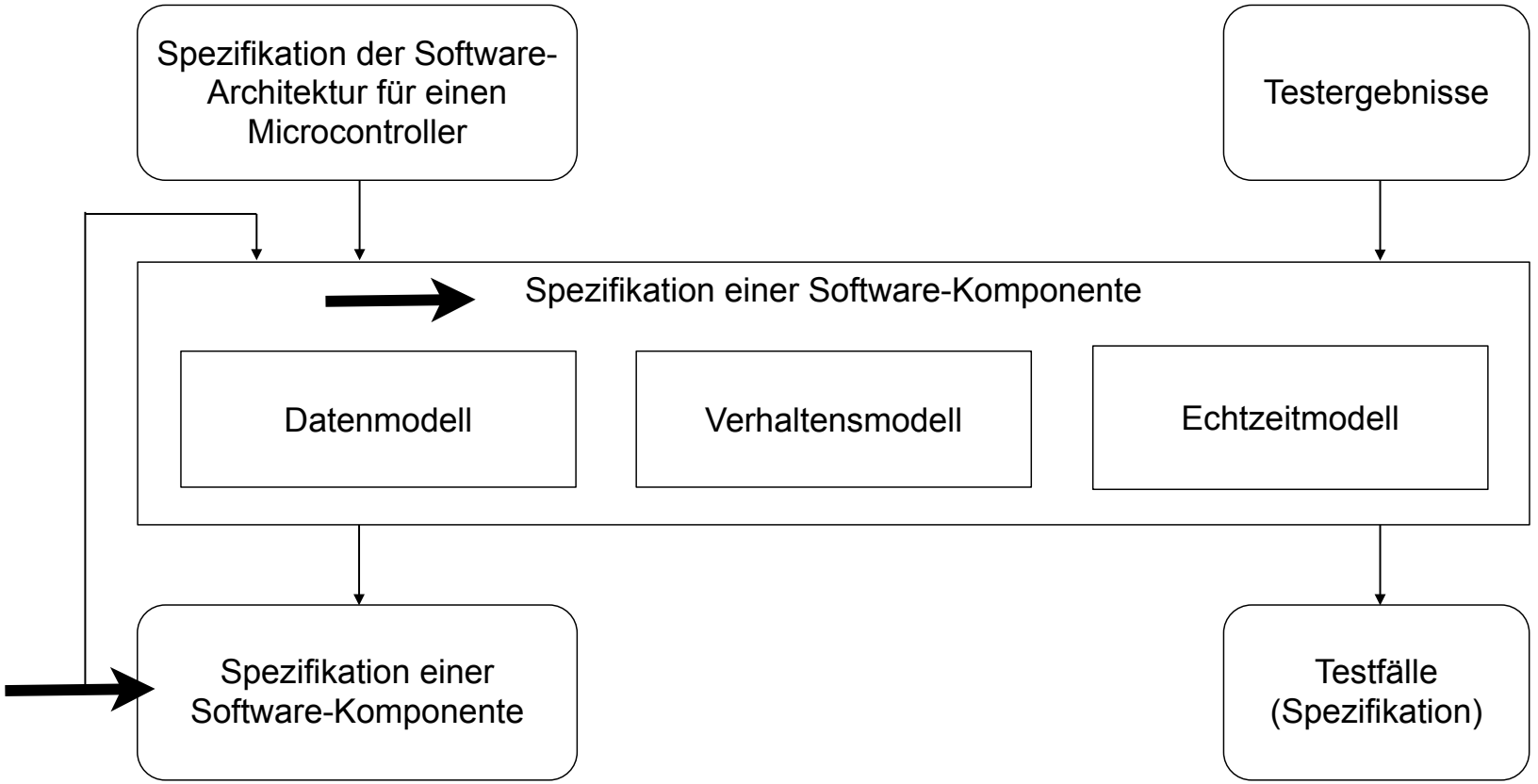
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



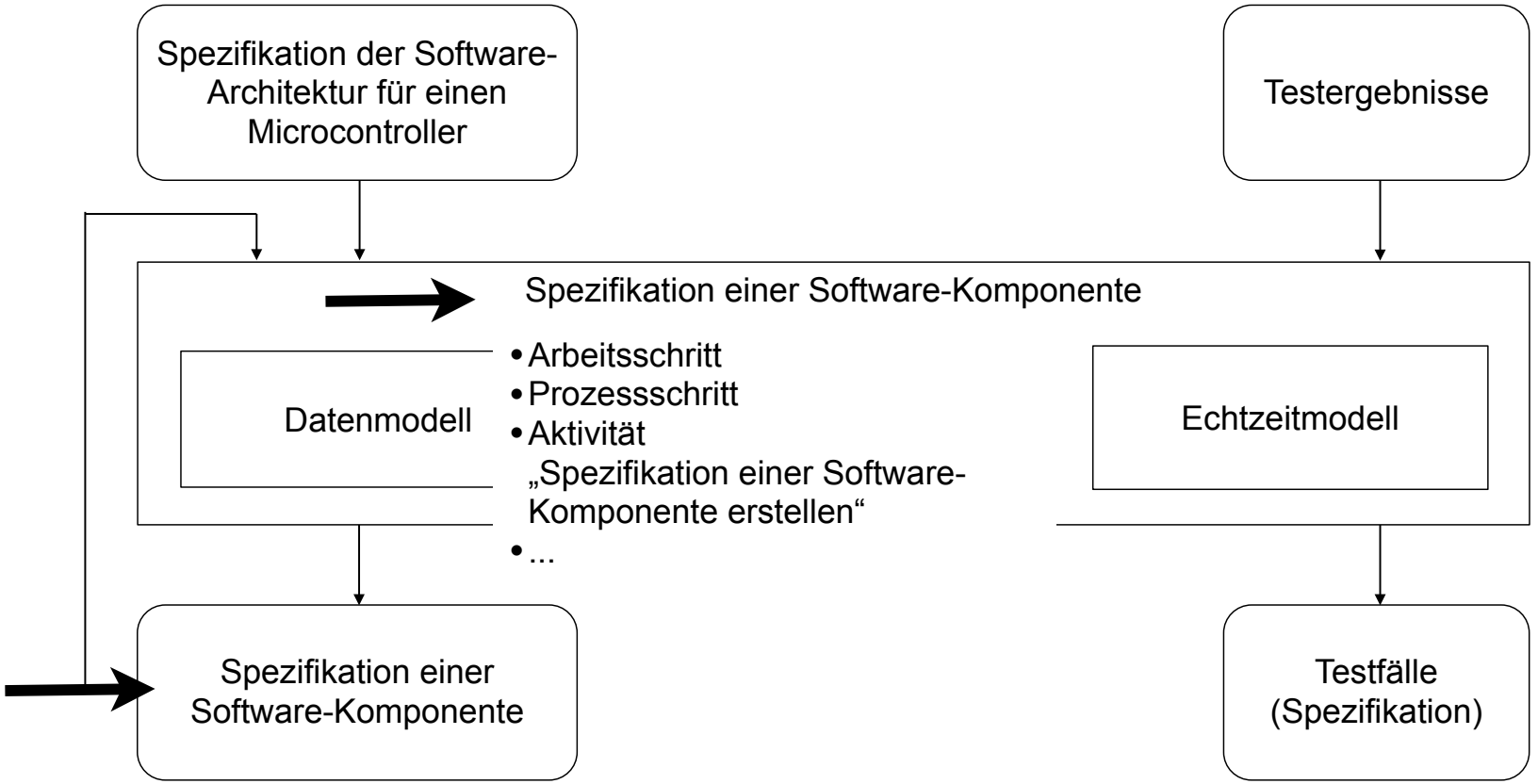
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



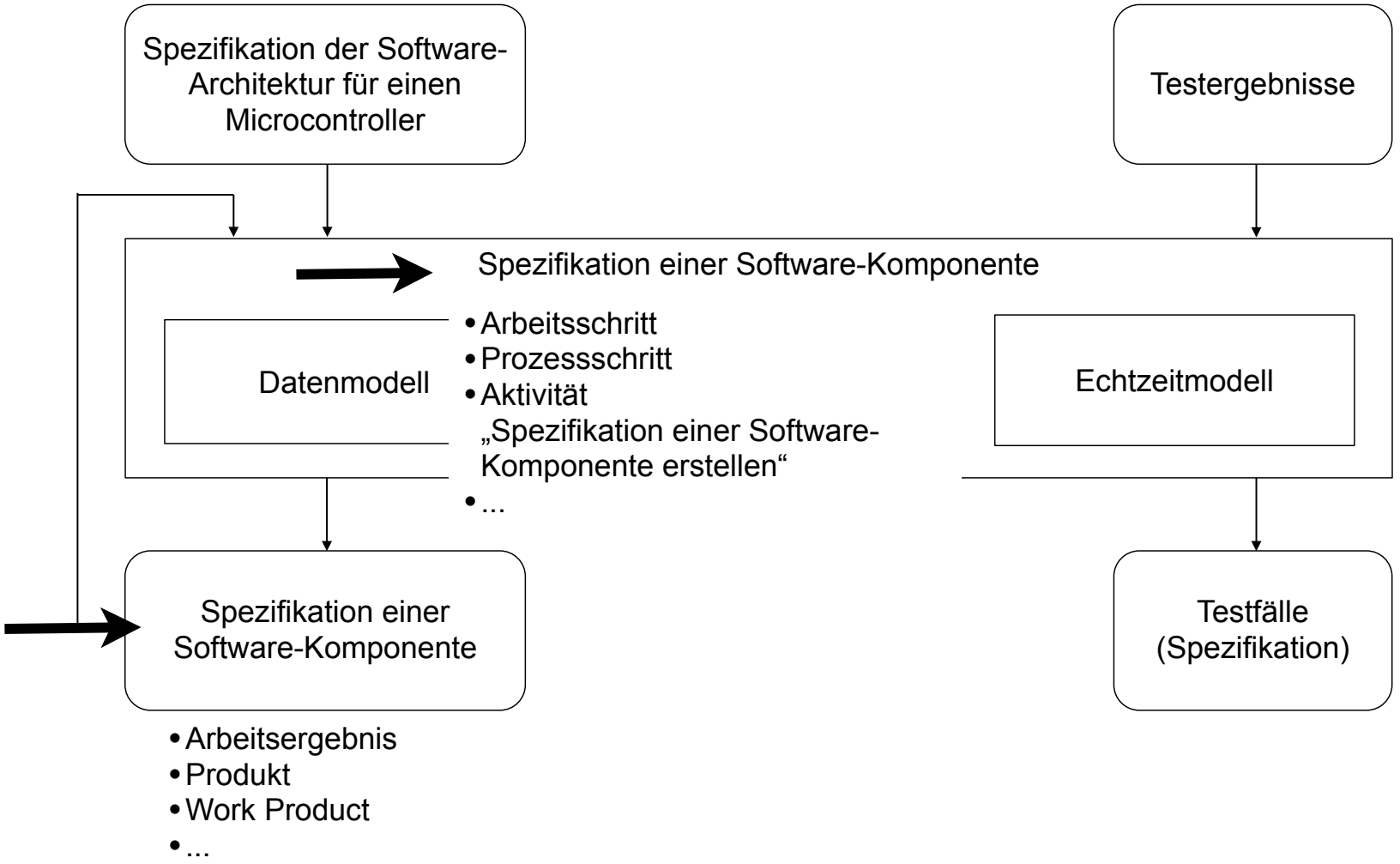
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



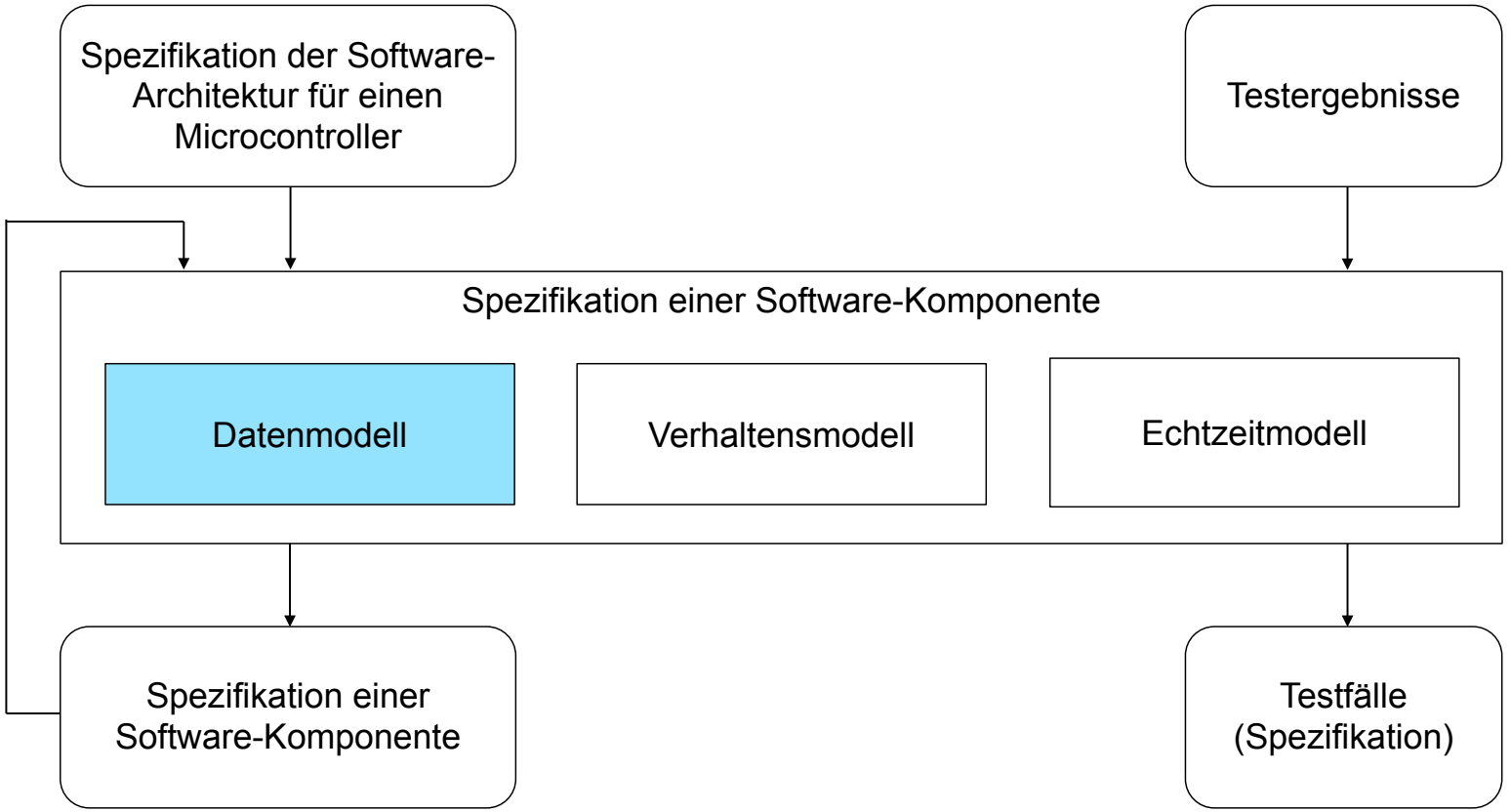
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)

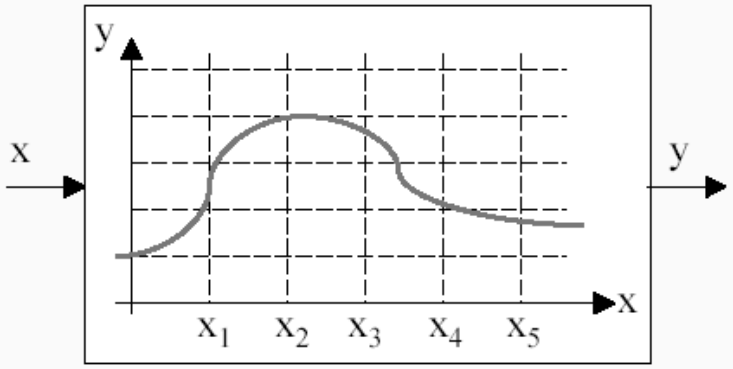


Datenstrukturen

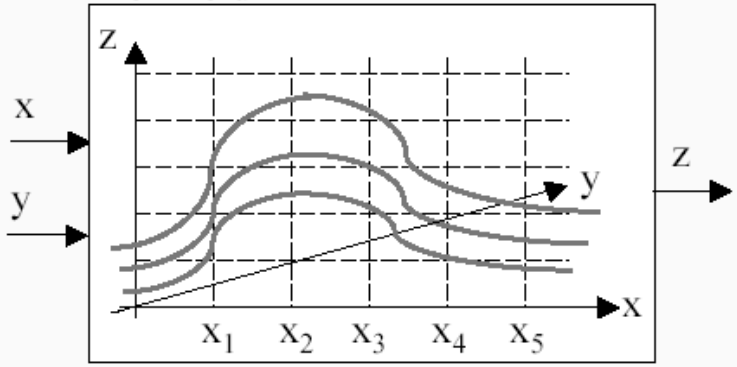
- Einfache Datenstrukturen
  (skalare Größen, Vektoren, Matritzen)
- komplexere Datenstrukturen

Grafische Darstellung:

Kennlinie



Kennfeld



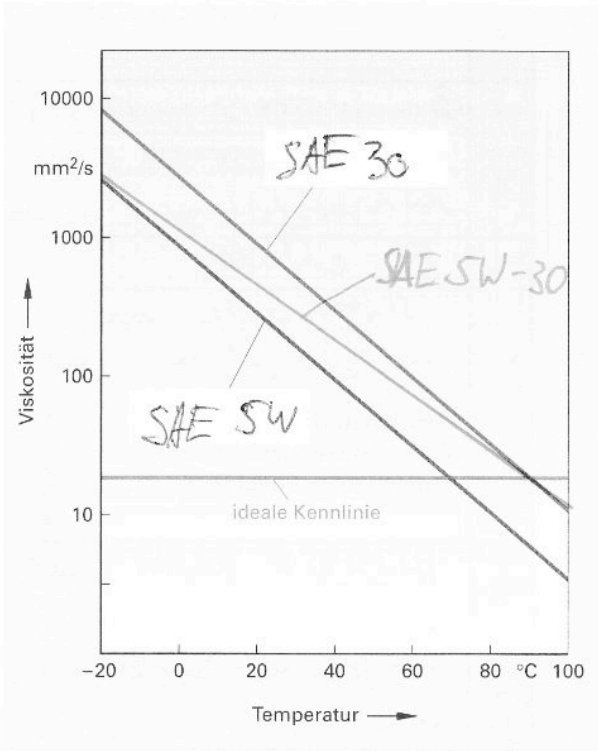
Tabellarische Darstellung:

x_1	x_2	x_3	x_4	x_5
y_1	y_2	y_3	y_4	y_5

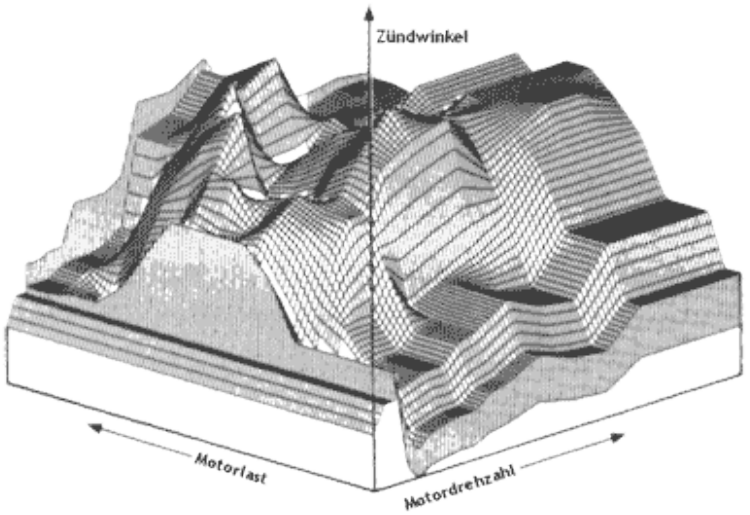
	x_1	x_2	x_3	x_4	x_5
y_1	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
y_2	z_{21}	z_{22}	z_{23}	z_{24}	z_{25}
y_3	z_{31}	z_{32}	z_{33}	z_{34}	z_{35}

Beispiele

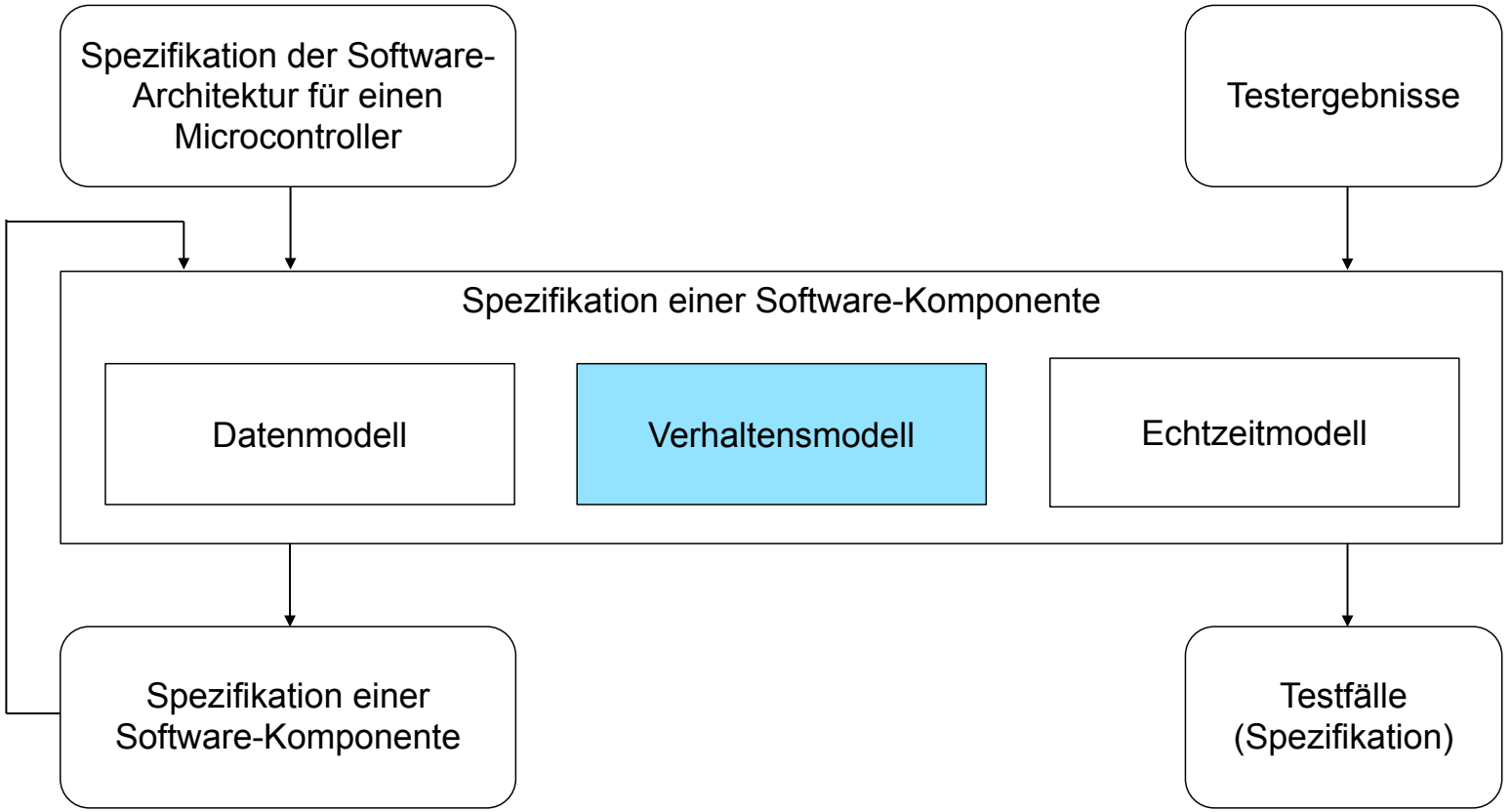
- Kennlinie
  - 2-dimensional
  - Geschwindigkeit y in Abhängigkeit von Motordrehzahl x bei konstantem Gang
  - Viskosität in Abhängigkeit von Temperatur



- 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



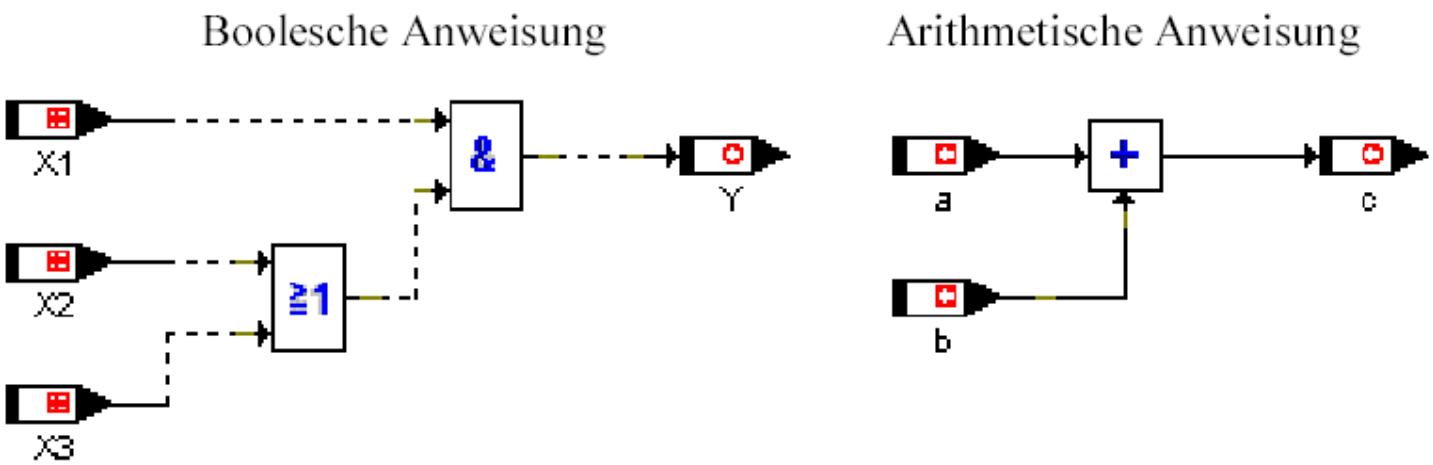
Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)



Spezifikation des Datenflusses

Beispiel: Datenfluss für eine boolsche und eine arithmetische Anweisung in ASCET-SD-Blockschaltbildern

y= X1 & (X2 || X3)
c = a+b



Erläuterung

- Y = X1 & (X2 | | X3)
- &: Logisches „und“

&	0	1
0	0	0
1	0	1

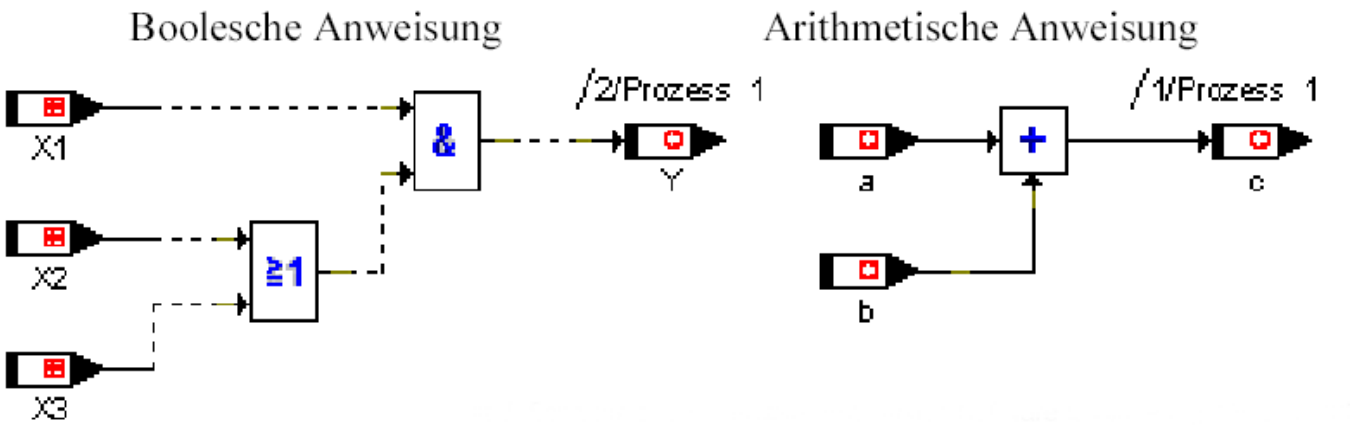
- | |: Exclusives „oder“

		0	1
0	0	1
1	1	0

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

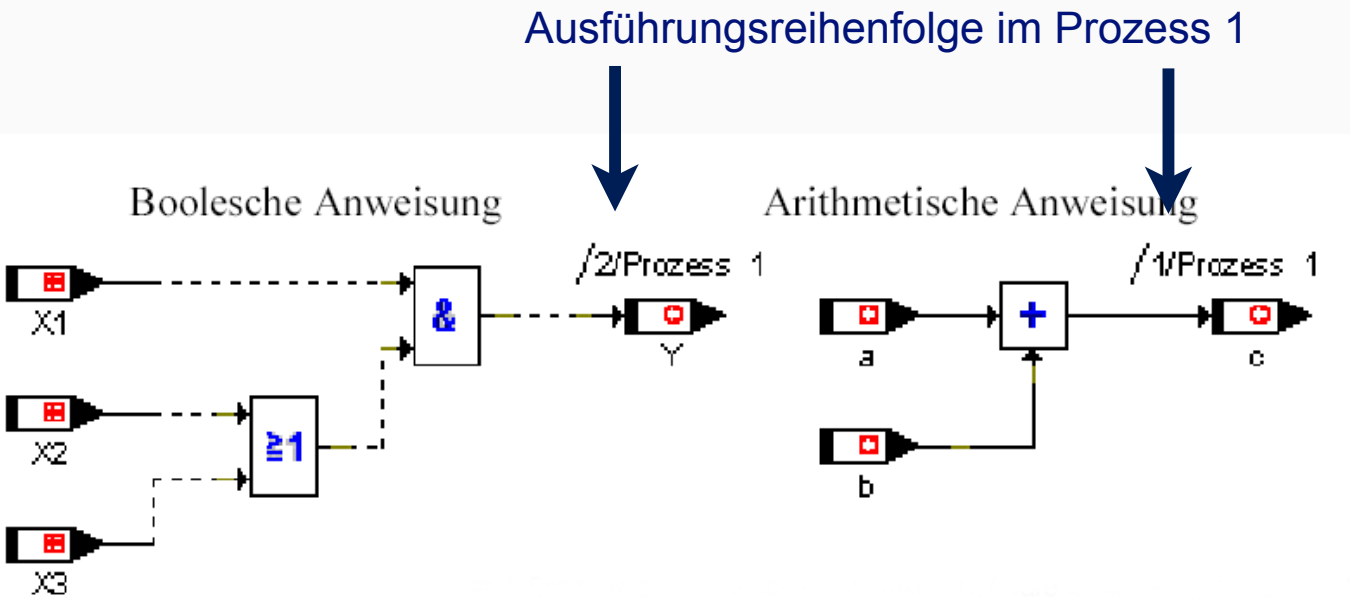
Kontrollfluss zur Festlegung der Ausführungsreihenfolge in ASCET-SD



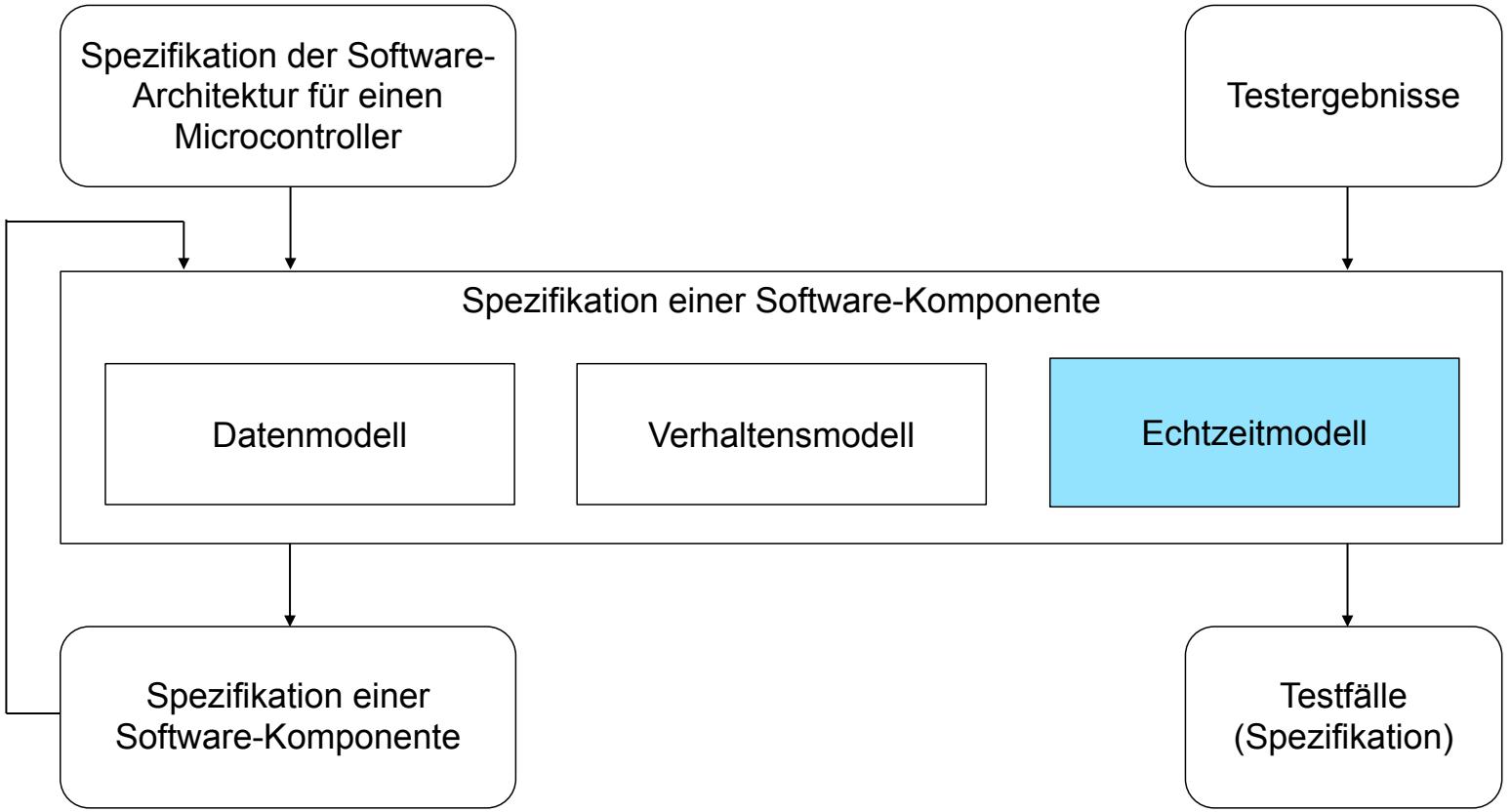
Spezifikation des Kontrollflusses

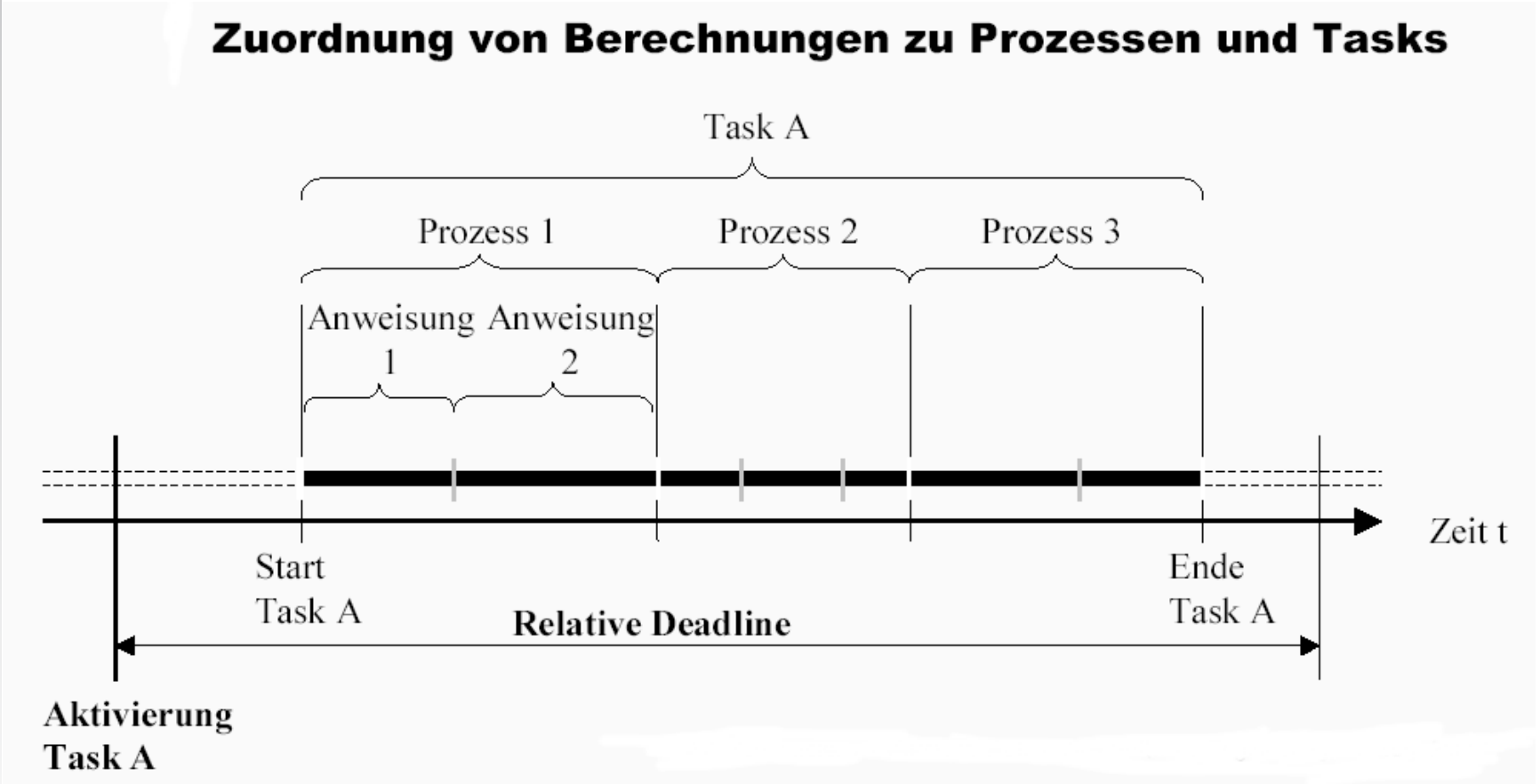
- klassische Ansätze aus der Programmierung, z.B. Struktogramme
- Ansätze aus dem objekt-orientierten Entwurf, z.B. für die Aufrufstruktur zwischen Software-Komponenten

Kontrollfluss zur Festlegung der Ausführungsreihenfolge in ASCET-SD



Spezifikation der Software-Komponenten (Nach Schäuffele, Zurawka)

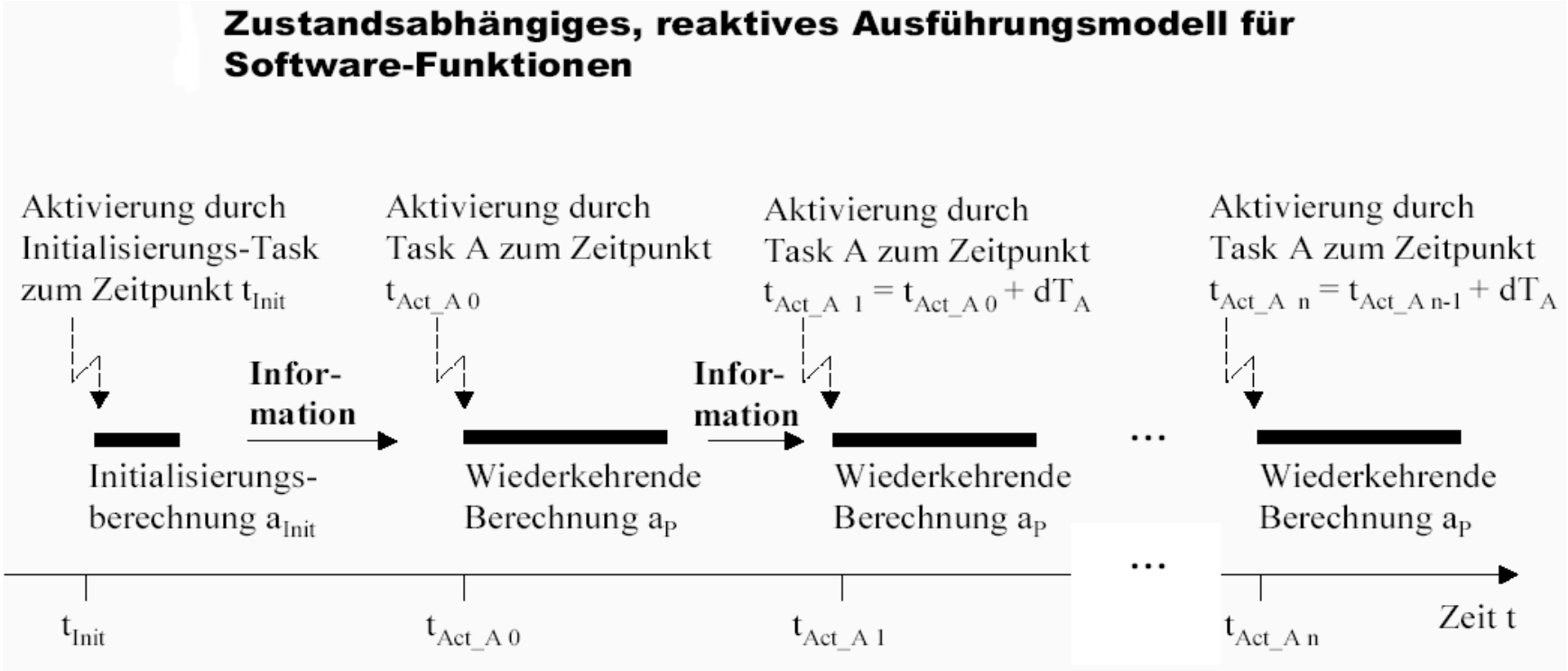



Zuordnung von Berechnungen zu Prozessen und Tasks
Task A
Prozess 1
Prozess 2
Prozess 3
Anweisung 1
Anweisung 2
Zeit t
Start Task A
Ende Task A
Relative Deadline
Aktivierung Task A

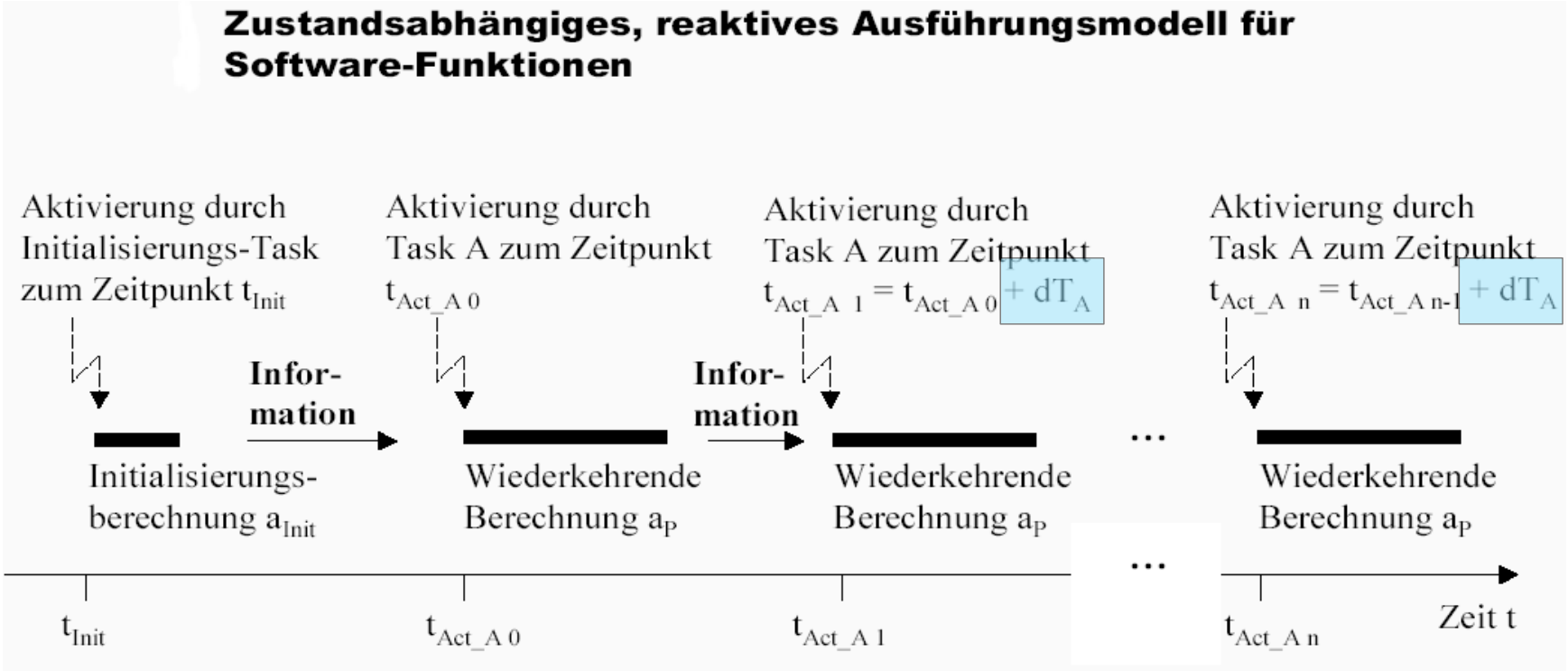
Ausführungsmodelle

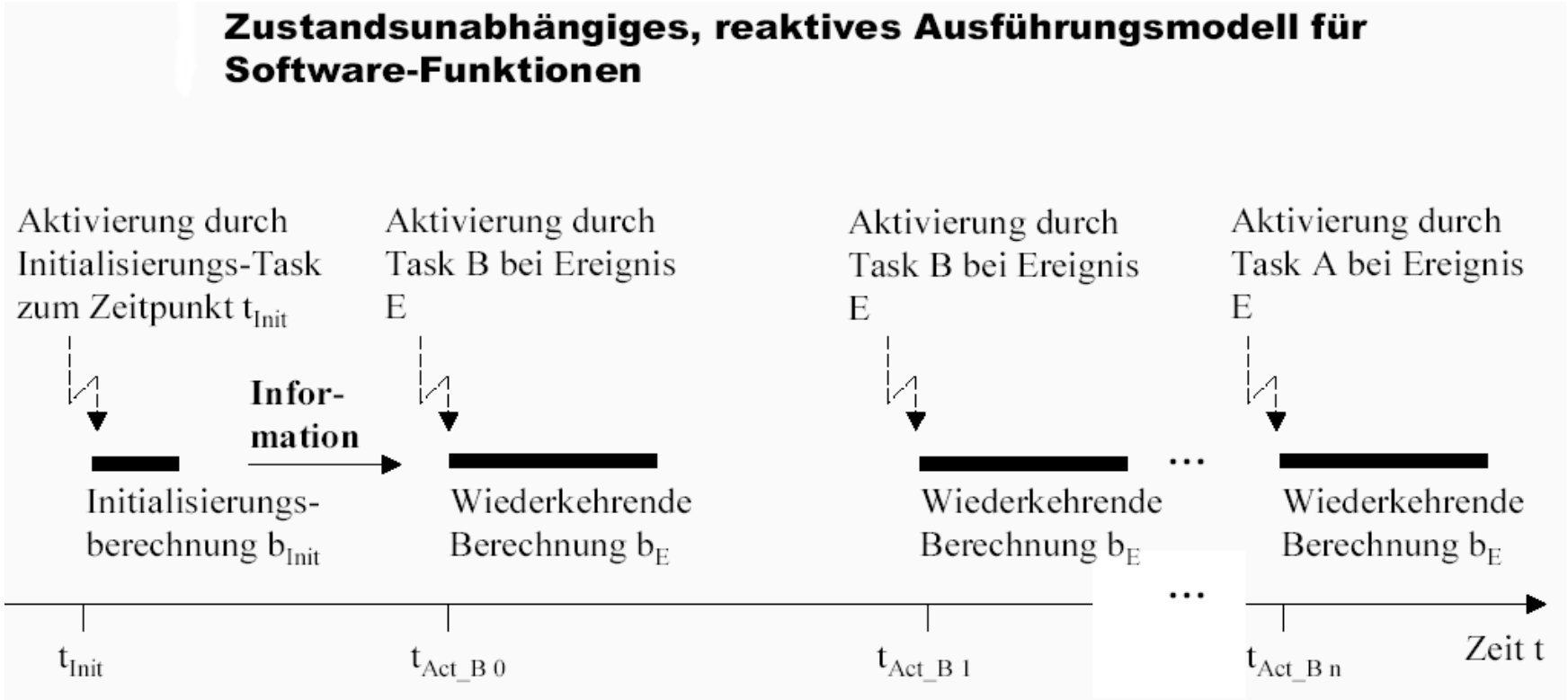
- Zeitabhängige Ausführung („Gedächtnis“):
  Zustandsabhängiges, reaktives Ausführungsmodell
- Ereignisabhängige Ausführung:
  Zustandsunabhängiges, reaktives Ausführungsmodell
- In der Praxis Mischformen

Spezifikation der Software-Komponenten: Zeitabhängige Ausführung („Gedächtnis“)

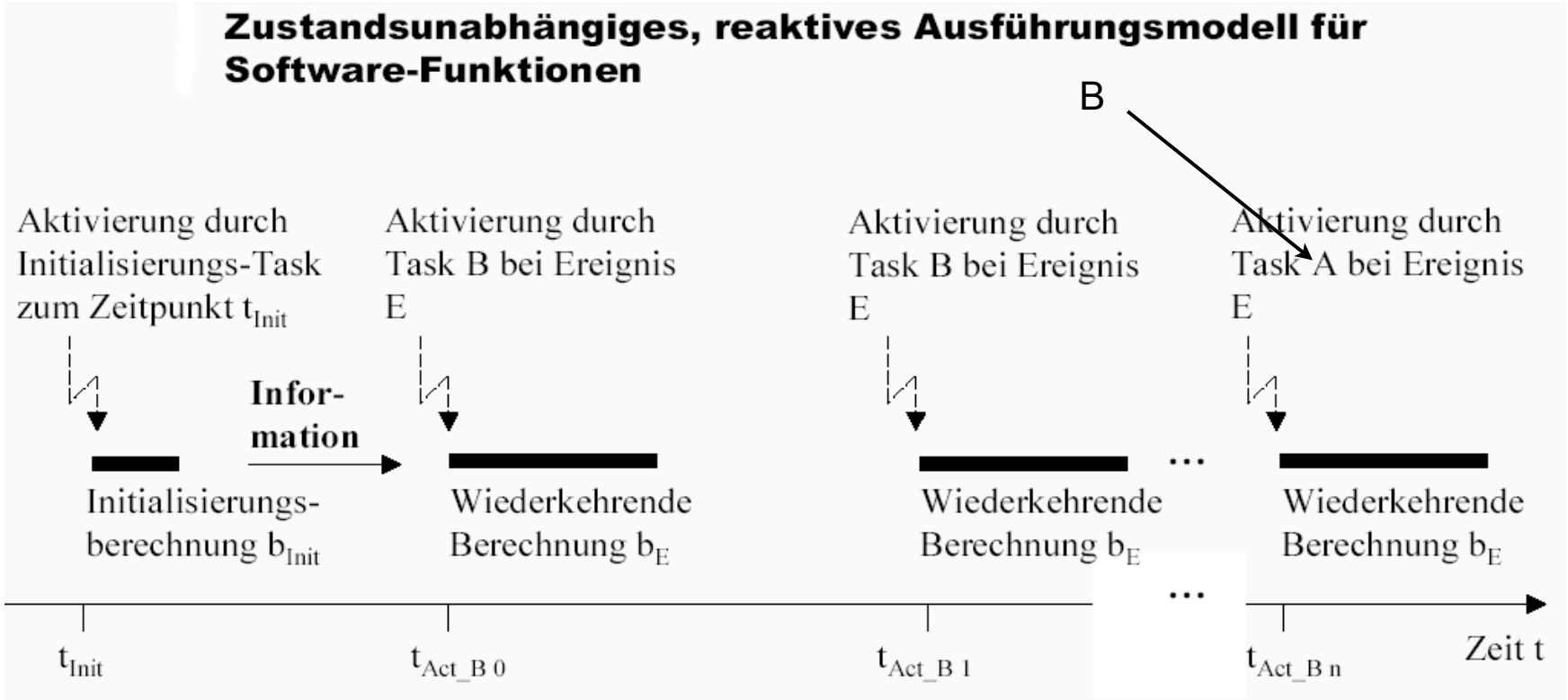


Spezifikation der Software-Komponenten: Zeitabhängige Ausführung




Zustandsunabhängiges, reaktives Ausführungsmodell für Software-Funktionen
Aktivierung durch Initialisierungs-Task zum Zeitpunkt t_{Init}
Aktivierung durch Task B bei Ereignis E
Aktivierung durch Task B bei Ereignis E
Aktivierung durch Task A bei Ereignis E
Infor-mation
Initialisierungs-berechnung b_{Init}
Wiederkehrende Berechnung b_E
Wiederkehrende Berechnung b_E
Wiederkehrende Berechnung b_E
t_{Init}
$t_{Act\_B\ 0}$
$t_{Act\_B\ 1}$
$t_{Act\_B\ n}$
Zeit t

Spezifikation der Software-Komponenten
Ereignisabhängige Ausführung
(Nach Schäuffele, Zurawka)



Ausführungsmodelle

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

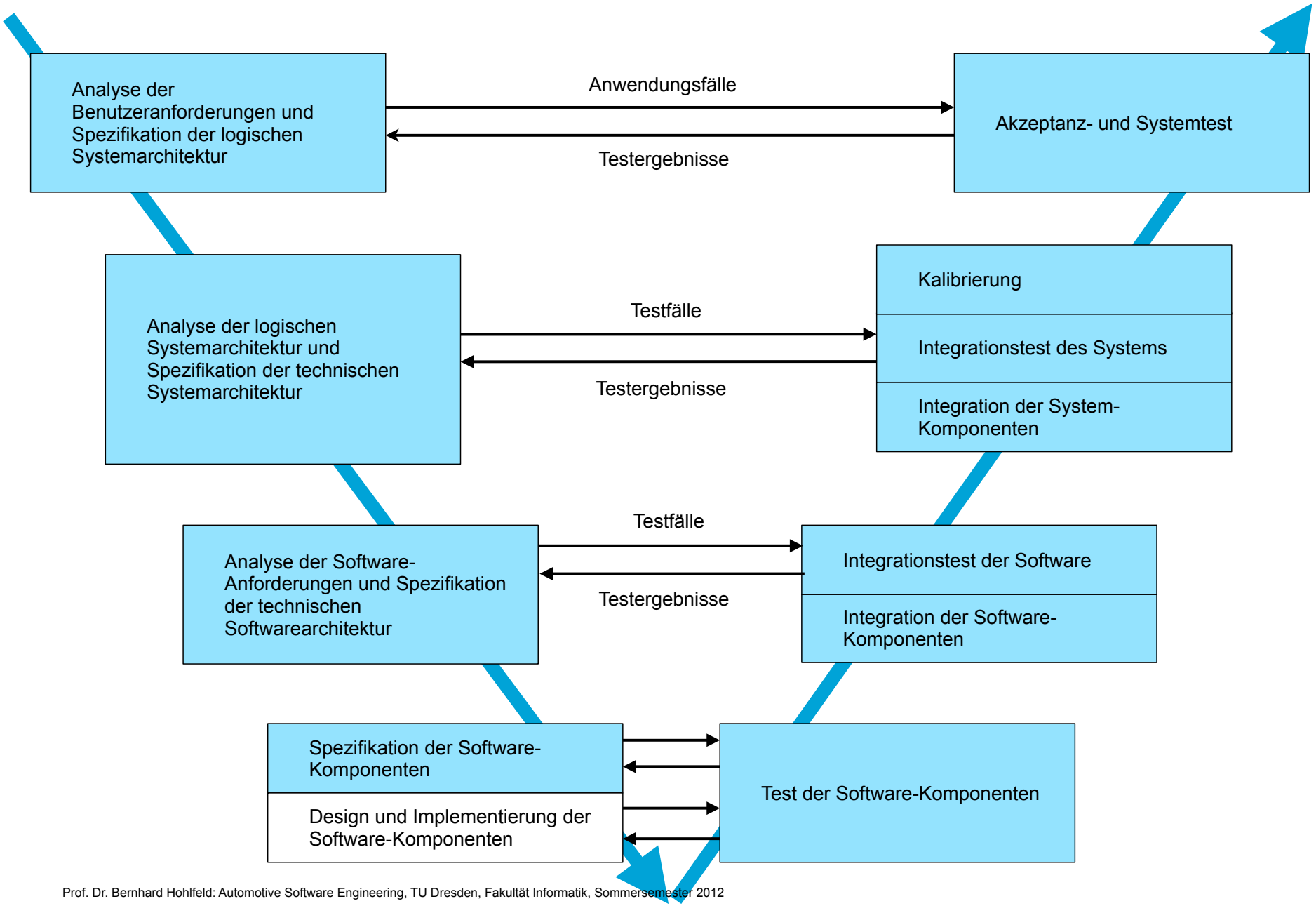
Beispiele

	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 / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

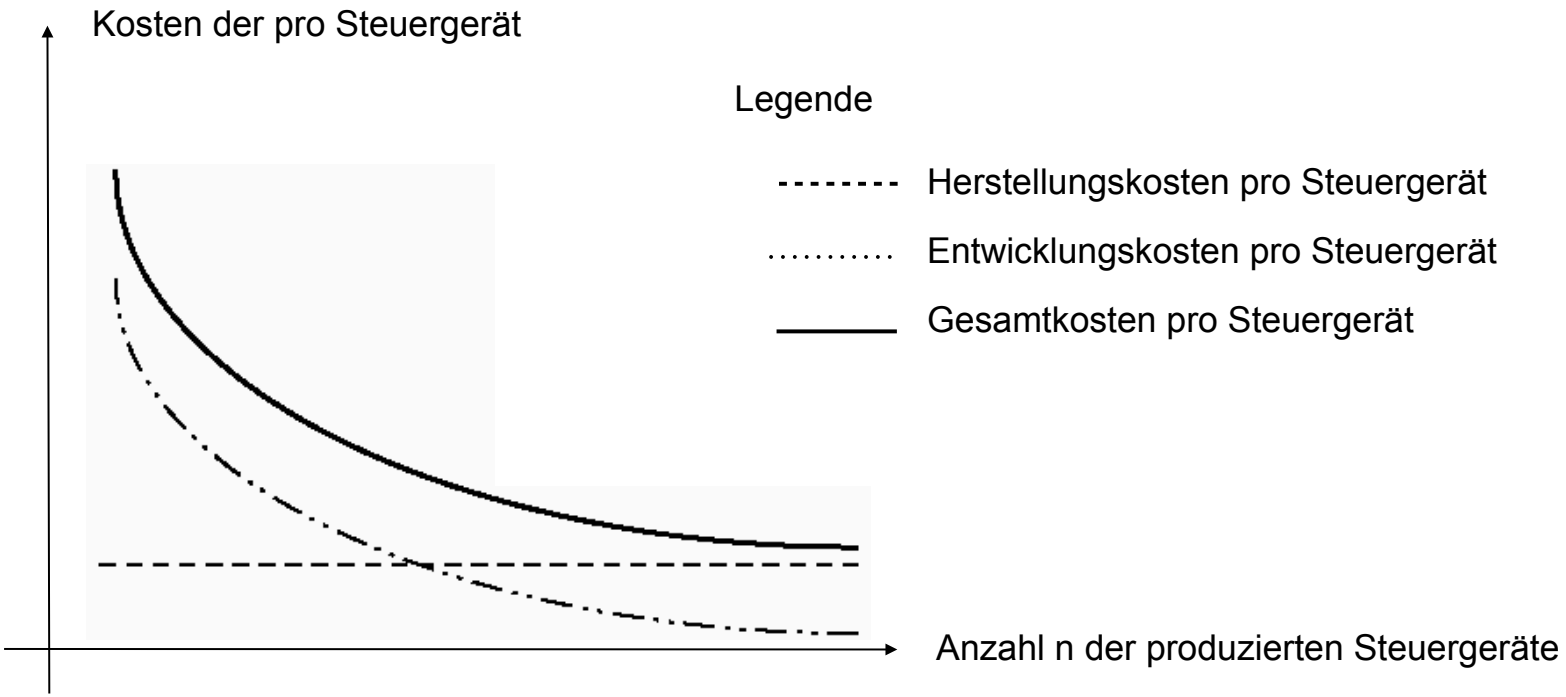
1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen 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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)

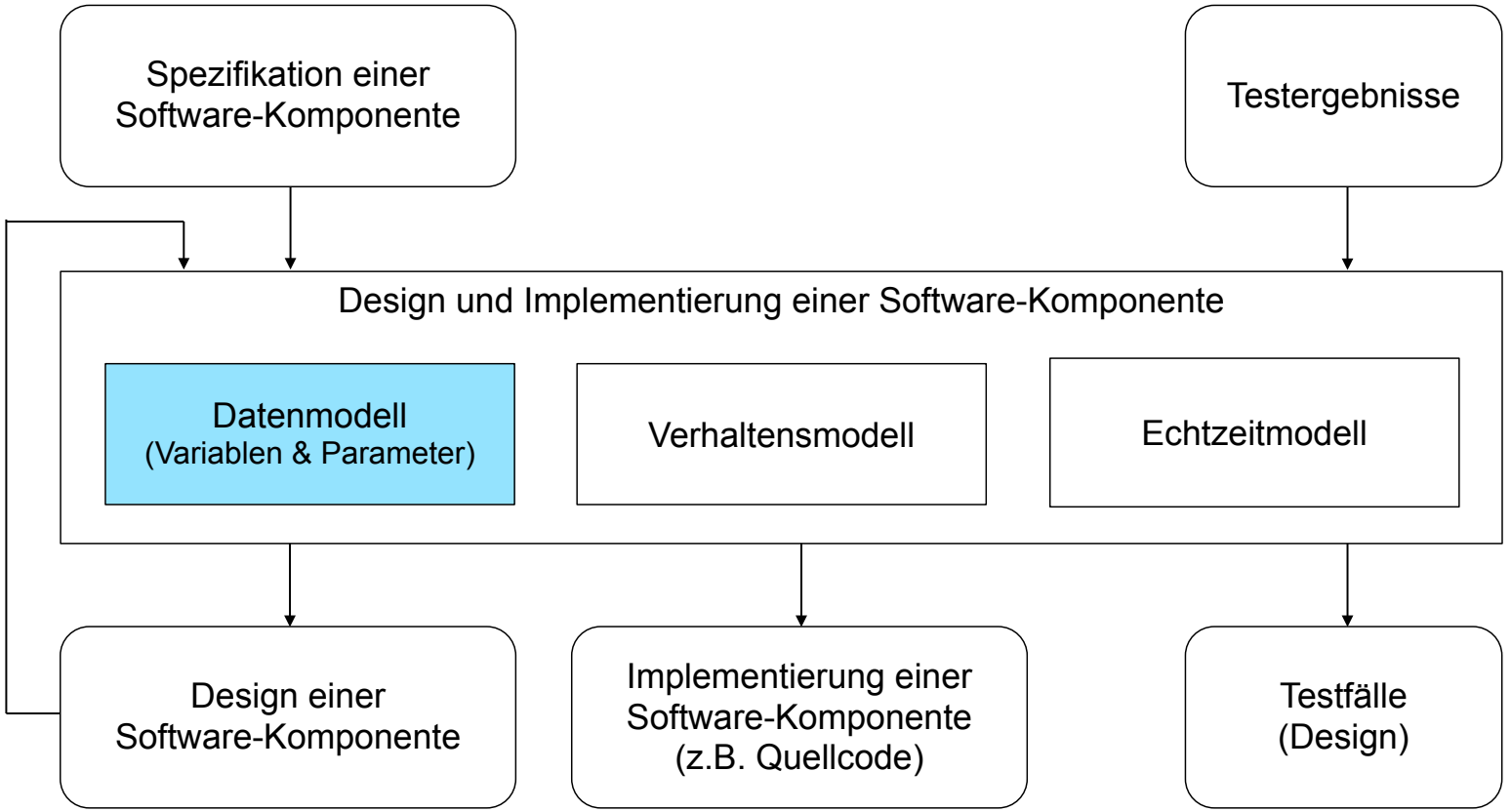


Design und Implementierung der Software-Komponenten

- 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 + Herstellungskosten) / n



Design und Implementierung der Software-Komponenten(Nach Schäuffele, Zurawka)

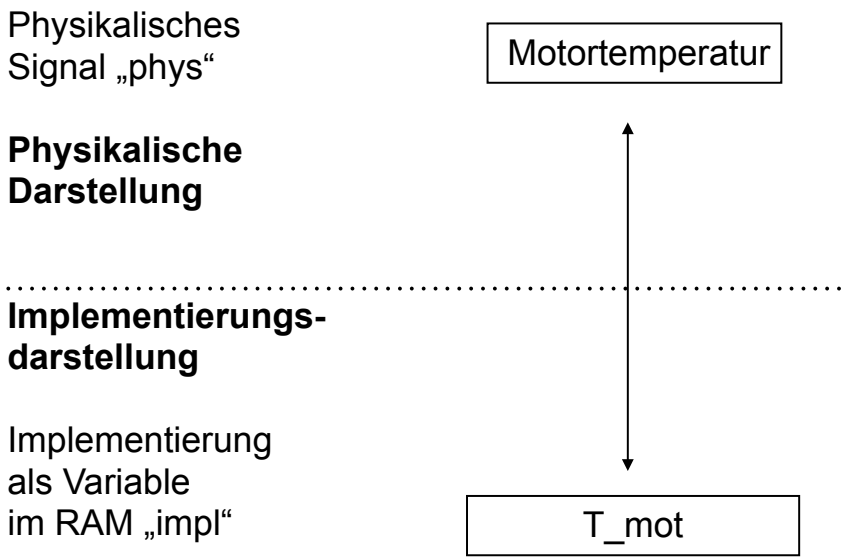


Design und Implementierung der Software-Komponenten

- 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 der Software-Komponenten

- Beispiel Motortemperatur:
  Abbildung der physikalischen Spezifikation auf die Implementierung

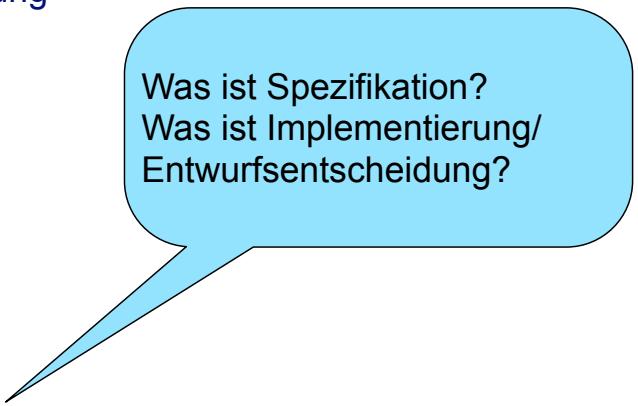


Design und Implementierung der Software-Komponenten

- Beispiel Motortemperatur:
  Abbildung der physikalischen Spezifikation auf die Implementierung
- Physik
  - Bezeichnung im Klartext: Motortemperatur
  - 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

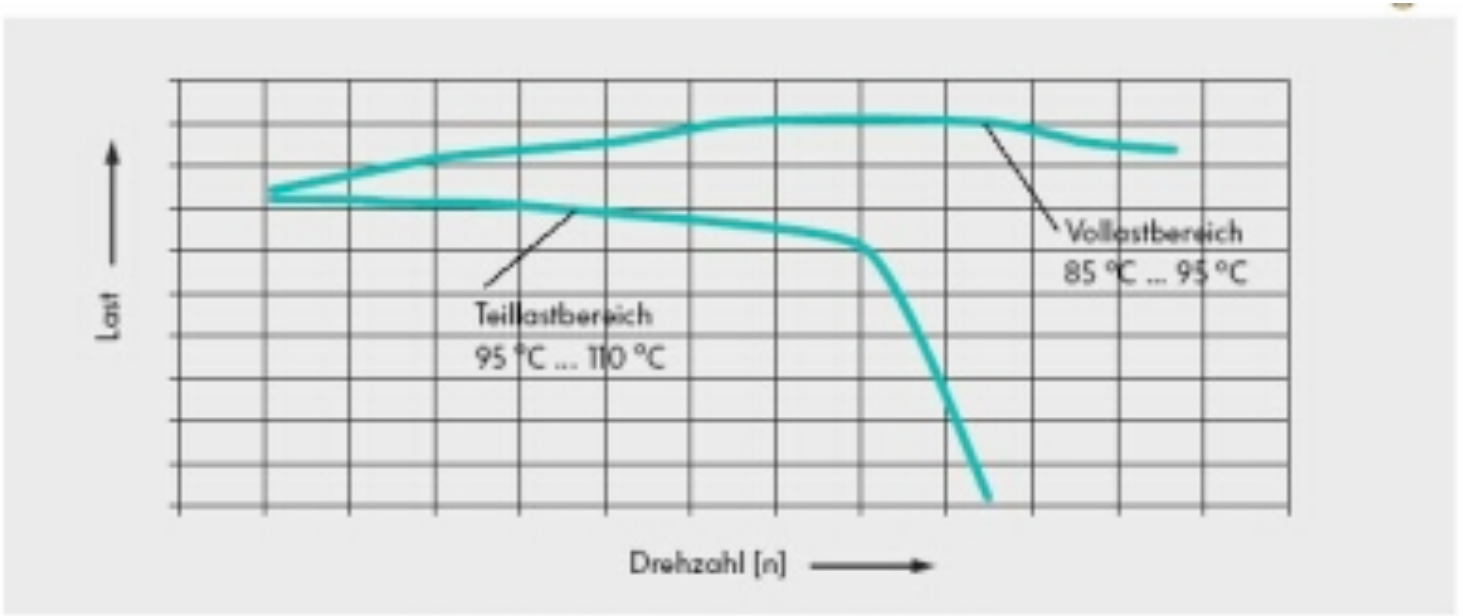
Design und Implementierung der Software-Komponenten

- Beispiel Motortemperatur:
  Abbildung der physikalischen Spezifikation auf die Implementierung
- Physik
  - Bezeichnung im Klartext: Motortemperatur
  - 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

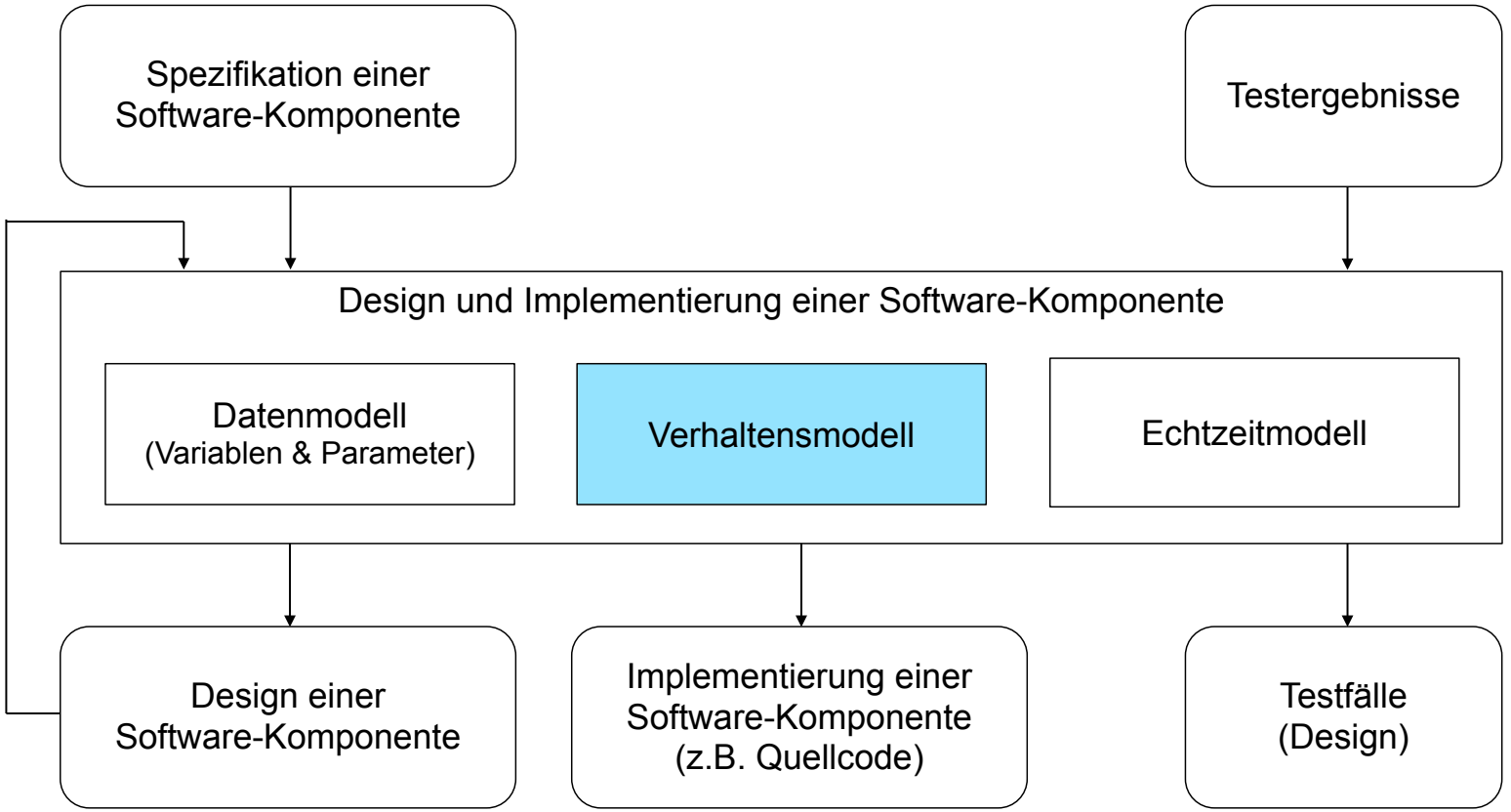
- Quelle: http://www.kfztech.de/kfztechnik/motor/kuehlung/wasserkuehlung.htm
- Die im Verbrennungsmotor erzeugten Temperaturen von über 2000°C bedrohen die nur begrenzt hitzebeständigen Motorbauteile. Die überschüssige Wärme muss deshalb schnell und zuverlässig abgeleitet werden. Bei Volllastbetrieb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

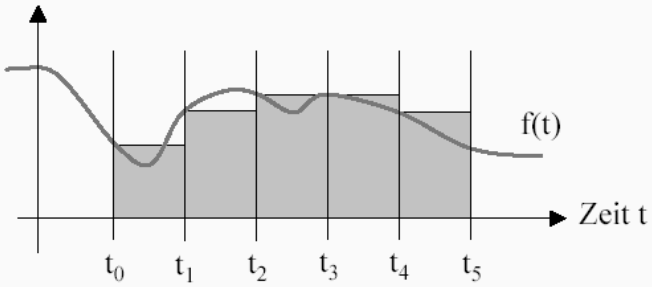
Design und Implementierung der Software-Komponenten (Nach Schäuffele, Zurawka)



Genauigkeit

- Fehler in den Eingabedaten
- Rundungsfehler
- Approximationsfehler

Integrationsverfahren nach Euler



$F(t_n) = \int_{t_0}^{t_n} f(t)dt$ durch

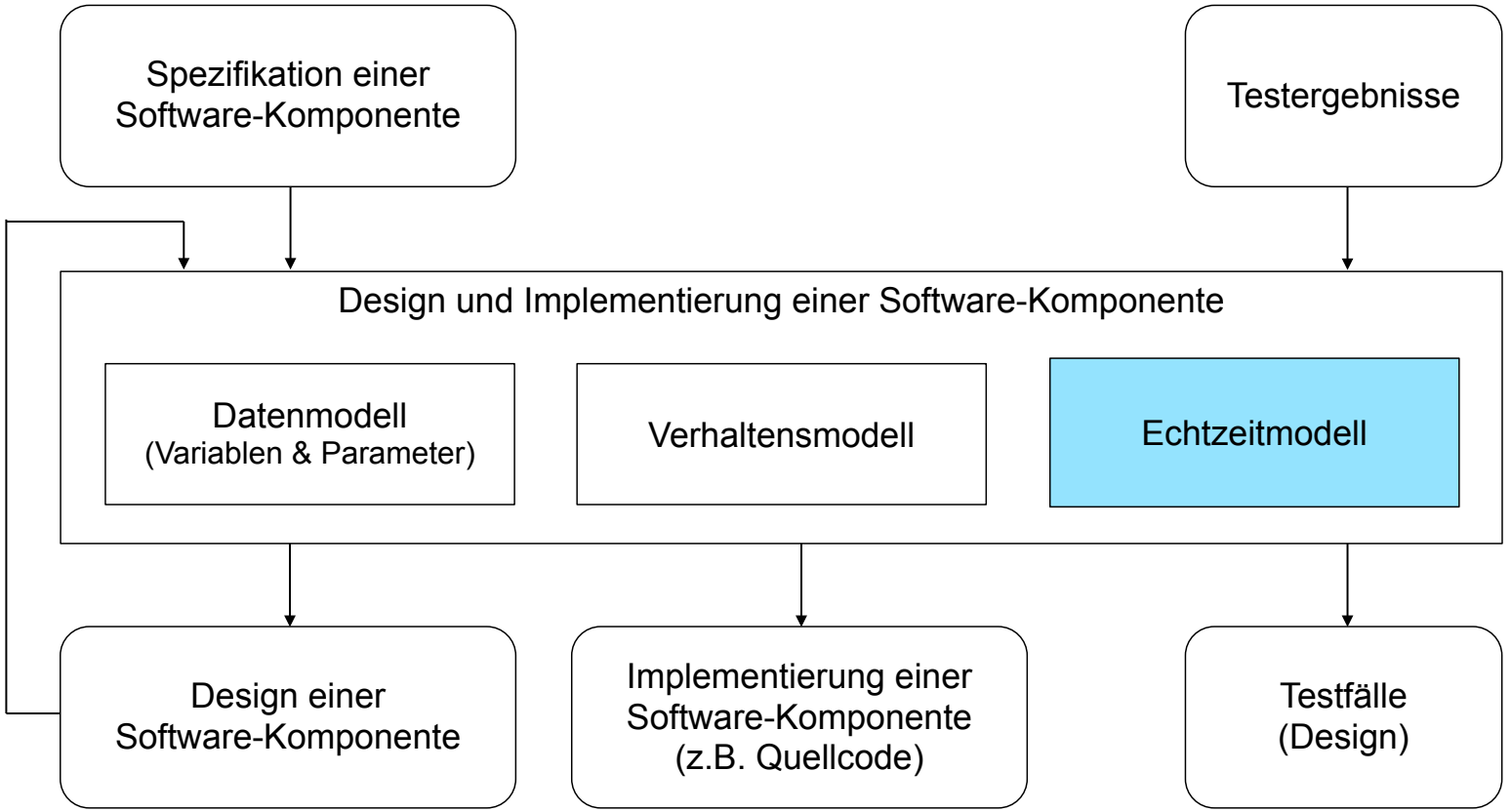
$F^*(t_n) = \sum_{i=0}^{n-1} (t_{i+1} - t_i) * f(t_i)$ approximiert

$dT_i = (t_{i+1} - t_i)$ (*Schrittweite*)

Inkrementelle Berechnung:

$F^*(t_{i+1}) = F^*(t_i) + dT_i * f(t_i)$

Design und Implementierung der Software-Komponenten (Nach Schäuffele, Zurawka)



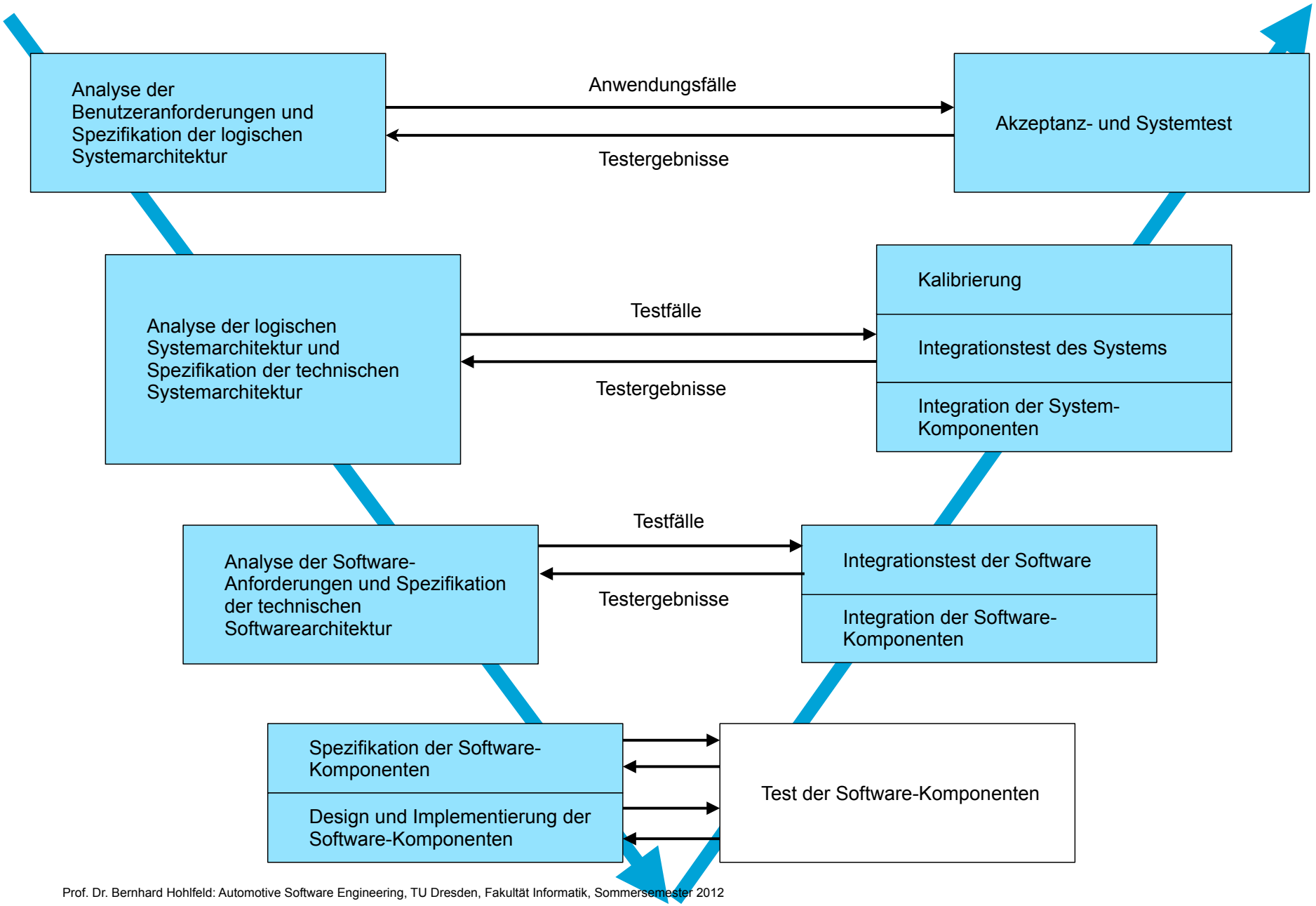
Design und Implementierung des Echtzeitmodells

- Hardware- und Software-Interrupt-System des Mikrocontrollers berücksichtigen
- Festlegung der Konfiguration des Echtzeitbetriebssystems

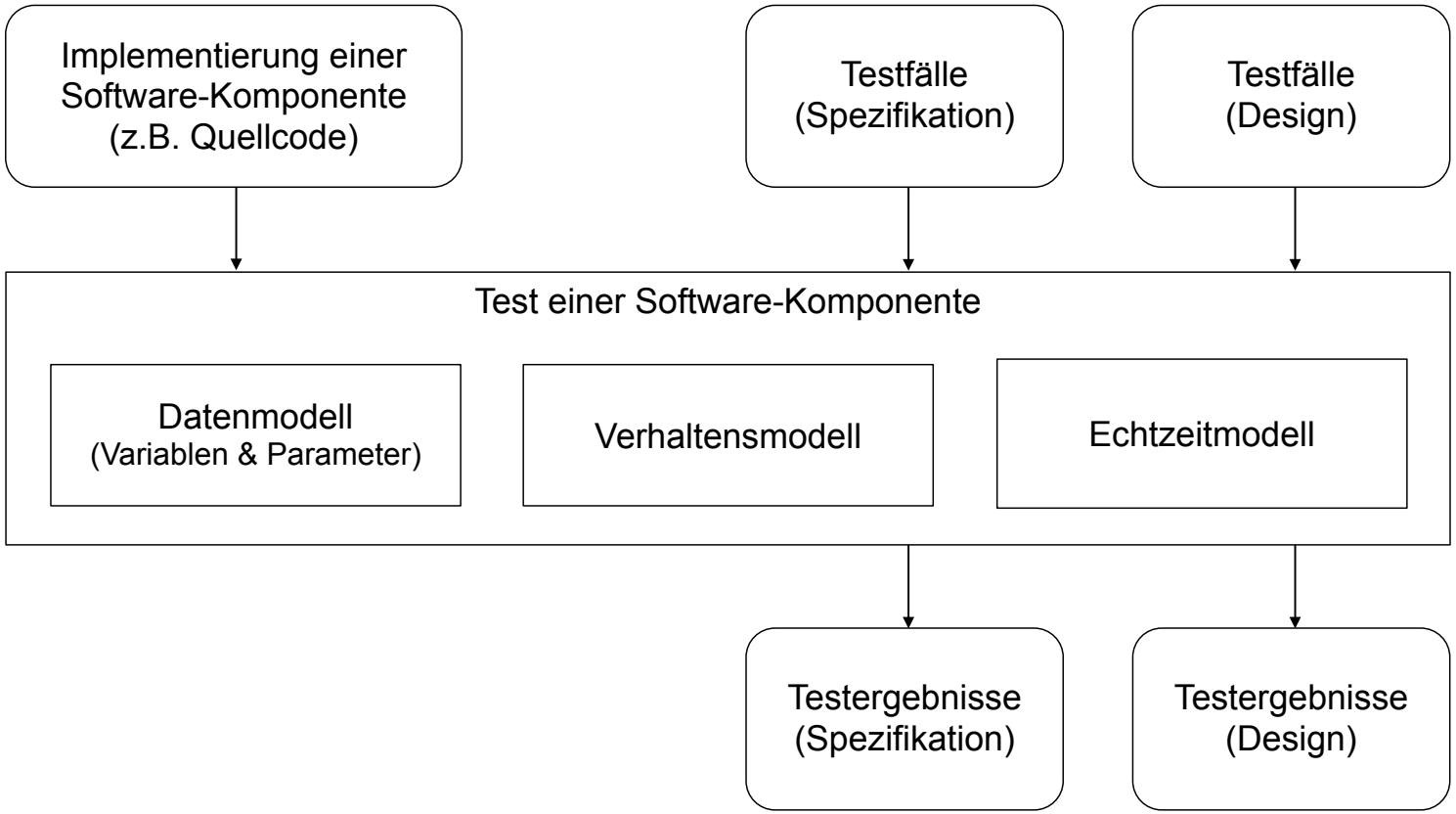
6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

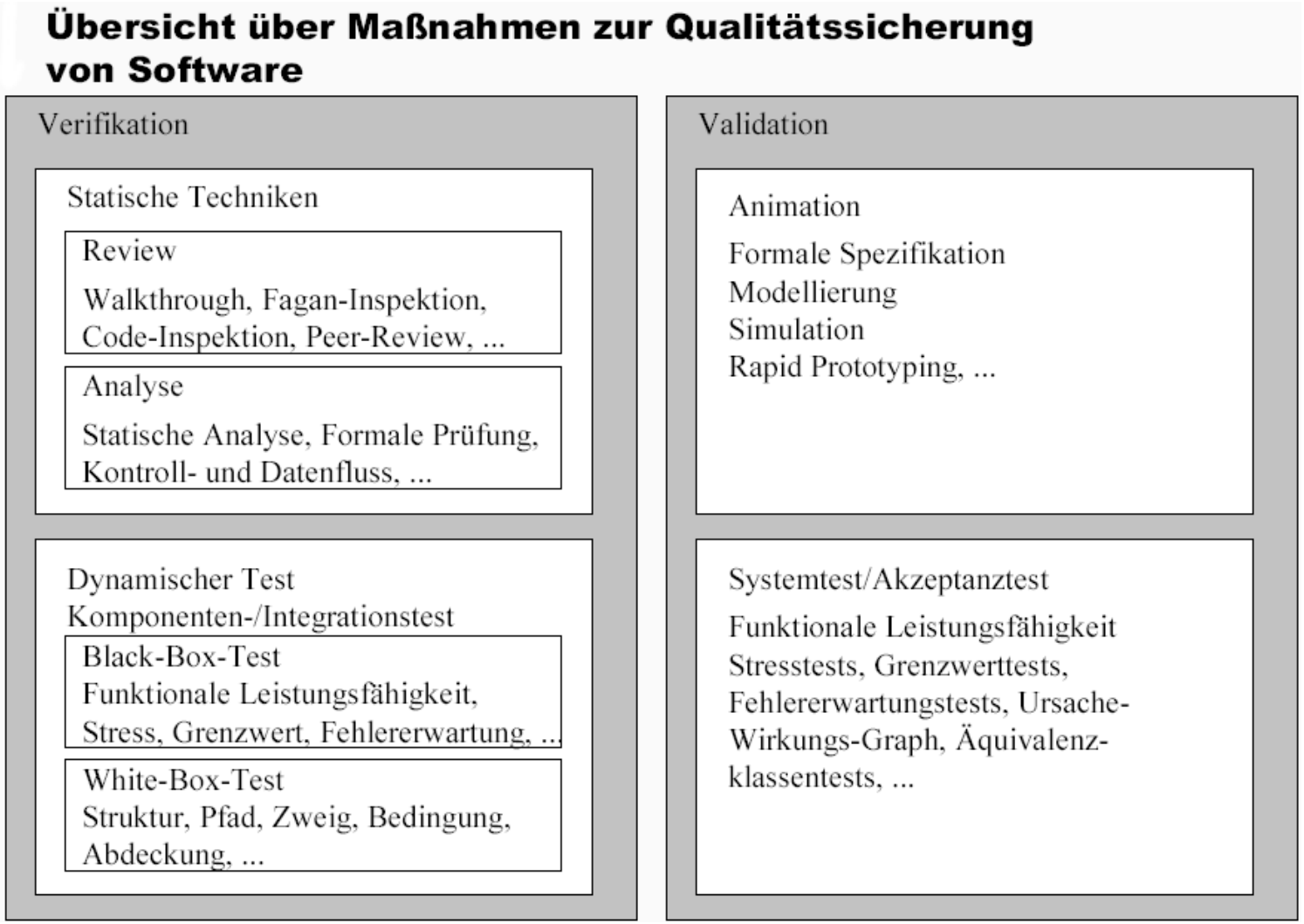
1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen 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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)

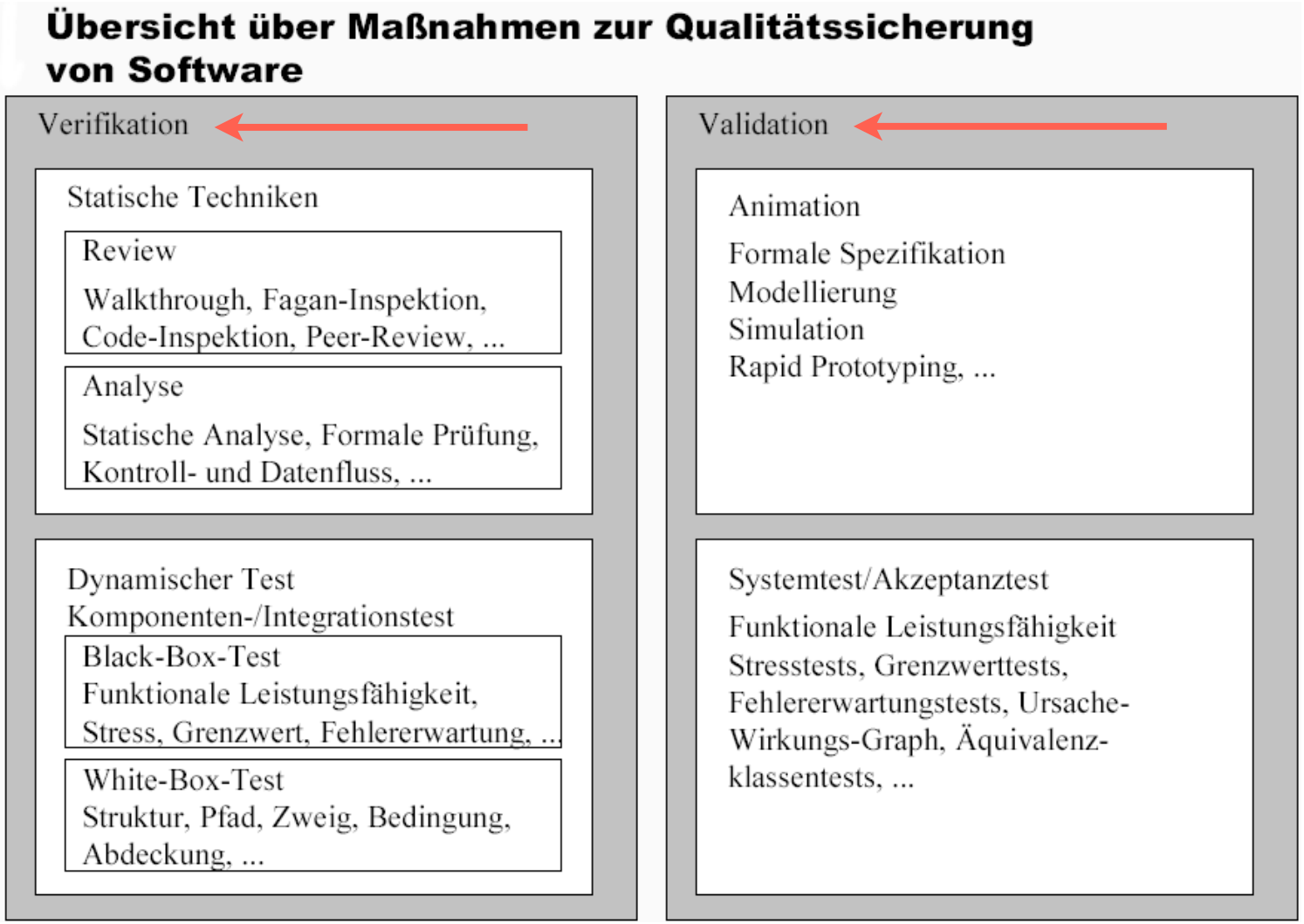


Test der Software-Komponenten (Nach Schäuffele, Zurawka)




Übersicht über 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, ...

Test der Software-Komponenten



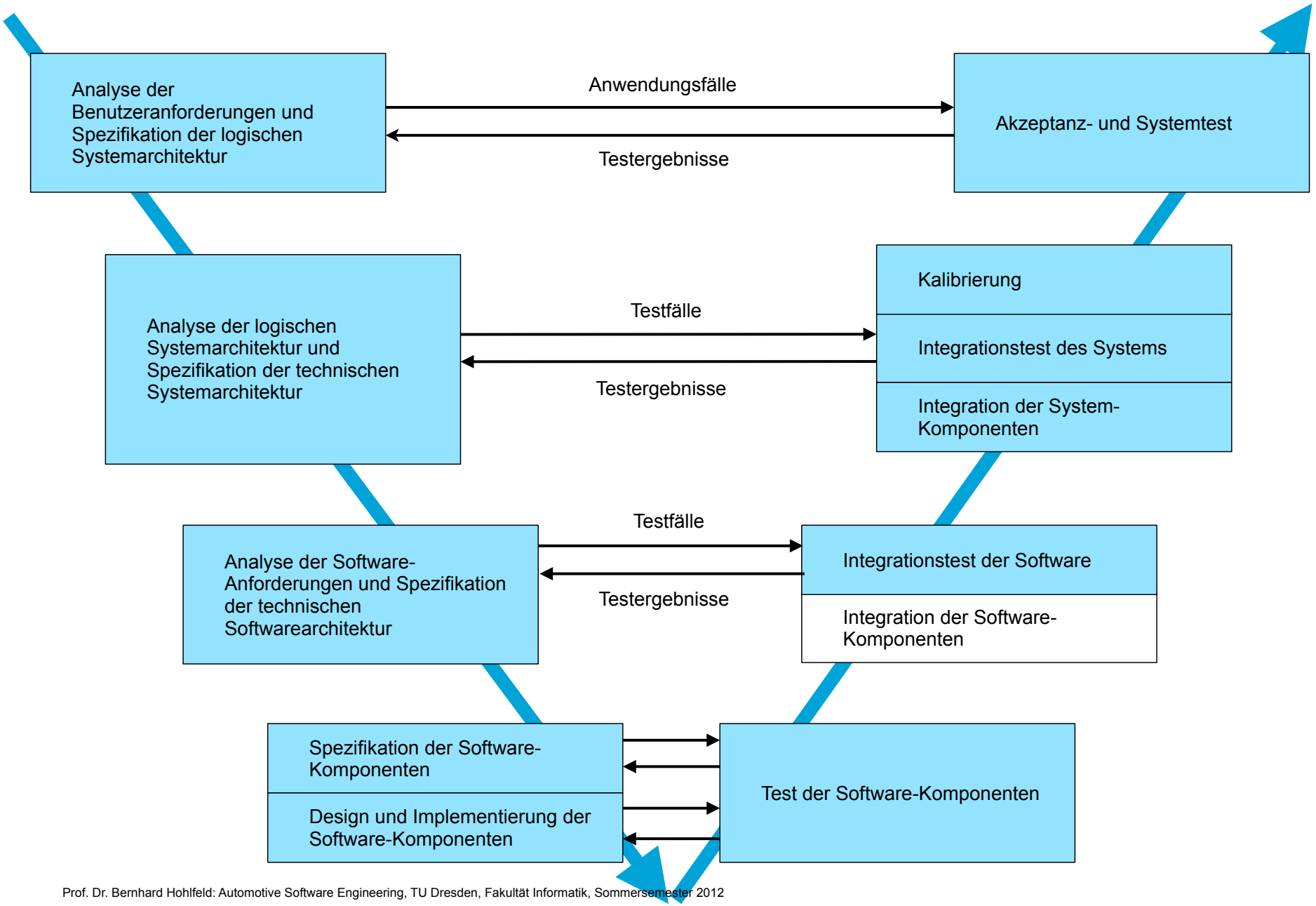
Verifikation und Validation/Validierung

- 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 / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)



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

Software-Stand für ein Seriensteuergerät

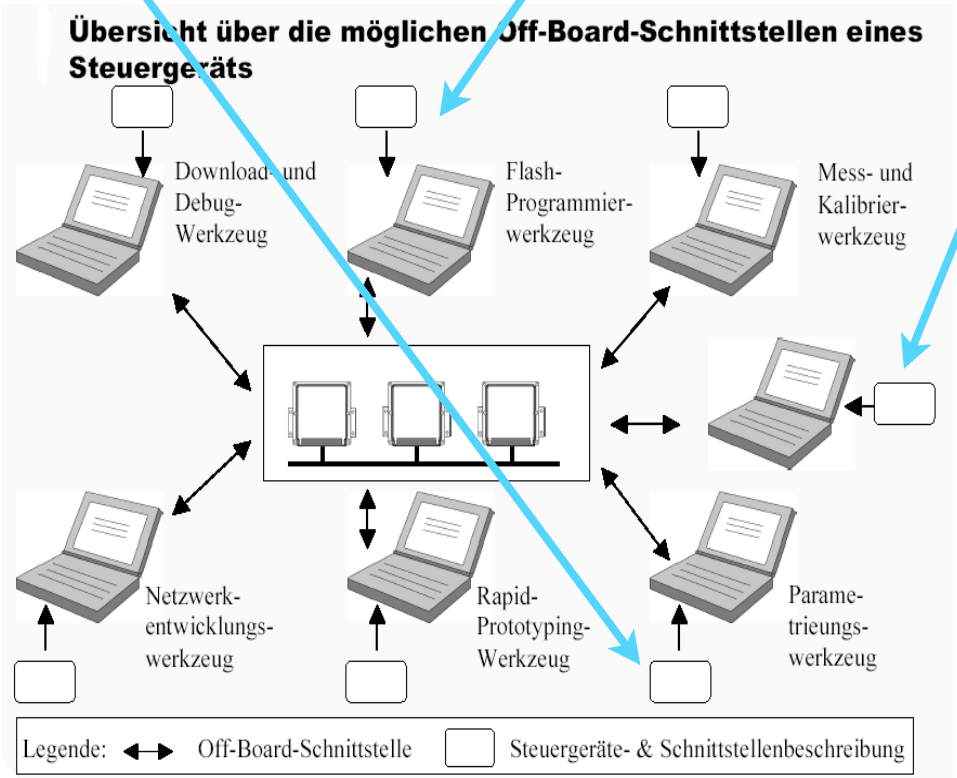
- *Programm- und Datenstände* für alle Mikrokontroller des Steuergeräts
- *Dokumentation*
- *Beschreibungsdateien* für Produktions- und Sevicewerkzeuge, wie z.B. Diagnose-, Software-Parametrisierungs- oder Flash-Programmierwerkzeuge.

Beschreibungsdateien für Entwicklungssteuergeräte

- Beschreibungsdateien für die Mess- und Kalibrierungswerkzeuge
- Beschreibungsdateien der On-Board-Kommunikation für Werkzeuge zur Netzwerkentwicklung
- Beschreibungsdateien der so genannten Bypass-Schnittstelle, falls Rapid-Prototyping-Werkzeuge eingesetzt werden.

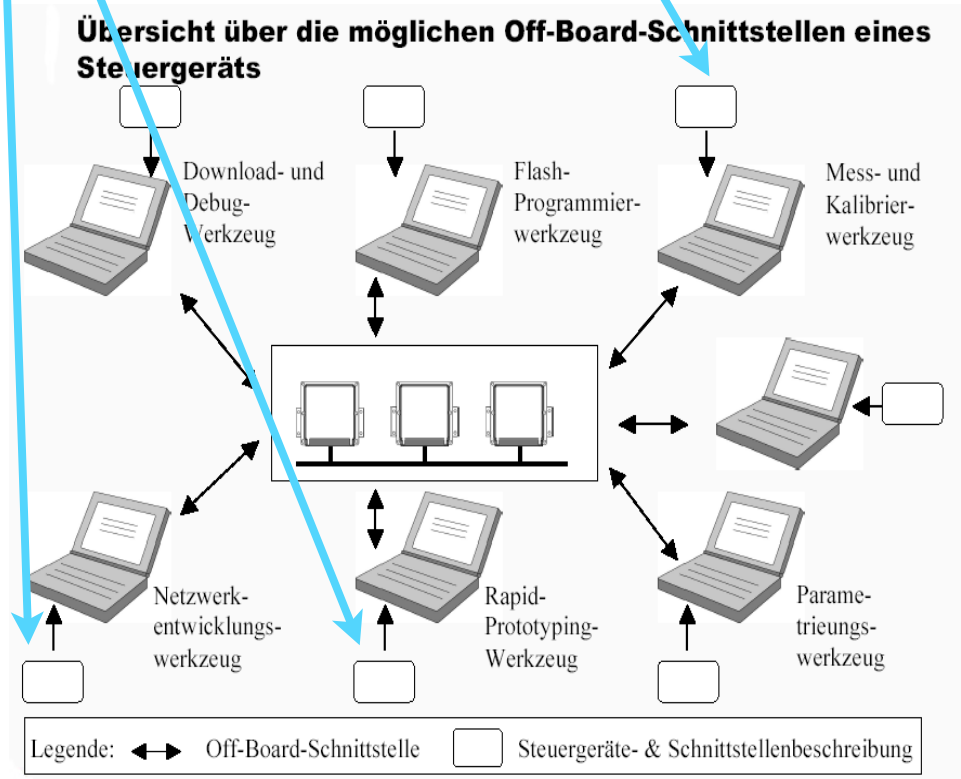
Software-Stand für ein Seriensteuergerät

- *Programm- und Datenstände* für alle Mikrokontroller des Steuergeräts
- *Dokumentation*
- *Beschreibungsdateien* für Produktions- und Sevicewerkzeuge, wie z.B. Diagnose-, Software-Parametrisierungs- oder Flash-Programmierwerkzeuge.

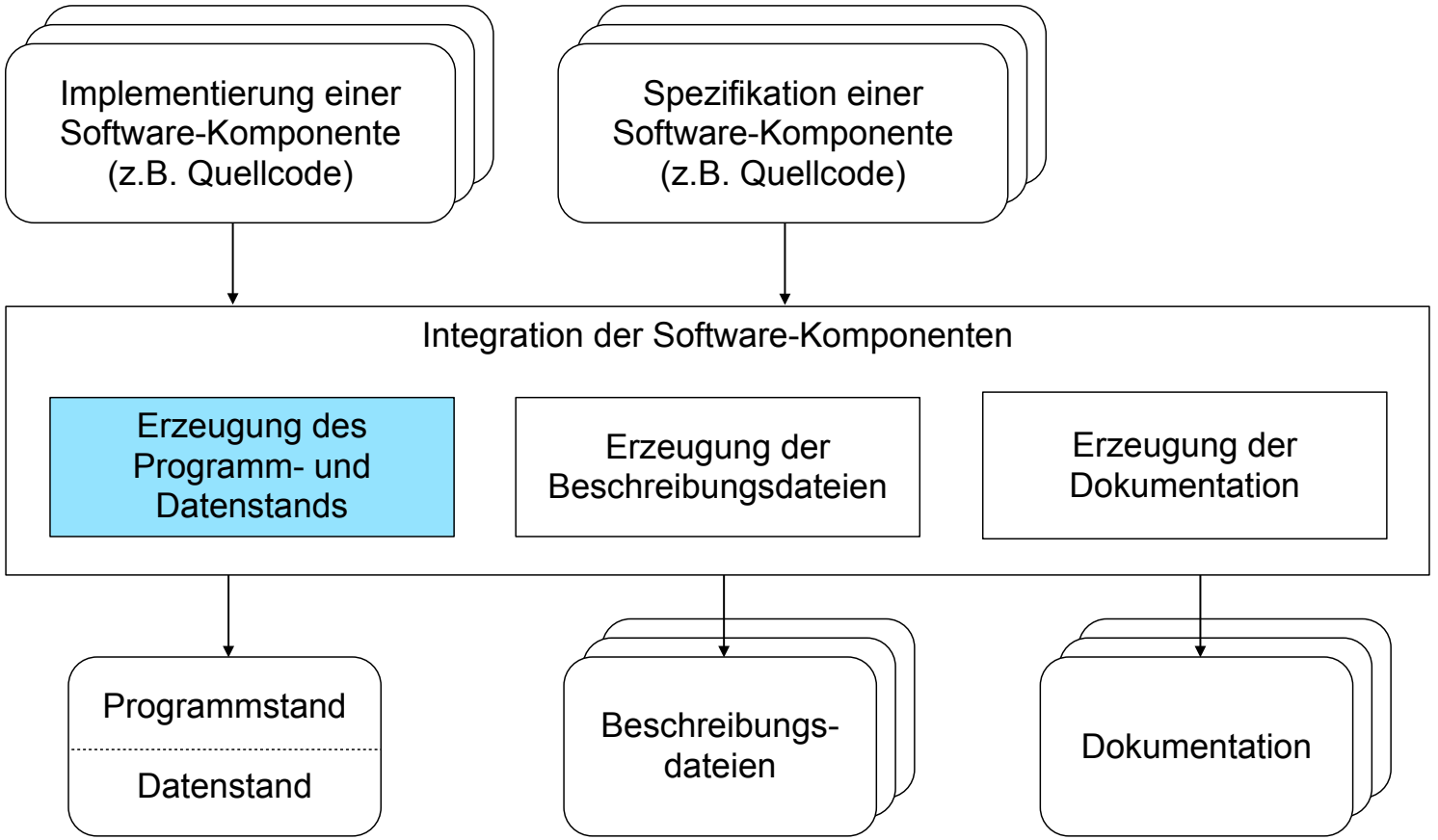


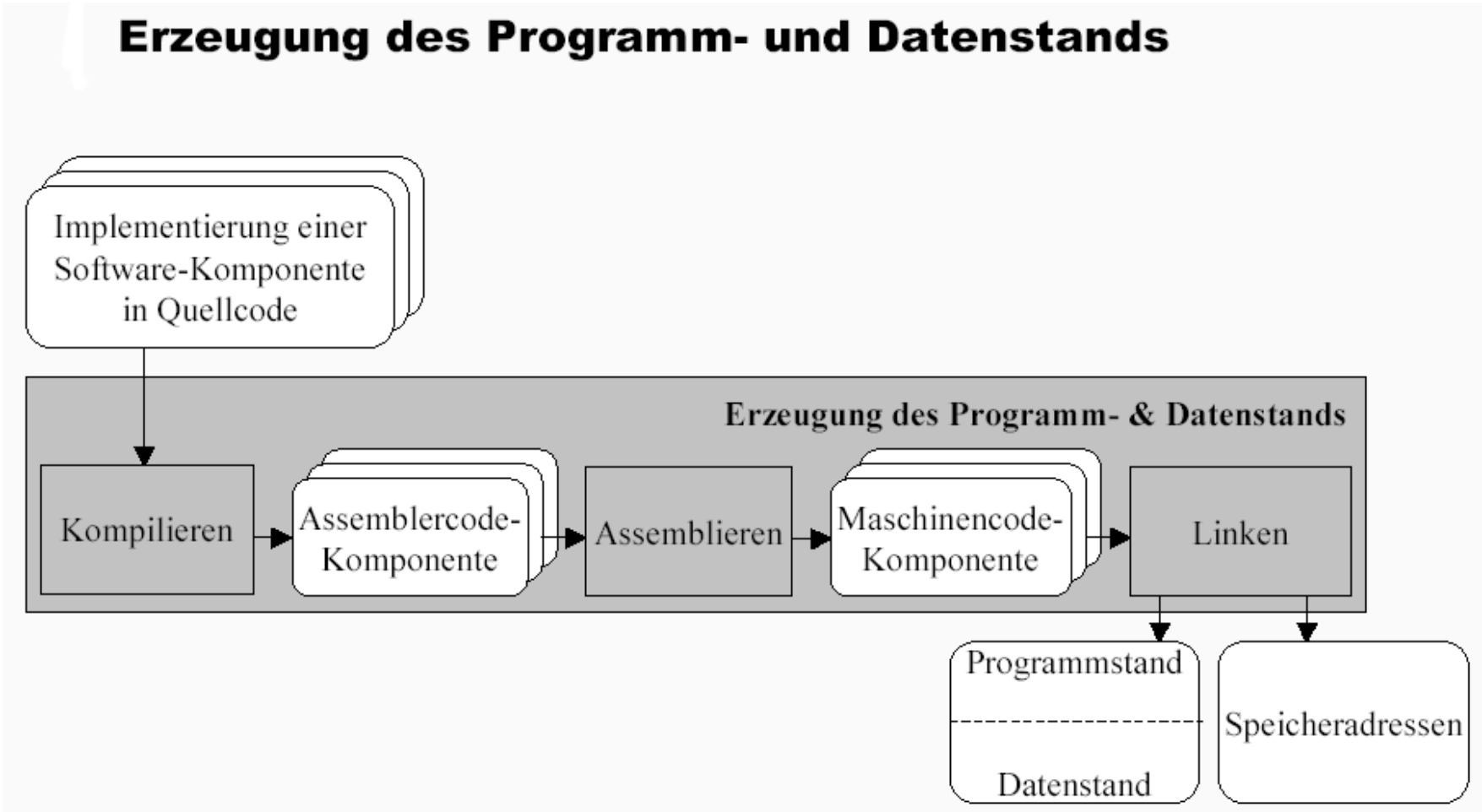
Beschreibungsdateien für Entwicklungssteuergeräte

- Beschreibungsdateien für die Mess- und Kalibrierungswerkzeuge
- Beschreibungsdateien der On-Board-Kommunikation für Werkzeuge zur Netzwerkentwicklung
- Beschreibungsdateien der so genannten Bypass-Schnittstelle, falls Rapid-Prototyping-Werkzeuge eingesetzt werden.

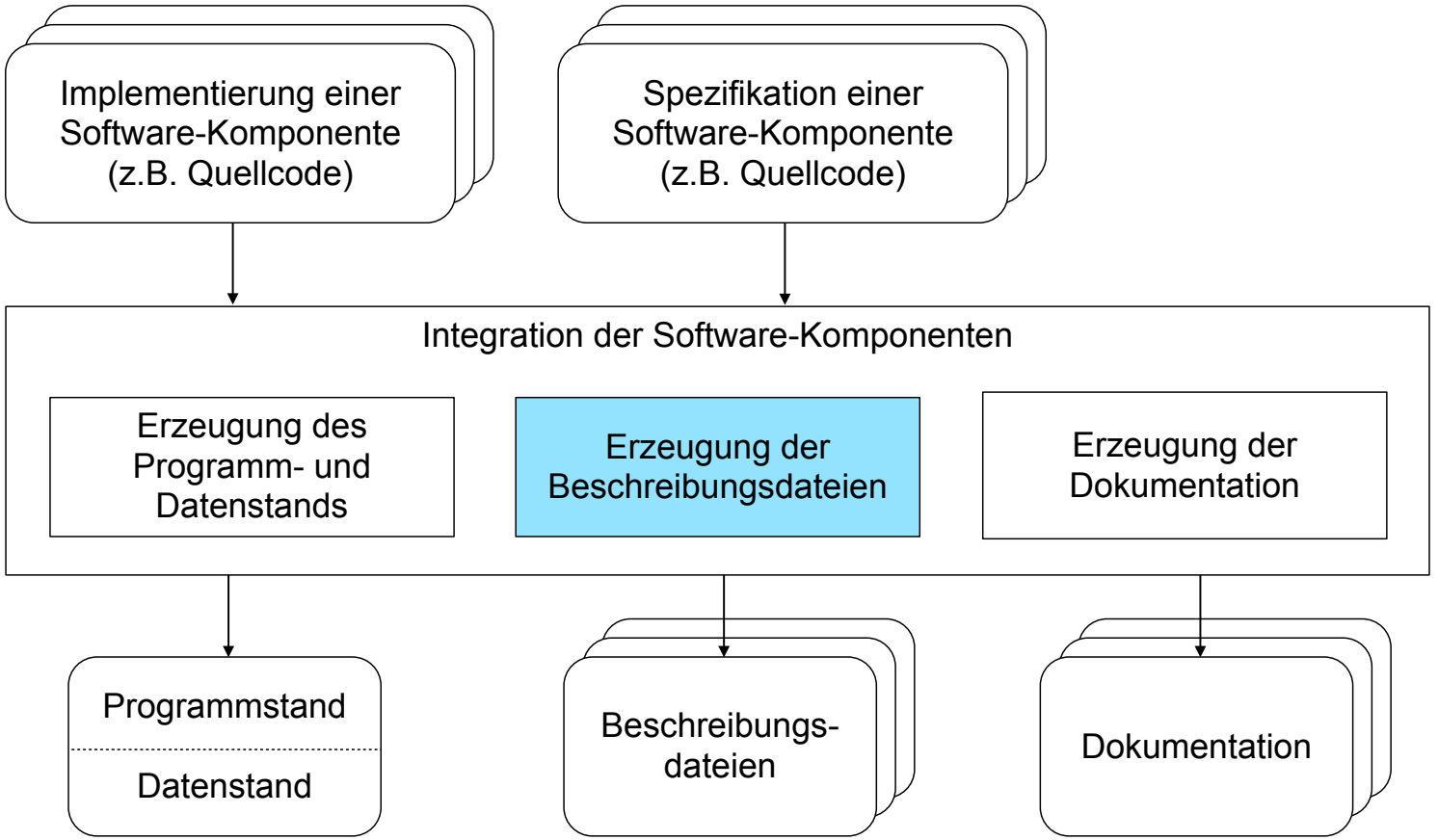


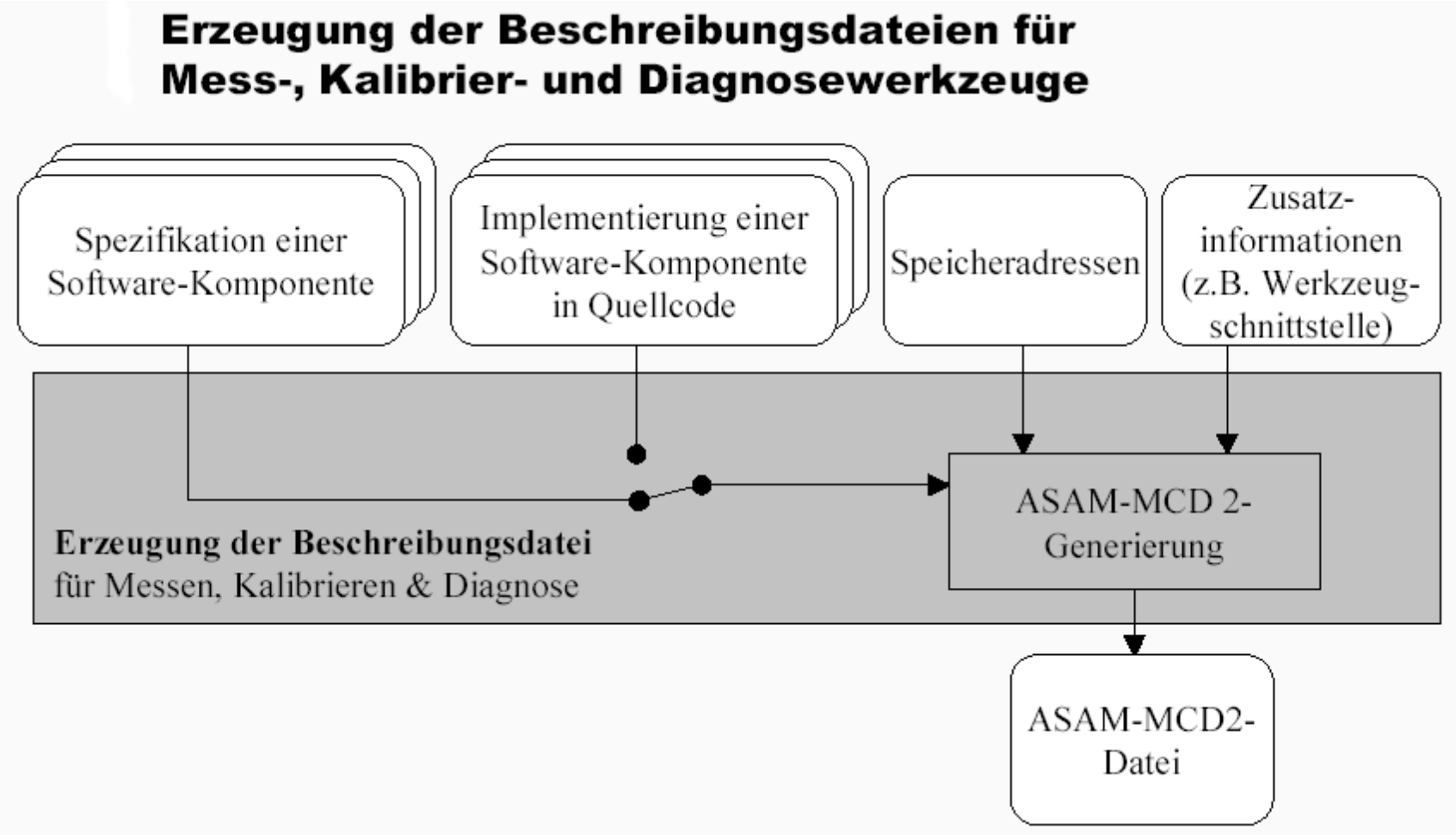
Integration der Software-Komponenten (Nach Schäuffele, Zurawka)




Erzeugung des Programm- und Datenstands
Implementierung einer Software-Komponente in Quellcode
Erzeugung des Programm- & Datenstands
Kompilieren
Assemblercode-Komponente
Assemblieren
Maschinencode-Komponente
Linken
Programmstand
Datenstand
Speicheradressen

Integration der Software-Komponenten (Nach Schäuffele, Zurawka)



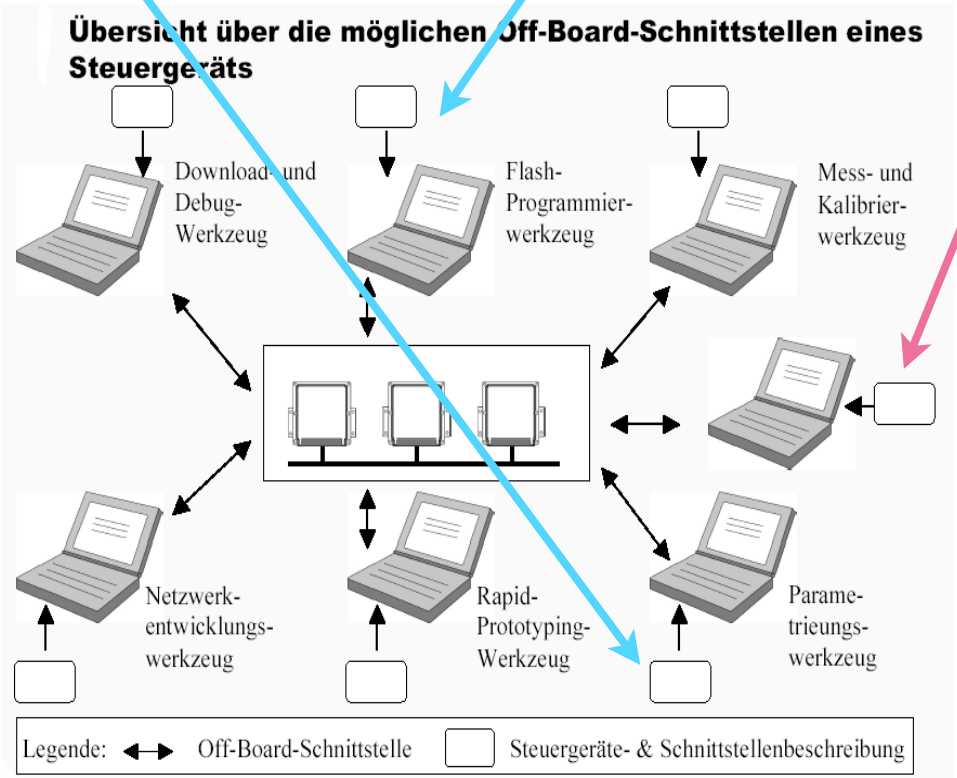

Erzeugung der Beschreibungsdateien für Mess-, Kalibrier- und Diagnosewerkzeuge
Spezifikation einer Software-Komponente
Implementierung einer Software-Komponente in Quellcode
Speicheradressen
Zusatz-informationen (z.B. Werkzeug-schnittstelle)
Erzeugung der Beschreibungsdatei für Messen, Kalibrieren & Diagnose
ASAM-MCD 2-Generierung
ASAM-MCD2-Datei

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

- 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. Therefore it might be necessary to update an ASAP2 file with the compile and link run during ECU software development process.

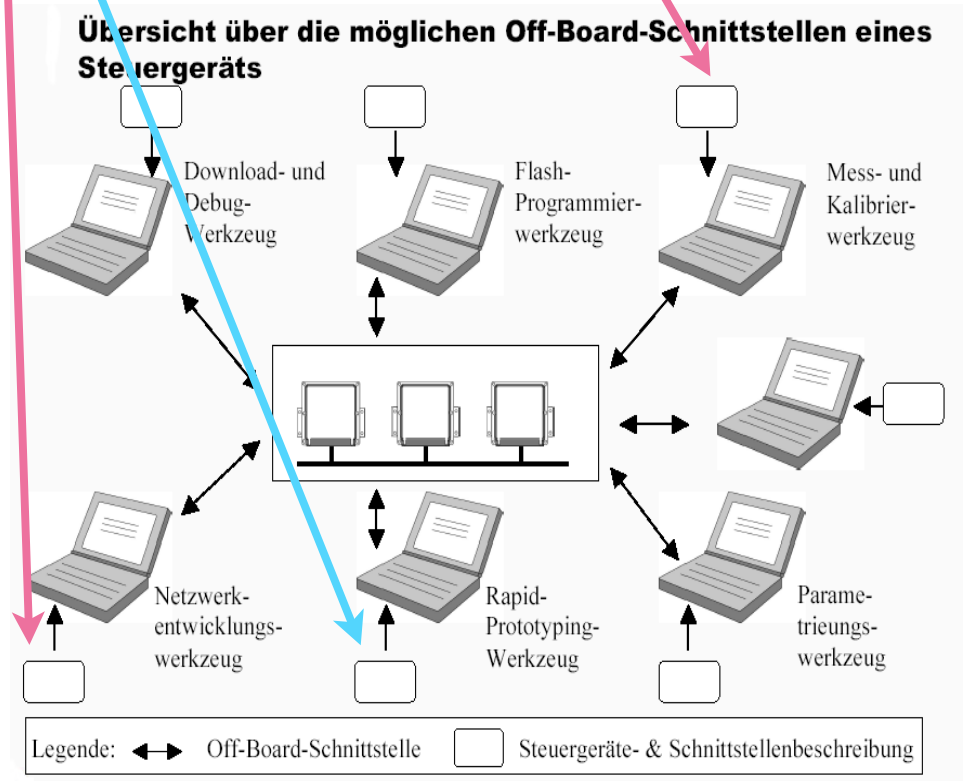
Software-Stand für ein Seriensteuergerät

- *Programm- und Datenstände* für alle Mikrokontroller des Steuergeräts
- *Dokumentation*
- *Beschreibungsdateien* für Produktions- und Sevicewerkzeuge, wie z.B. Diagnose-, Software-Parametrisierungs- oder Flash-Programmierwerkzeuge.

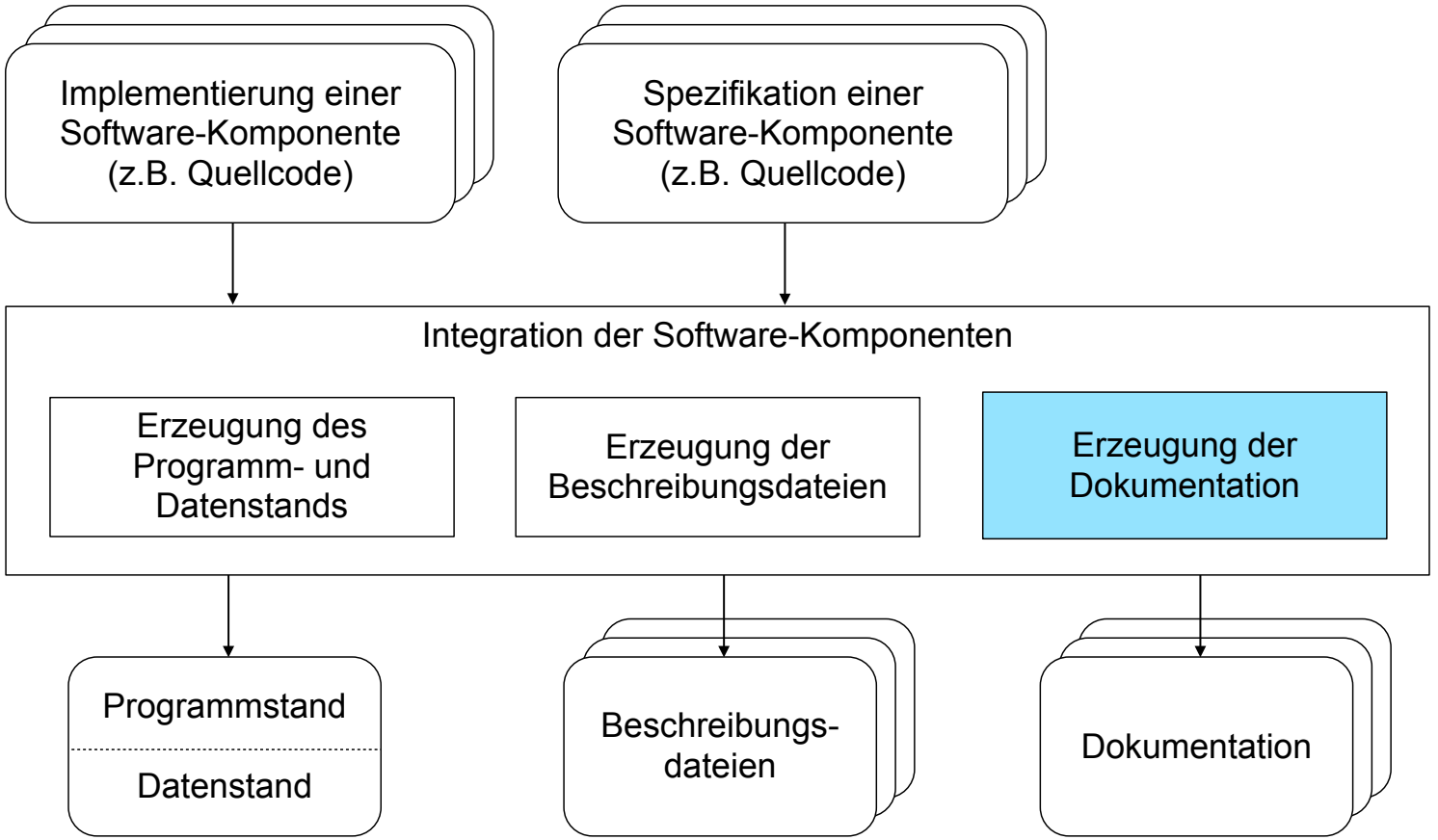


Beschreibungsdateien für Entwicklungssteuergeräte

- Beschreibungsdateien für die Mess- und Kalibrierungswerkzeuge
- Beschreibungsdateien der On-Board-Kommunikation für Werkzeuge zur Netzwerkentwicklung
- Beschreibungsdateien der so genannten Bypass-Schnittstelle, falls Rapid-Prototyping-Werkzeuge eingesetzt werden.



Integration der Software-Komponenten (Nach Schäuffele, Zurawka)



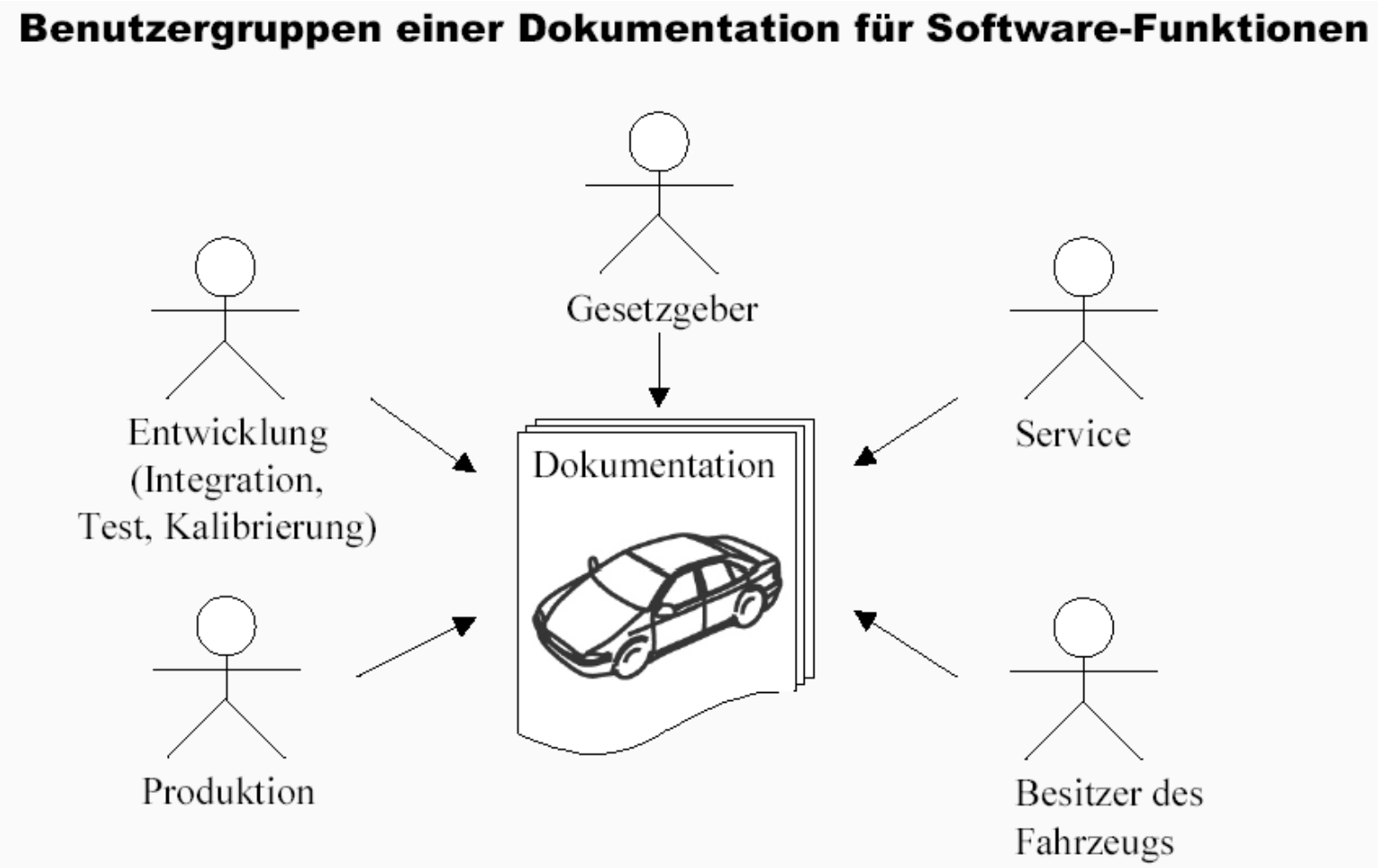
Dokumentation

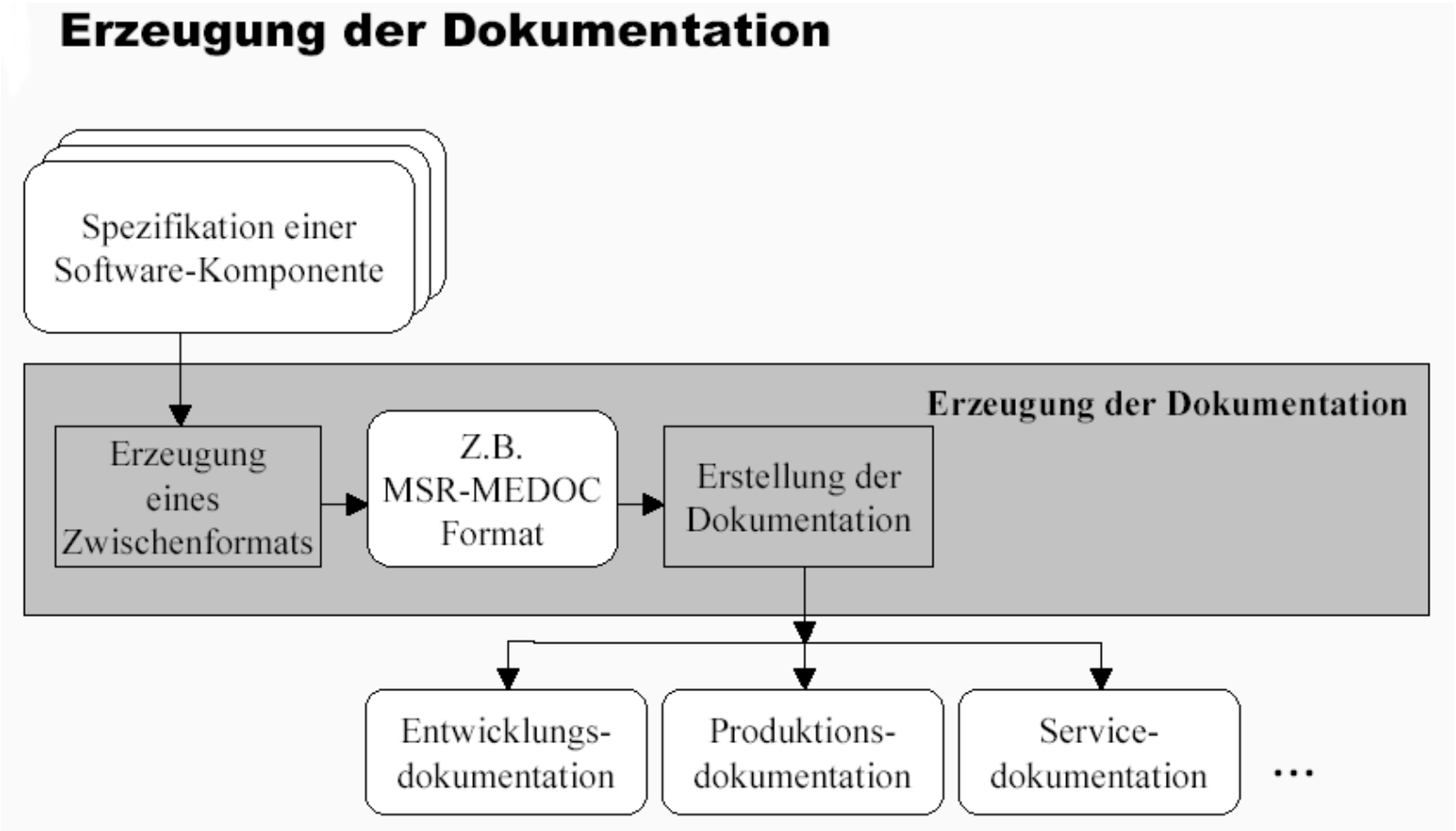
- Die Dokumentation stellt ein Artefakt dar, das für alle Unterstützungsprozesse in der Software-Entwicklung selbst benötigt wird. Auch die ausgeprägte und oft firmenübergreifende Arbeitsteilung, die langen Produktlebenszyklen, sowie die damit verbundene lange Wartungsphase für die Software erfordern eine ausführliche Dokumentation.
- Alle folgenden Entwicklungsschritte - wie Integration, Test und Kalibrierung des Systems - benötigen eine Dokumentation.
- Eine Dokumentation ist für die Fahrzeugproduktion und im weltweiten Service notwendig.
- Für den Gesetzgeber ist eine Dokumentation notwendig, etwa als Bestandteil für die Beantragung der Zulassung eines Fahrzeugs zum Straßenverkehr.

6. SW-Entwicklung
2. Unterstützungsprozesse

Dokumentation

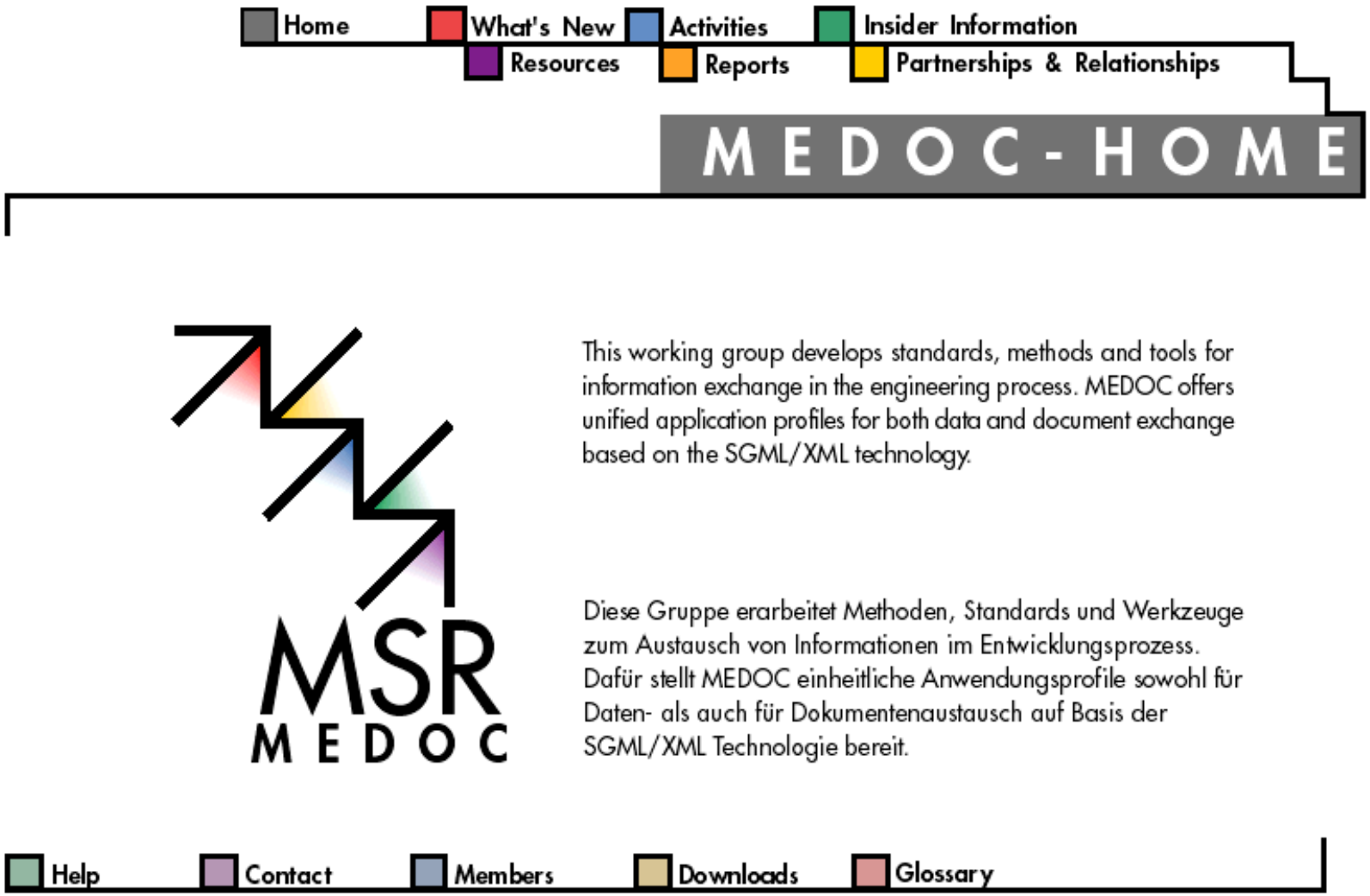
- Die Dokumentation stellt ein Artefakt dar, das für alle Unterstützungsprozesse in der Software-Entwicklung selbst benötigt wird. Auch die ausgeprägte und oft firmenübergreifende Arbeitsteilung, die langen Produktlebenszyklen, sowie die damit verbundene lange Wartungsphase für die Software erfordern eine ausführliche Dokumentation.
- Alle folgenden Entwicklungsschritte - wie Integration, Test und Kalibrierung des Systems - benötigen eine Dokumentation.
- Eine Dokumentation ist für die Fahrzeugproduktion und im weltweiten Service notwendig.
- Für den Gesetzgeber ist eine Dokumentation notwendig, etwa als Bestandteil für die Beantragung der Zulassung eines Fahrzeugs zum Straßenverkehr.


Benutzergruppen einer Dokumentation für Software-Funktionen
Gesetzgeber
Entwicklung
(Integration,
Test, Kalibrierung)
Service
Dokumentation
Produktion
Besitzer des
Fahrzeugs


Erzeugung der Dokumentation
Spezifikation einer Software-Komponente
Erzeugung der Dokumentation
Erzeugung eines Zwischenformats
Z.B. MSR-MEDOC Format
Erstellung der Dokumentation
Entwicklungs-dokumentation
Produktions-dokumentation
Service-dokumentation
...

MEDOC
MSR Engineering Data Objects and Contents

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



MSR/MEDOC
Quelle: http://www.msr-wg.de/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.

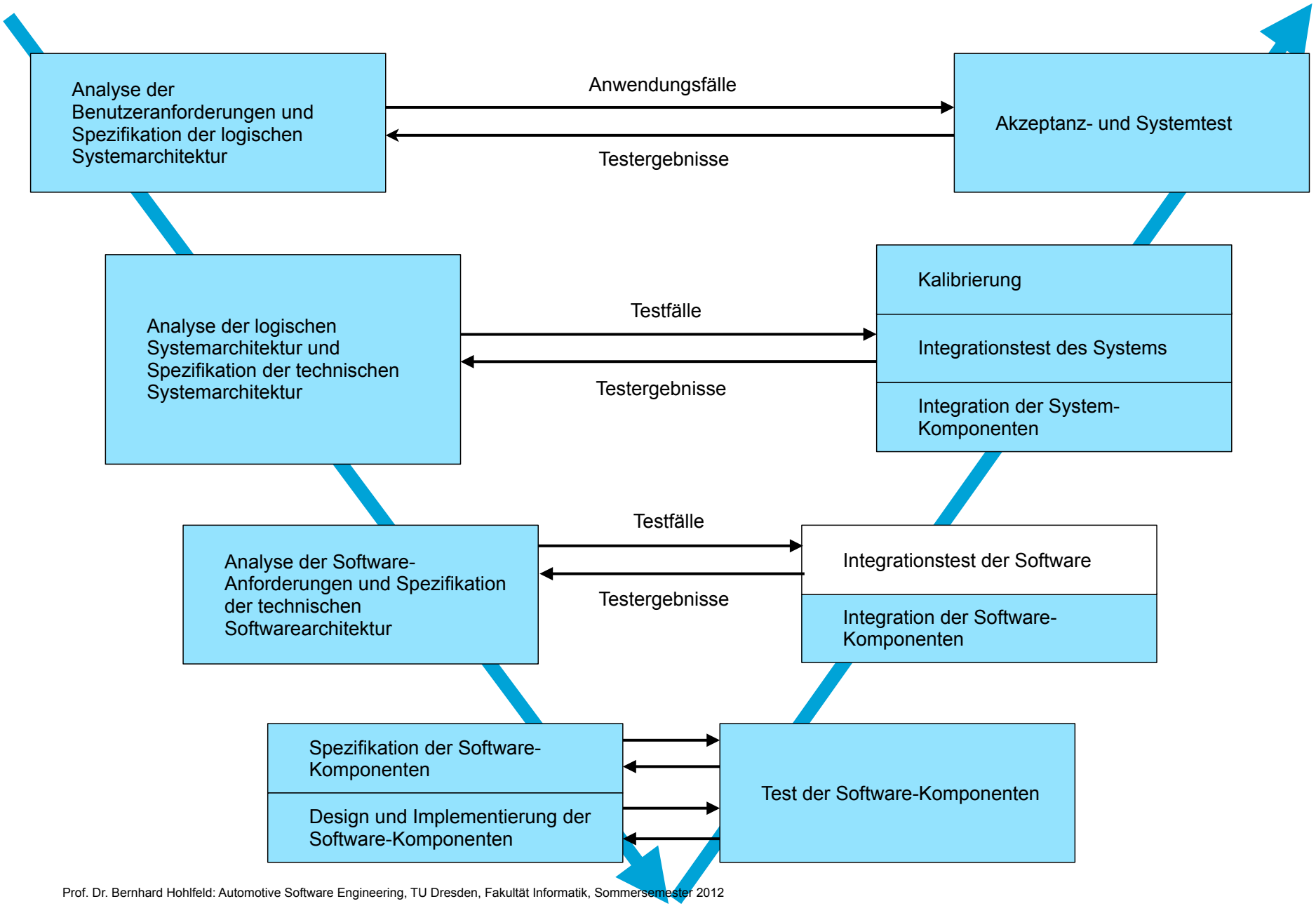
- What is MSR/MEDOC?

  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.

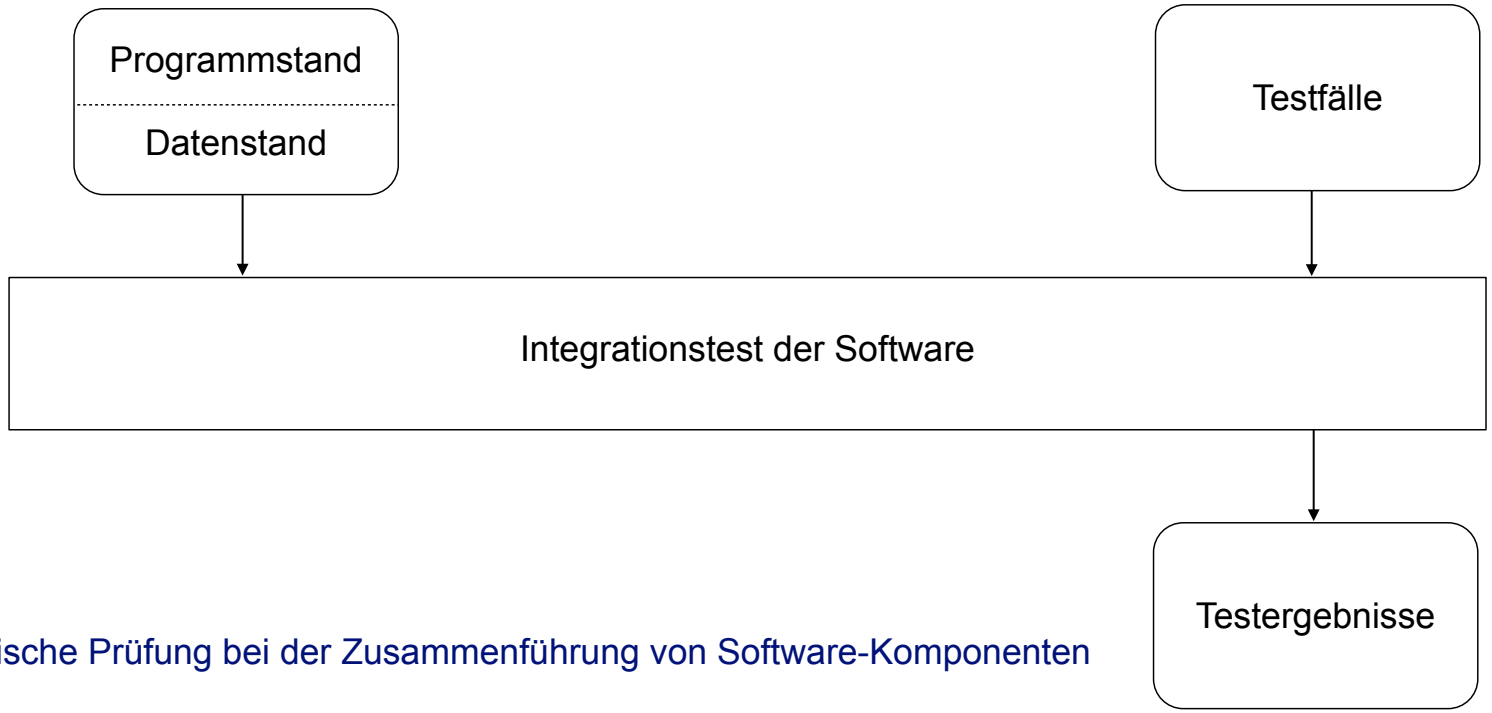
6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)



Integrationstest der Software (Nach Schäuffele, Zurawka)



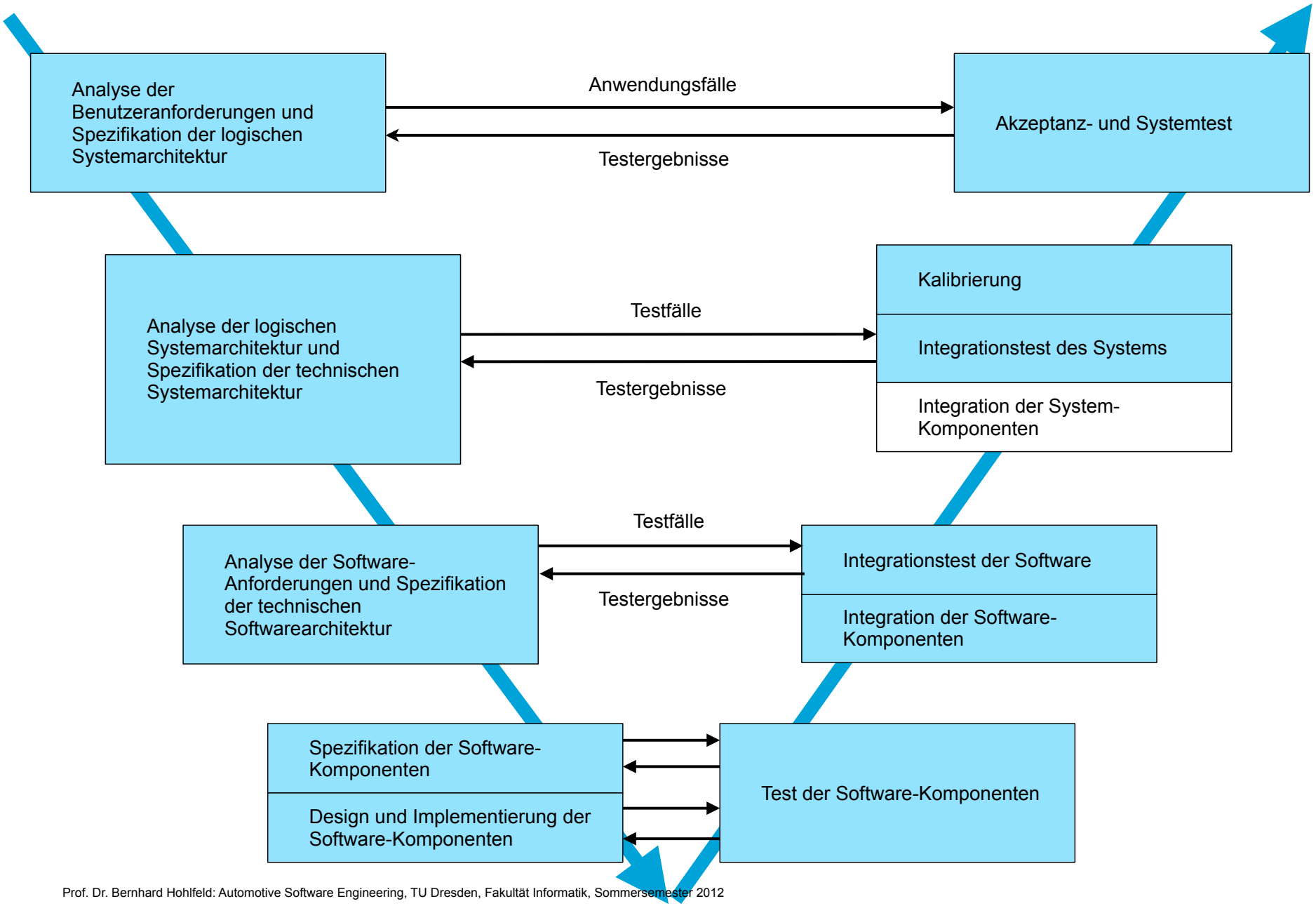
- Statische Prüfung bei der Zusammenführung von Software-Komponenten

- Teilweise manuell, teilweise automatisiert
- Schnittstellenspezifikation eingehalten?
- Namensraum für Variable eingehalten?
- Speicherlayout

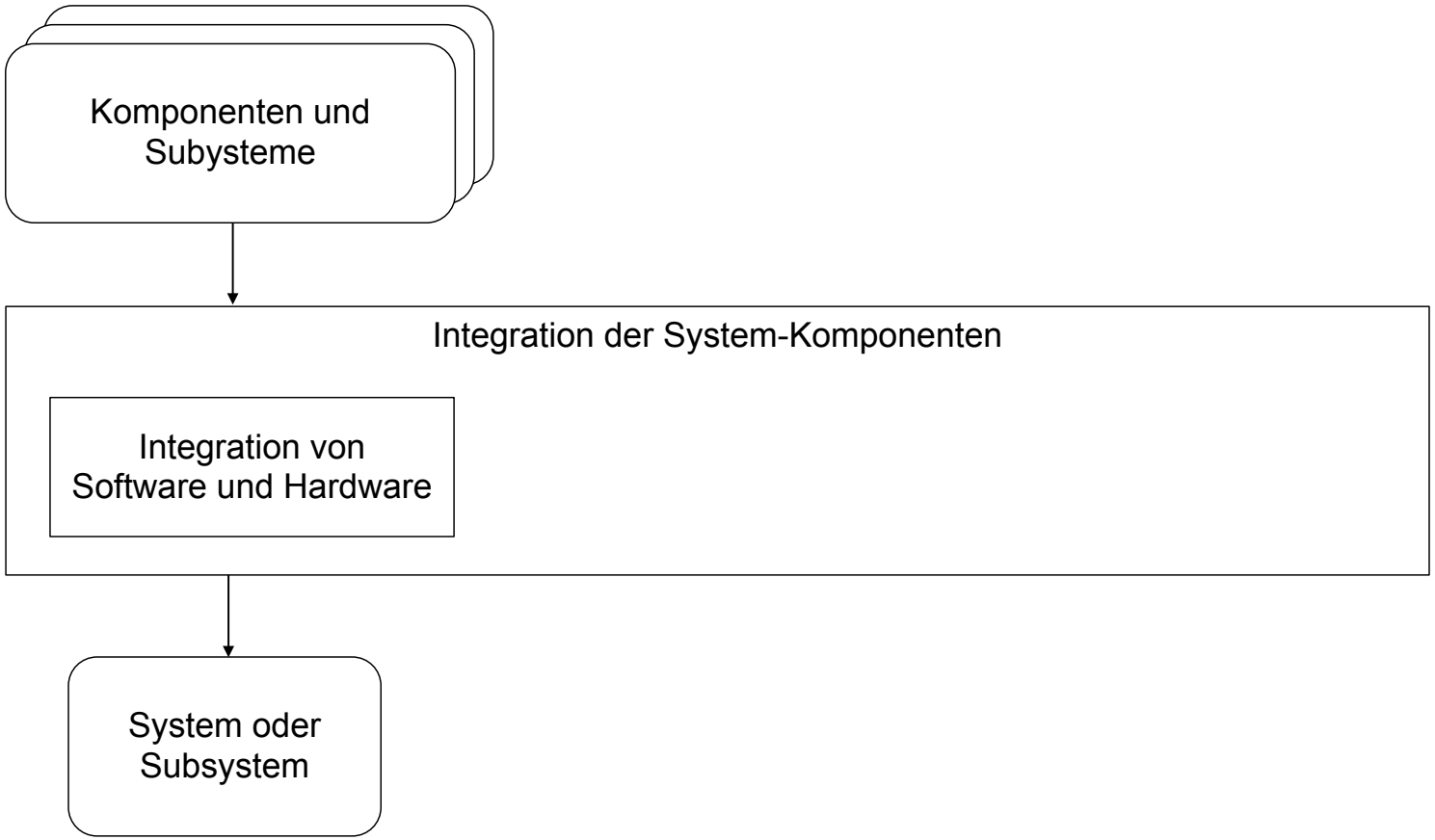
6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)

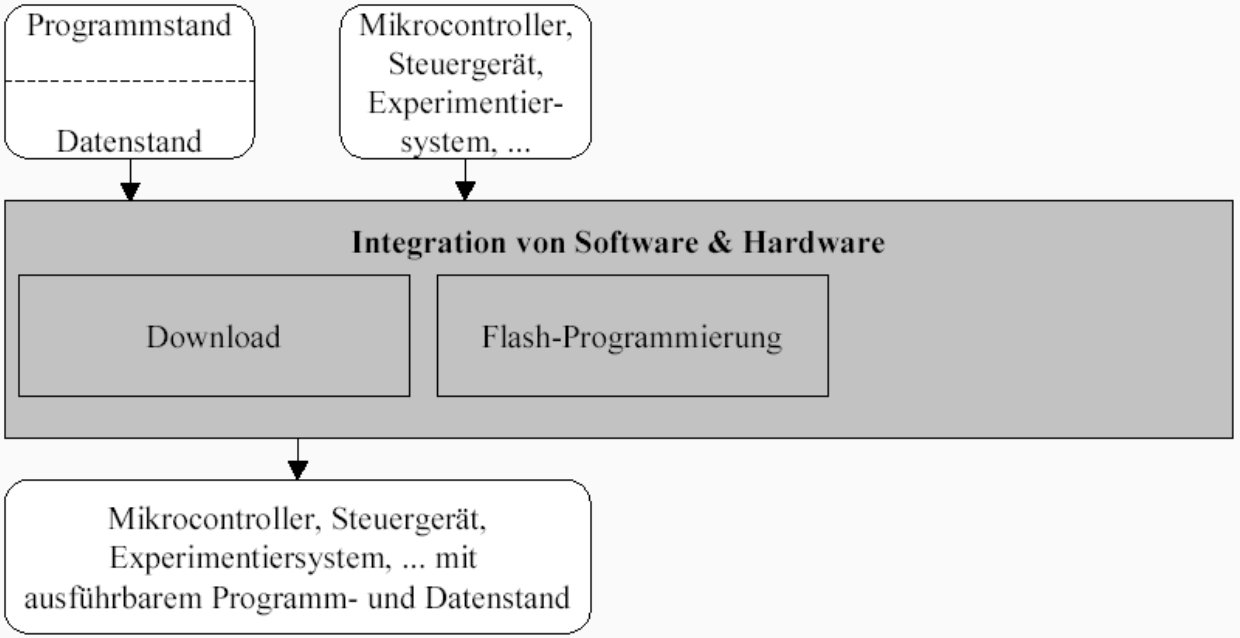


Integration der System-Komponenten (Nach Schäuffele, Zurawka)



Integration der System-Komponenten

Integration von Software und Hardware

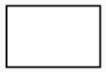
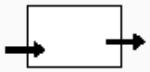
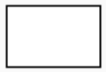
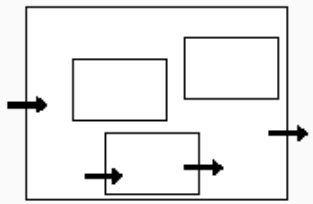
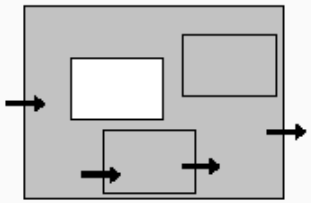
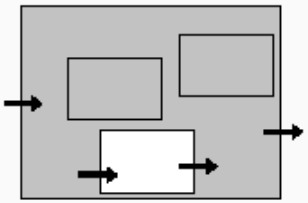
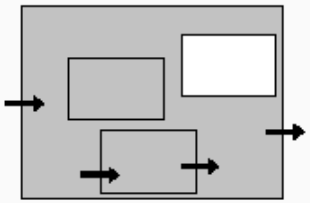
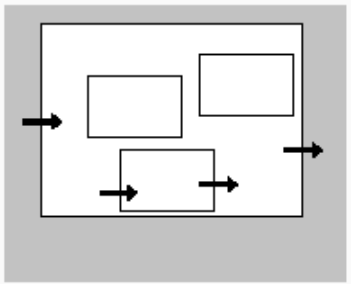


Integration von Steuergeräten, Sollwertgebern, Sensoren und Aktuatoren

- Abnahmetests für vom Zulieferer gelieferte Komponenten oder Subsysteme
- nur eingeschränkte Testmöglichkeiten beim Zulieferer
- Komponentenintegration als Synchronisationspunkt

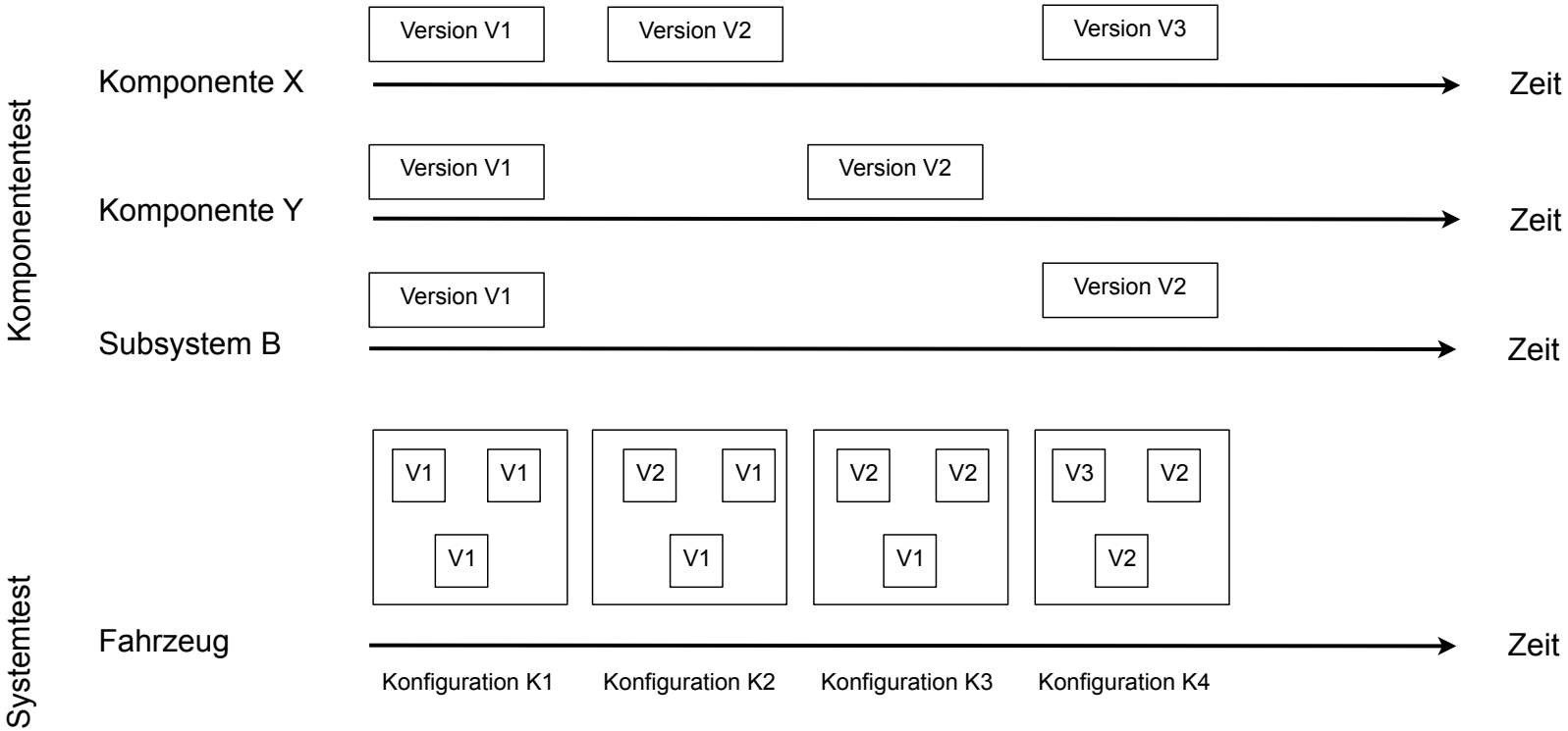
Integration der System-Komponenten

Unterschiedliche Umgebungen für Komponenten, Subsysteme und Systeme

Lieferant 1	Lieferant 2	Lieferant 3	Fahrzeughersteller
Komponente X	Subsystem B	Komponente Y	Fahrzeug
			
Komponenten-umgebung	Subsystem-umgebung	Komponenten-umgebung	Systemumgebung
			

Integration der System-Komponenten (Nach Schäuffele, Zurawka)

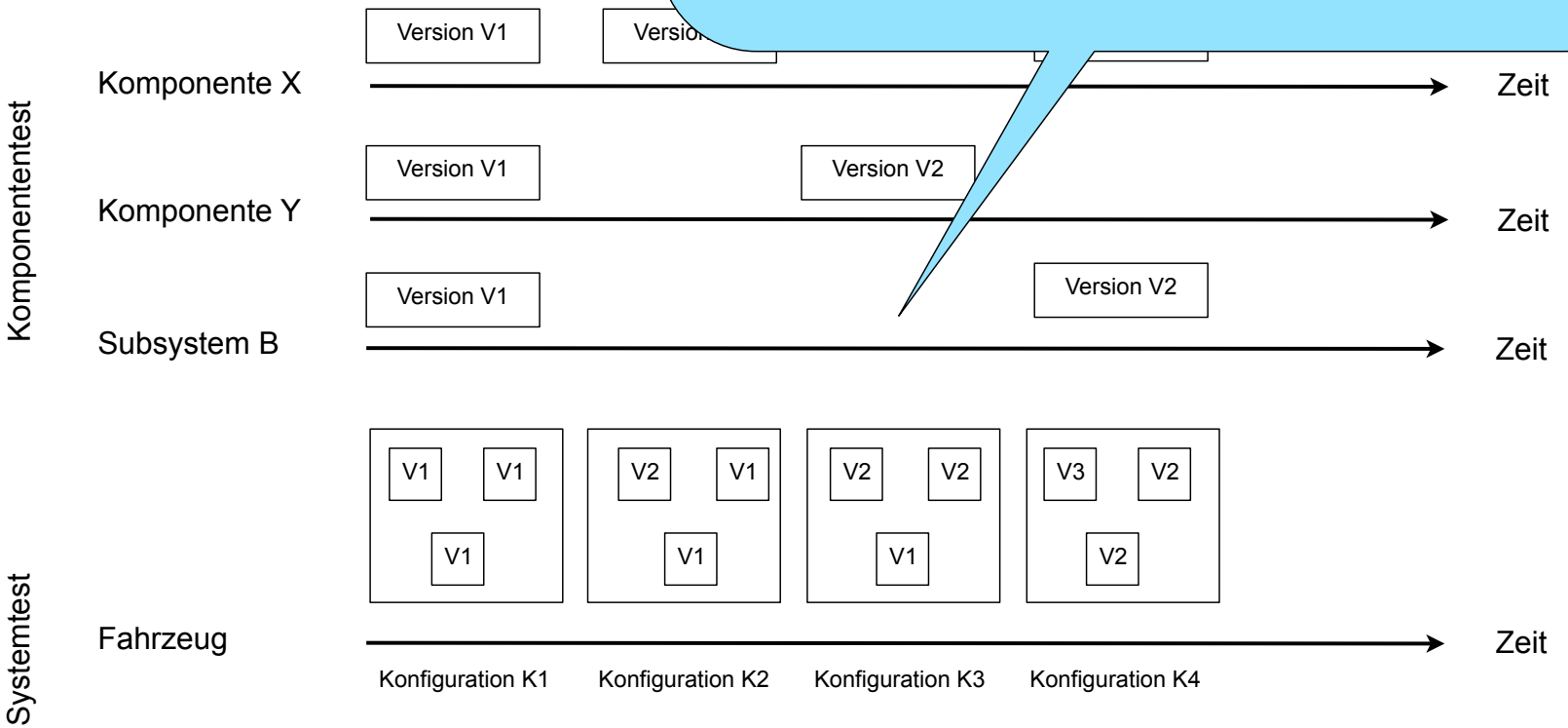
- Abhängigkeiten zwischen Komponenten- und Systemtest



Integration der System-Komponenten (Nach Schäuffele, Zurawka)

- Abhängigkeiten zwischen Komponenten- un

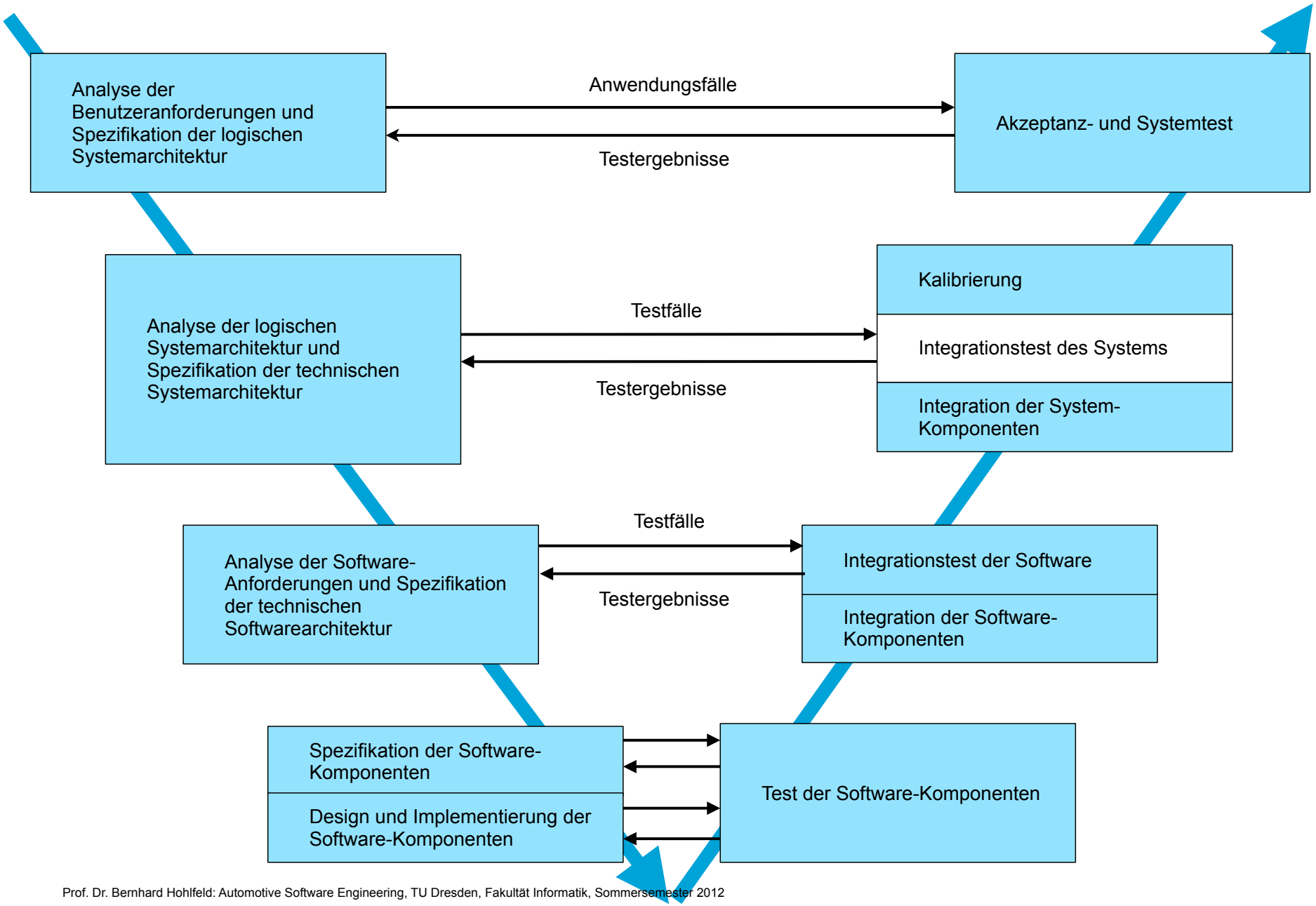
siehe auch
2. Unterstützungsprozesse
3. Konfigurationsmanangement
3. Versionen und Konfigurationen



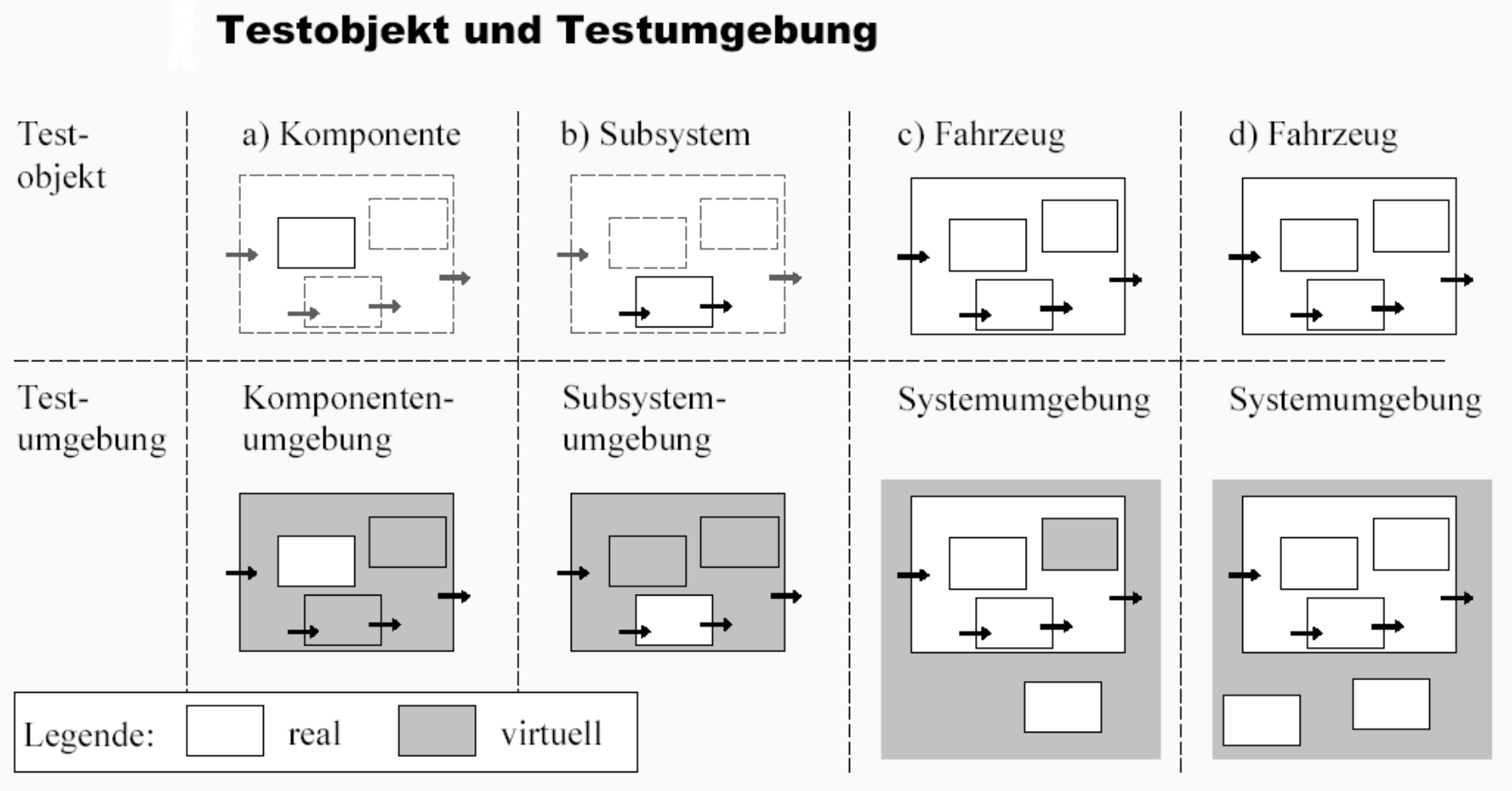
6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)



Integrationstest des Systems (Nach Schäuffele/Zurawka)


Testobjekt und Testumgebung
Test-objekt
a) Komponente
b) Subsystem
c) Fahrzeug
d) Fahrzeug
Test-umgebung
Komponenten-umgebung
Subsystem-umgebung
Systemumgebung
Systemumgebung
Legende:
real
virtuell

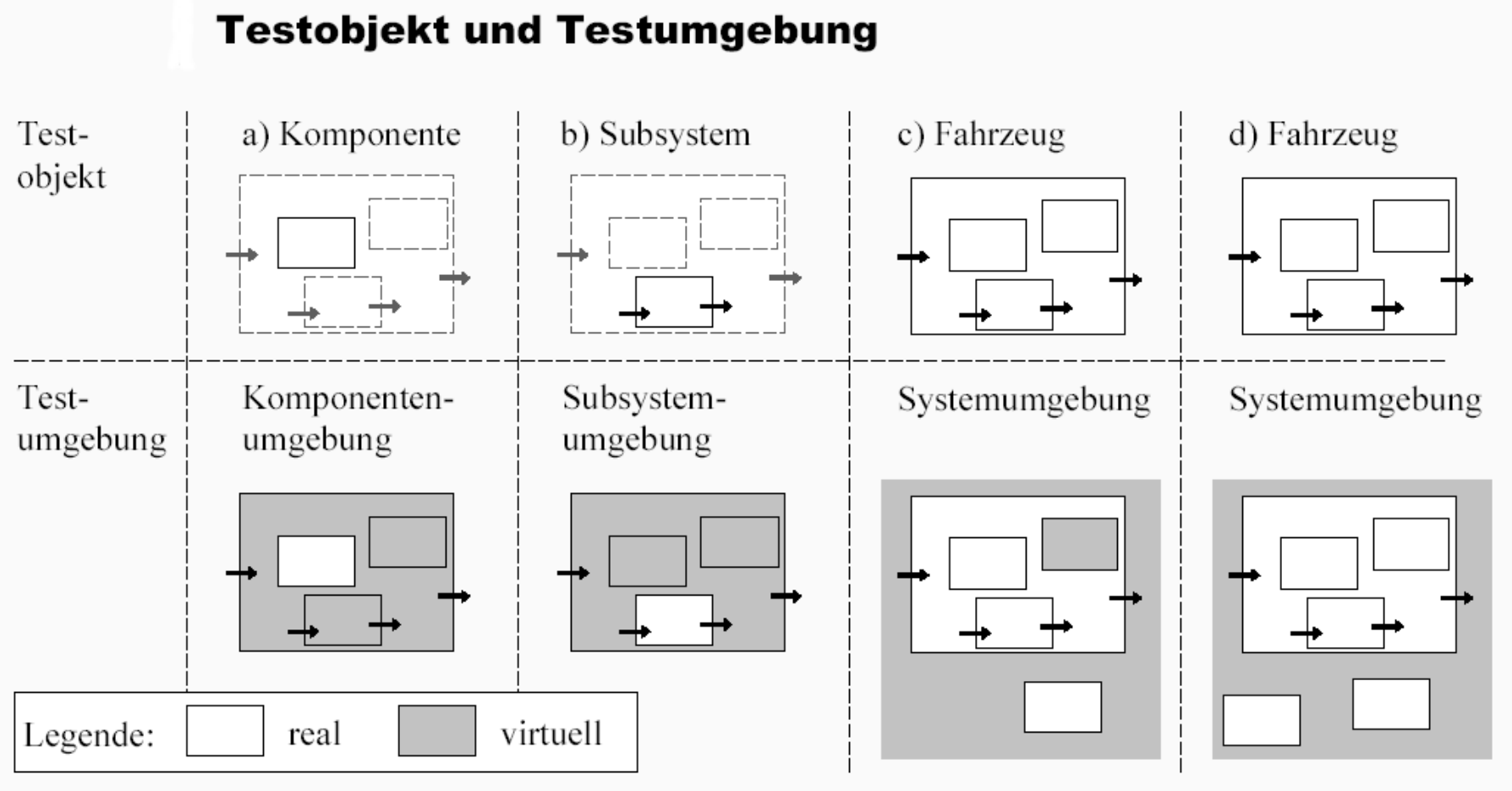
Entwicklungsprozesse
“Software/Hardware in the Loop”

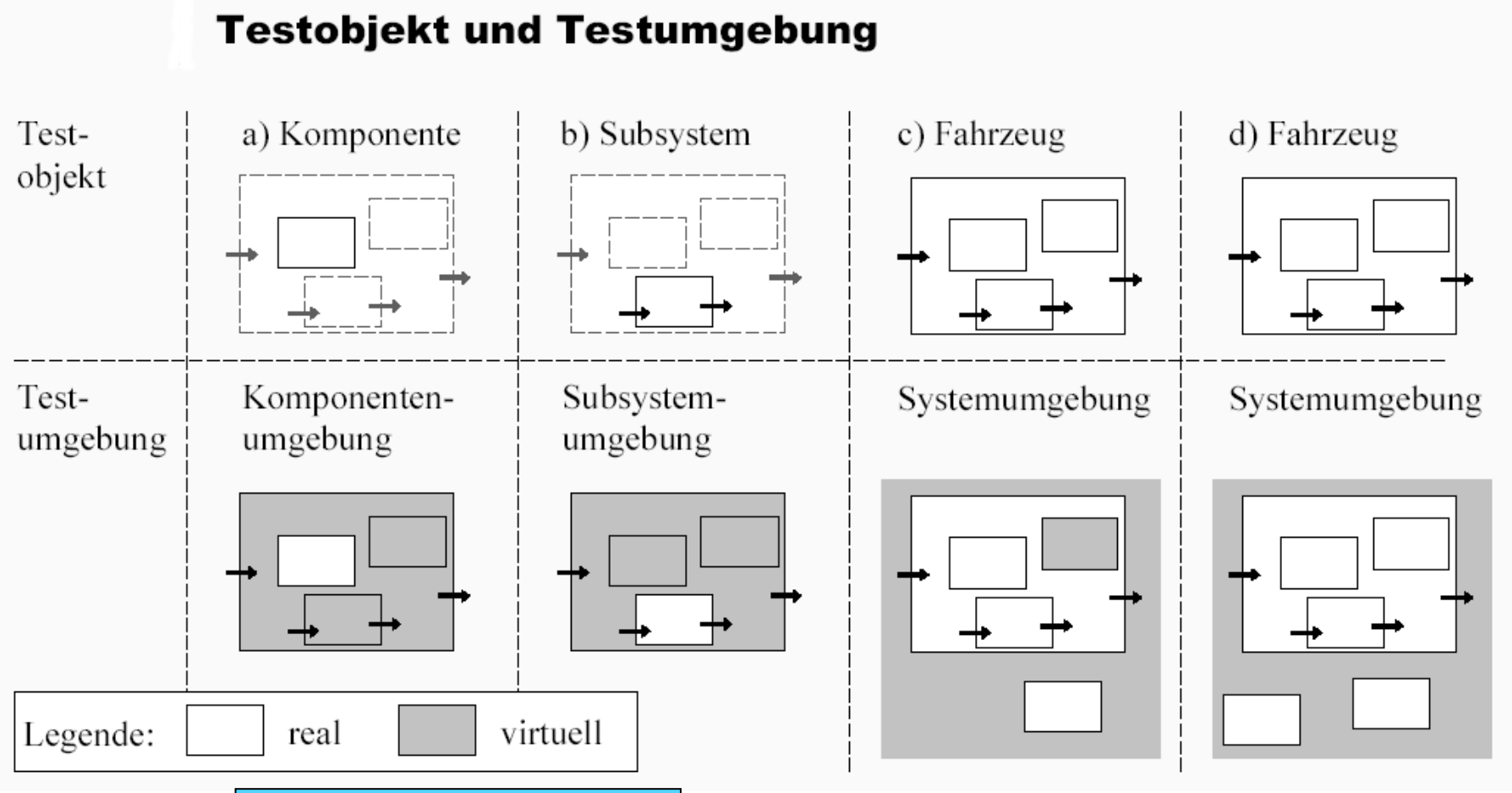
- Begriffsdefinitionen:
  - HIL Hardware in the Loop
    - Reale SG-Hardware wird in simulierter Fahrzeugumgebung gespeist, d.h. mit rechnergenerierten Sensor- und Bussignalen angesteuert.
  - SIL Software in the Loop
    - SG-Software „läuft“ auf simuliertem SG, das von simulierter Fahrzeugumgebung gespeist wird.

Steuergerät

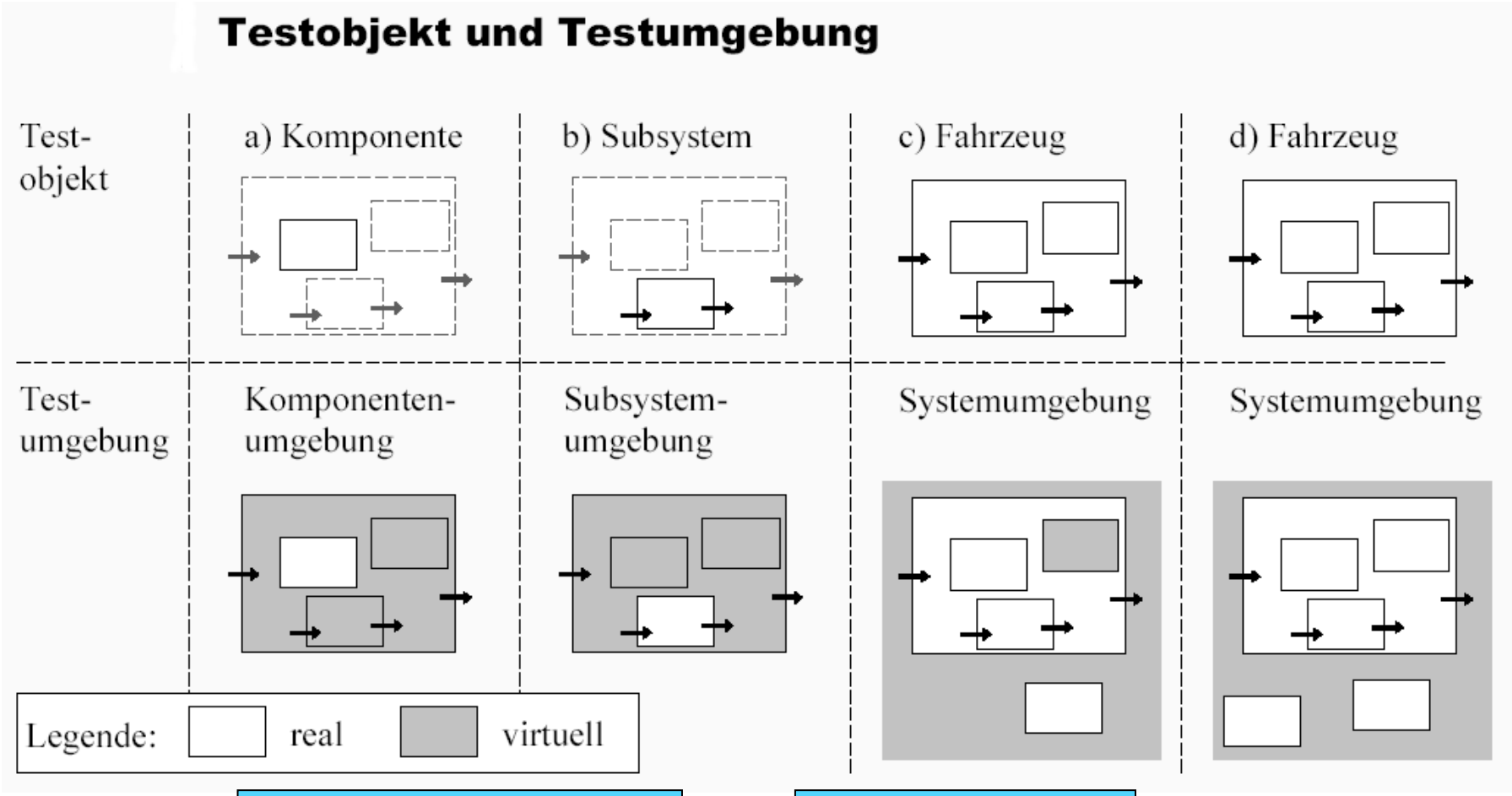
Fahrzeug / Umwelt

	simuliert	real
simuliert	**SIL**	**HIL**
real	**Prototyp**	**Fahrversuch**


Testobjekt und Testumgebung
Test-objekt
a) Komponente
b) Subsystem
c) Fahrzeug
d) Fahrzeug
Test-umgebung
Komponenten-umgebung
Subsystem-umgebung
Systemumgebung
Systemumgebung
Legende:
real
virtuell

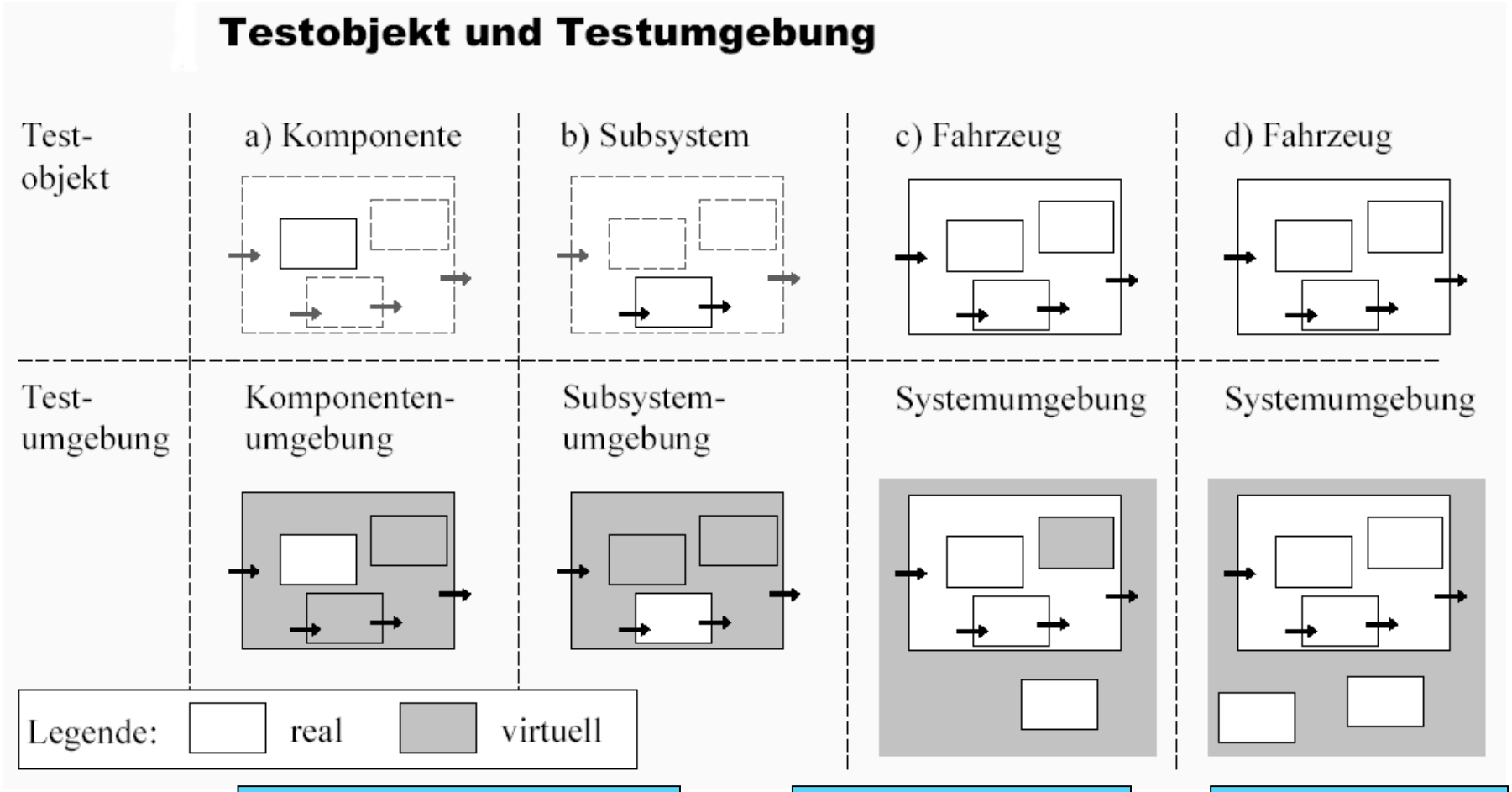


Hardware in the Loop HIL



Hardware in the Loop HIL

Prüfstand: Prototyp



Hardware in the Loop HIL

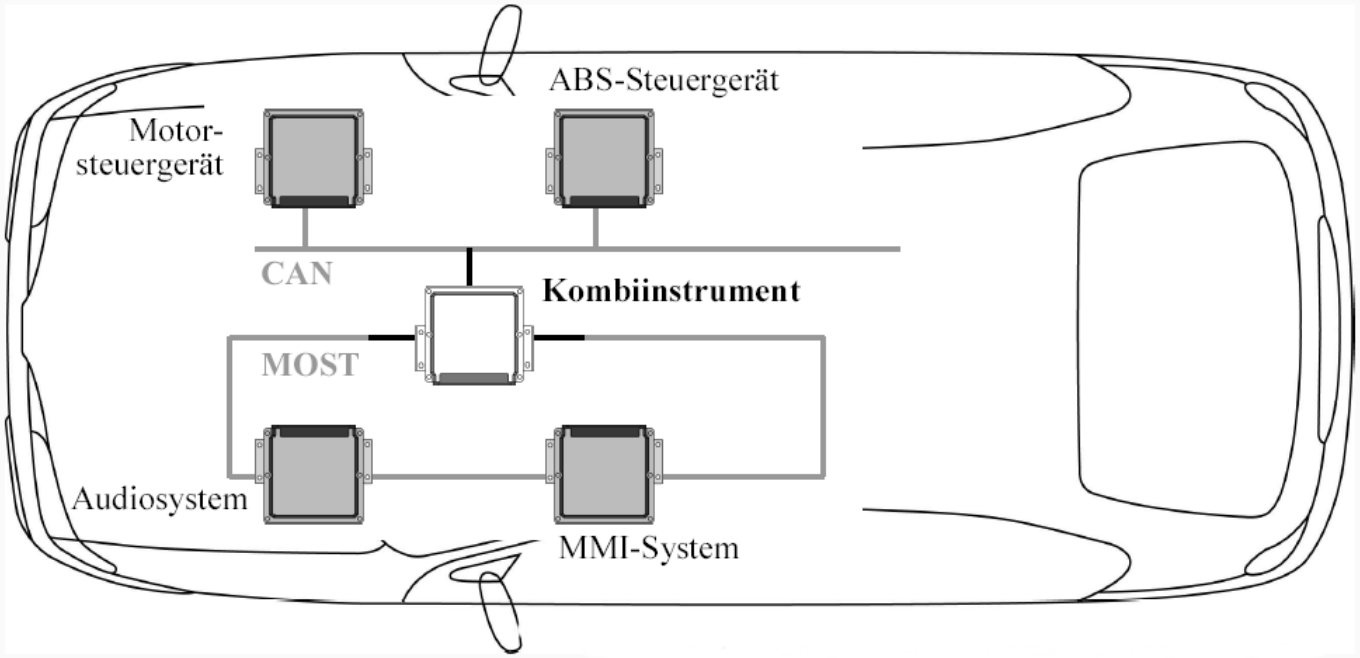
Prüfstand: Prototyp

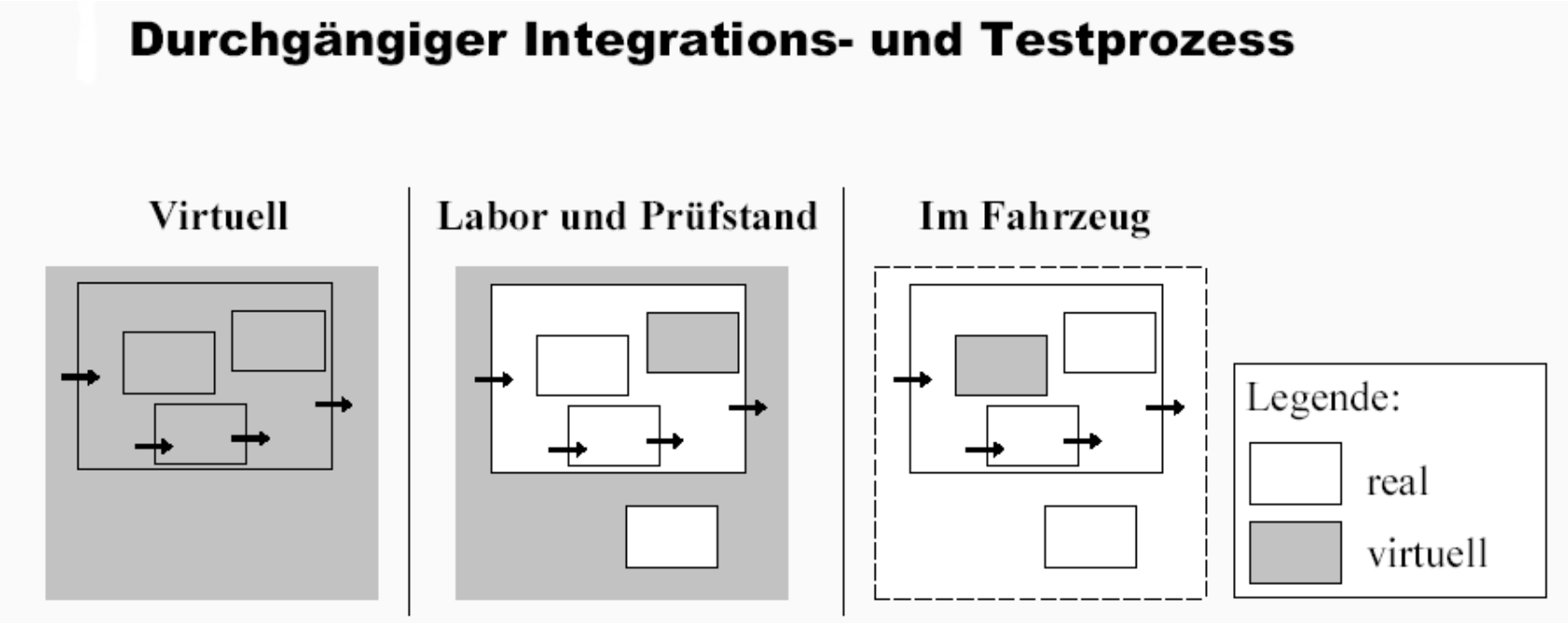
Prüfstand: Fahrversuch

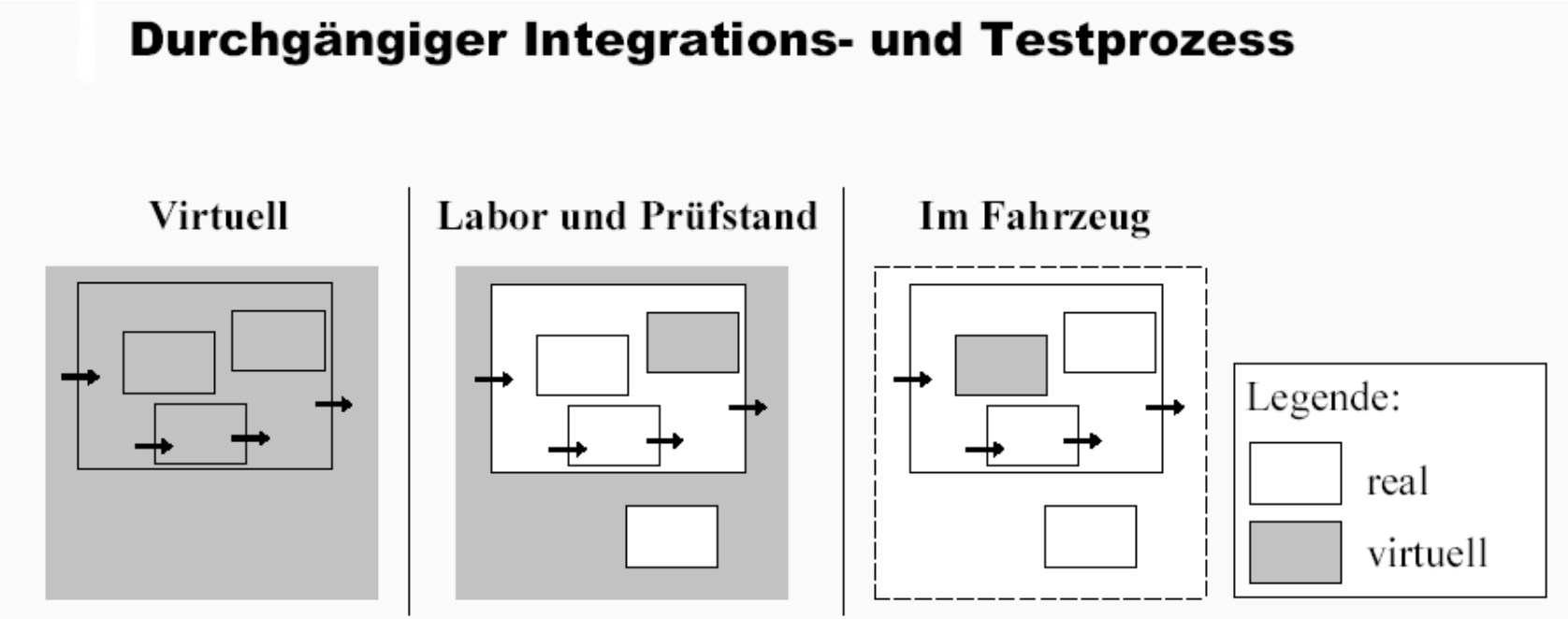
Vorteile virtueller Testumgebungen

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

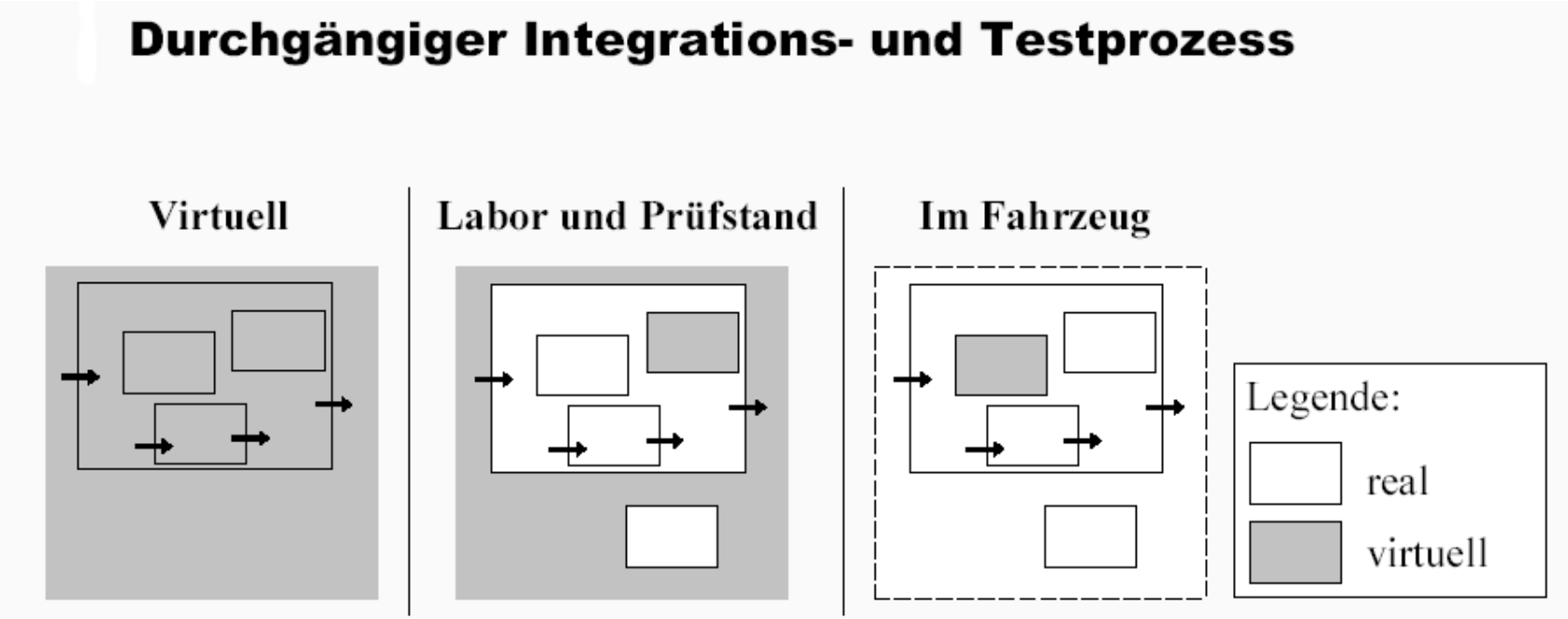
Virtuelle Netzwerkumgebung für das Kombiinstrument




Durchgängiger Integrations- und Testprozess
Virtuell
Labor und Prüfstand
Im Fahrzeug
Legende:
real
virtuell

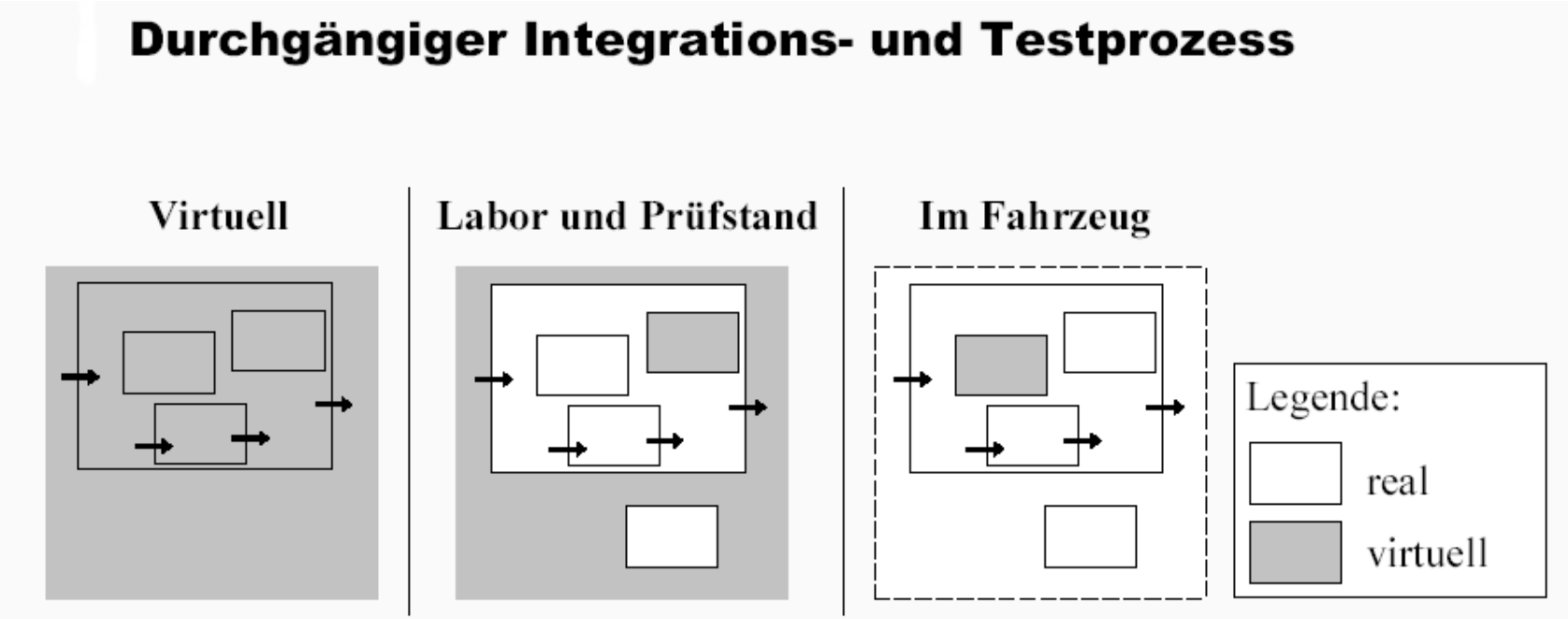


Software in the Loop SIL



Software in the Loop SIL

Prüfstand: Prototyp



Software in the Loop SIL

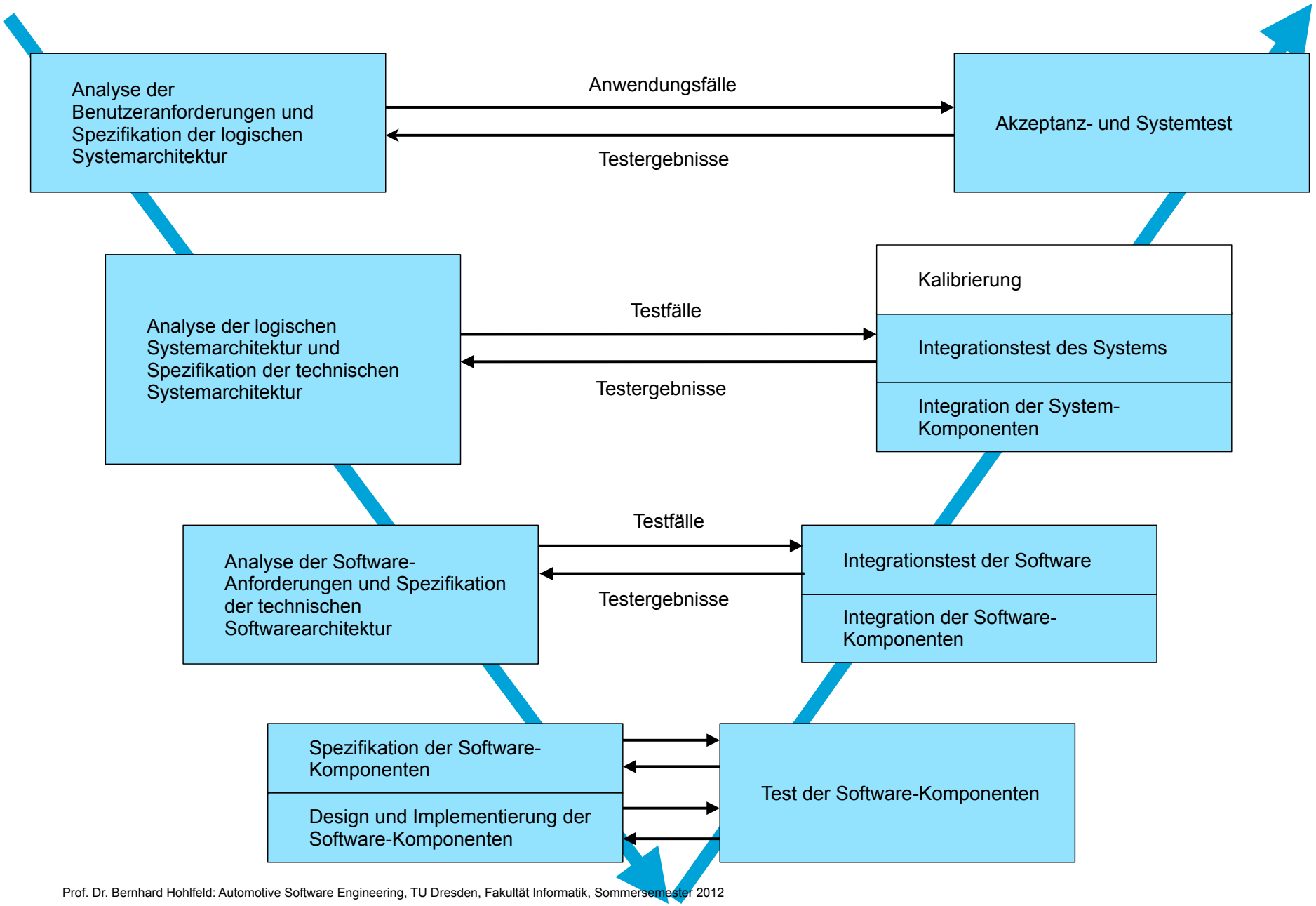
Prüfstand: Prototyp

Prototyp

6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

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

Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)

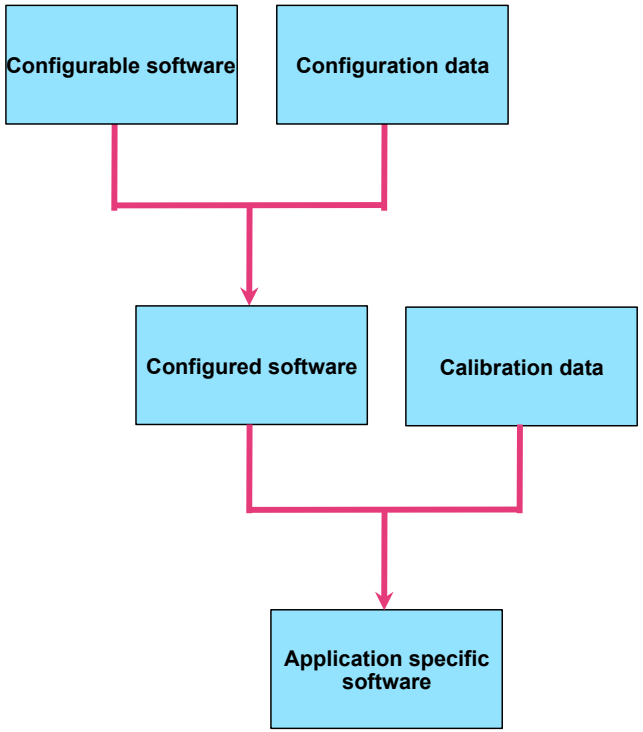


Kalibrierung
Quelle: Wikipedia

- 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.
- Messen der Abweichung vom „Normal“wert
- Korrektur
- Anpassen an die Physik

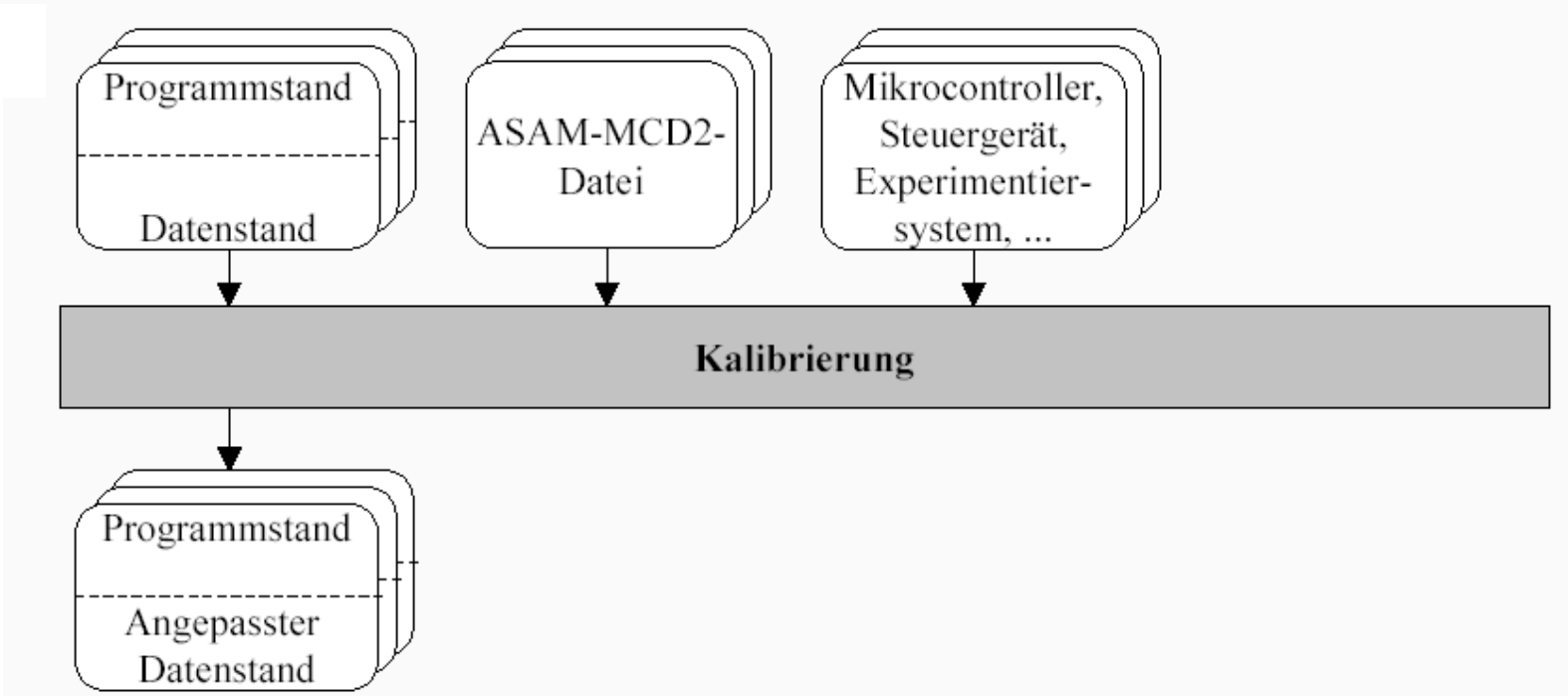
ISO 26262 Road vehicles — Functional safety
Part 6: Product development: software level

- The objective of software configuration is to enable controlled changes in the behaviour of the software for different applications.



Kalibrierung

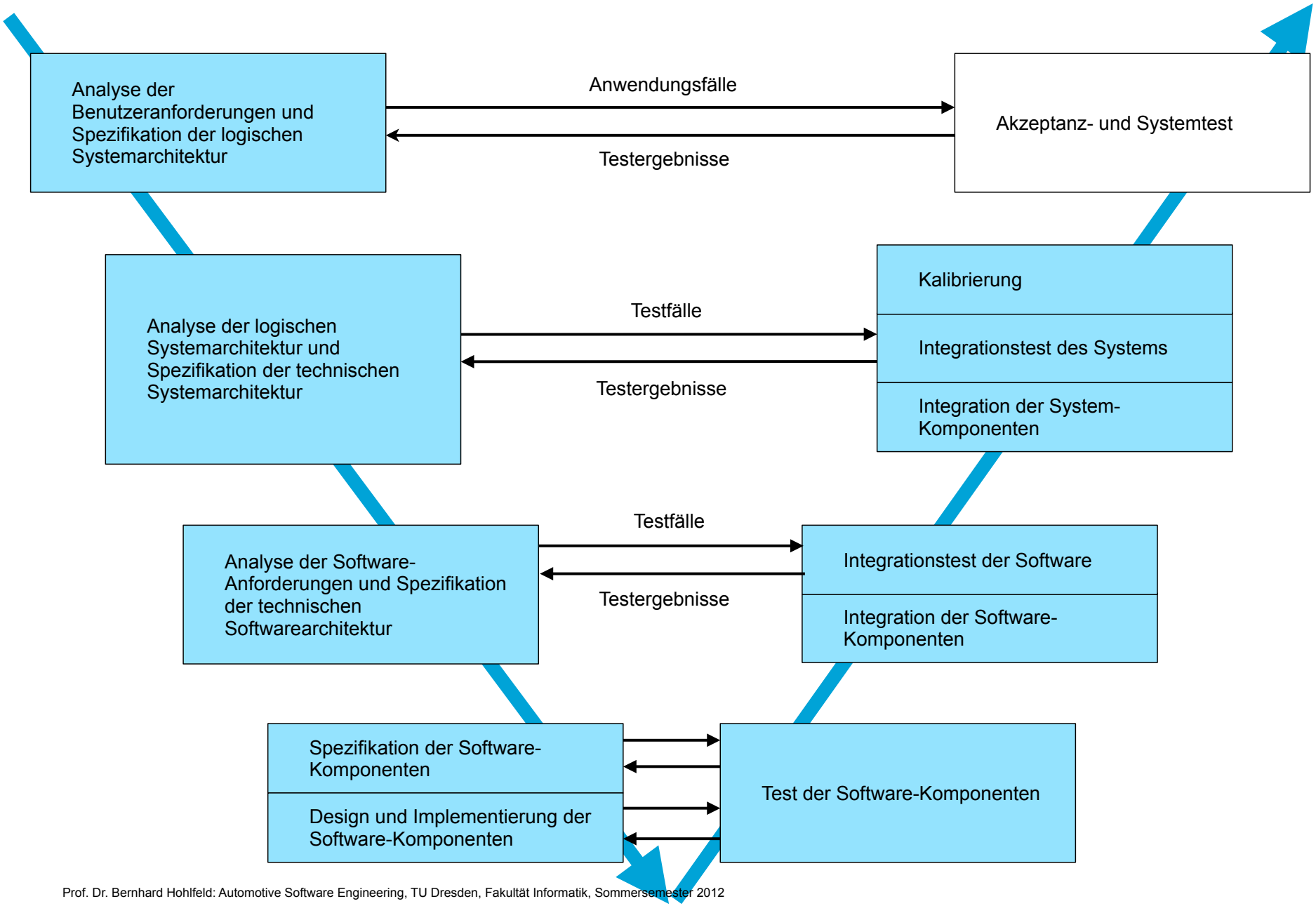
- fahrzeugindividuelle Einstellung der Parameter der Software-Funktionen
- häufig nur direkt im Fahrzeug bei laufenden Systemen
- *Kalibriersystem*: Steuergerät mit Off-Board-Schnittstelle zu einem Meß- und Kalibrierwerkzeug
- Datenstände in einem Festwertspeicher (ROM, EEPROM oder Flash)



6. SW-Entwicklung / 1. Kernprozess
Kernprozess zur Entwicklung von elektronischen Systemen und Software

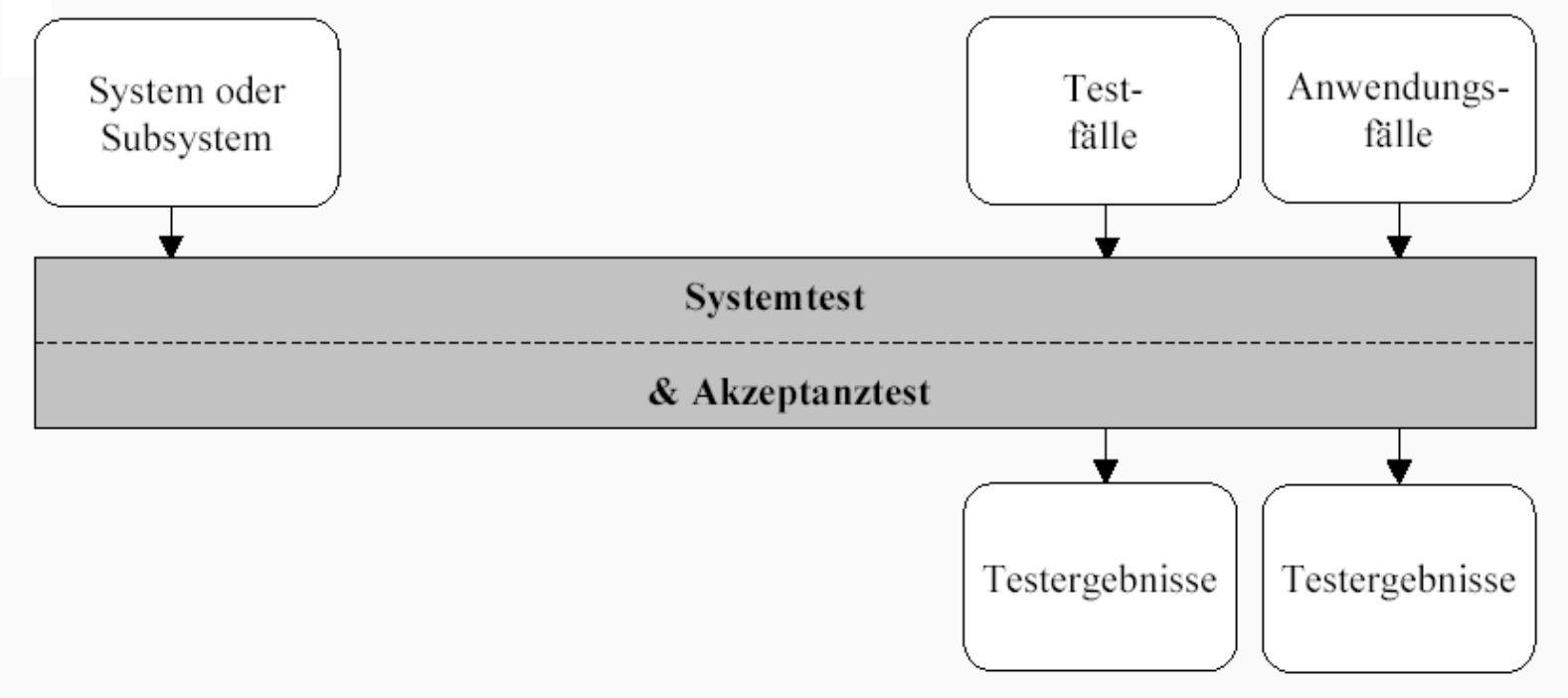
1. Grundbegriffe
2. Entwicklungsobjekt: Kombiinstrument
3. Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur
4. Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur
5. Analyse der Software-Anforderungen und Spezifikation der technischen 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**

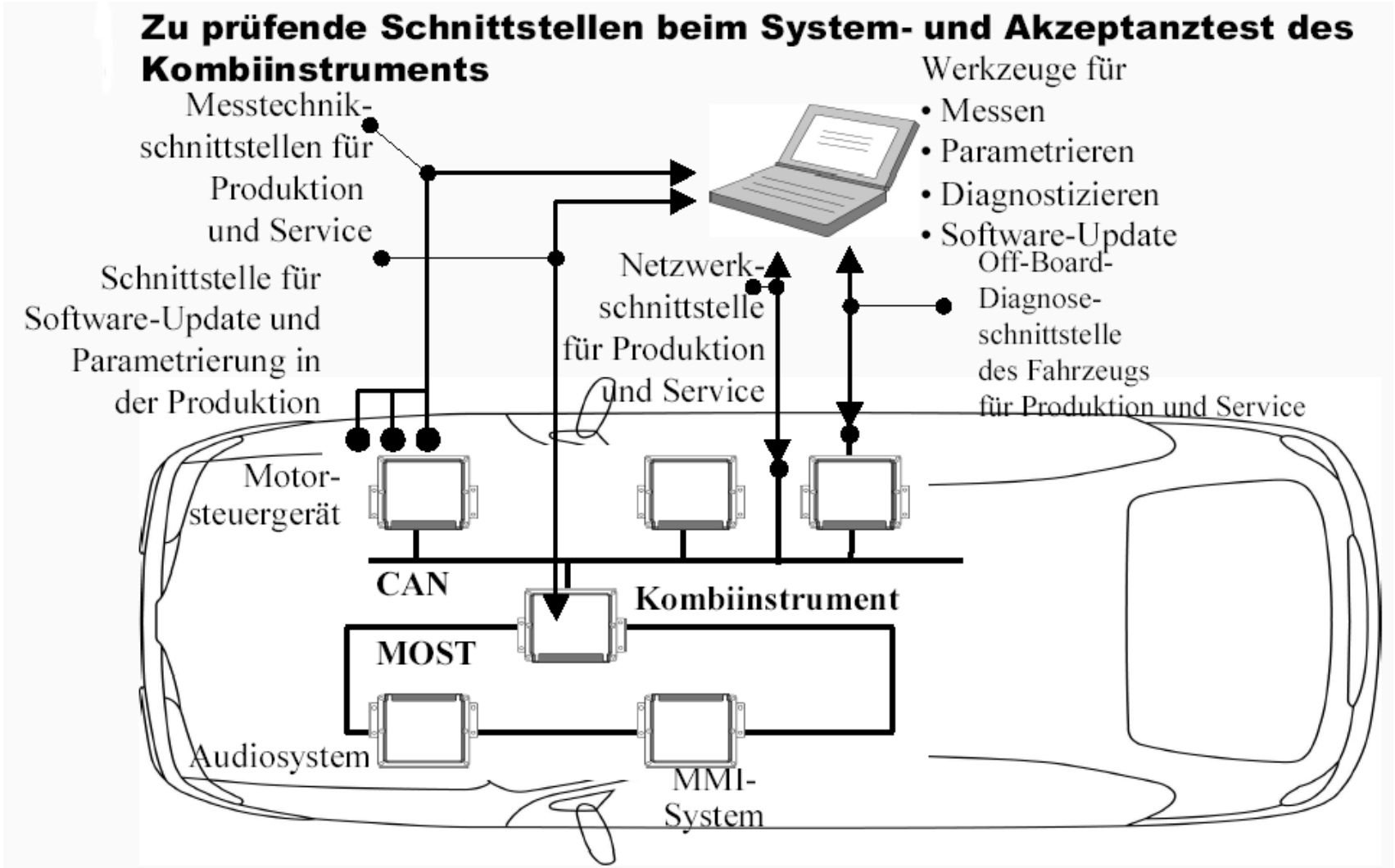
Kernprozess zur Entwicklung von elektronischen Systemen und Software (Nach Schäuffele, Zurawka)

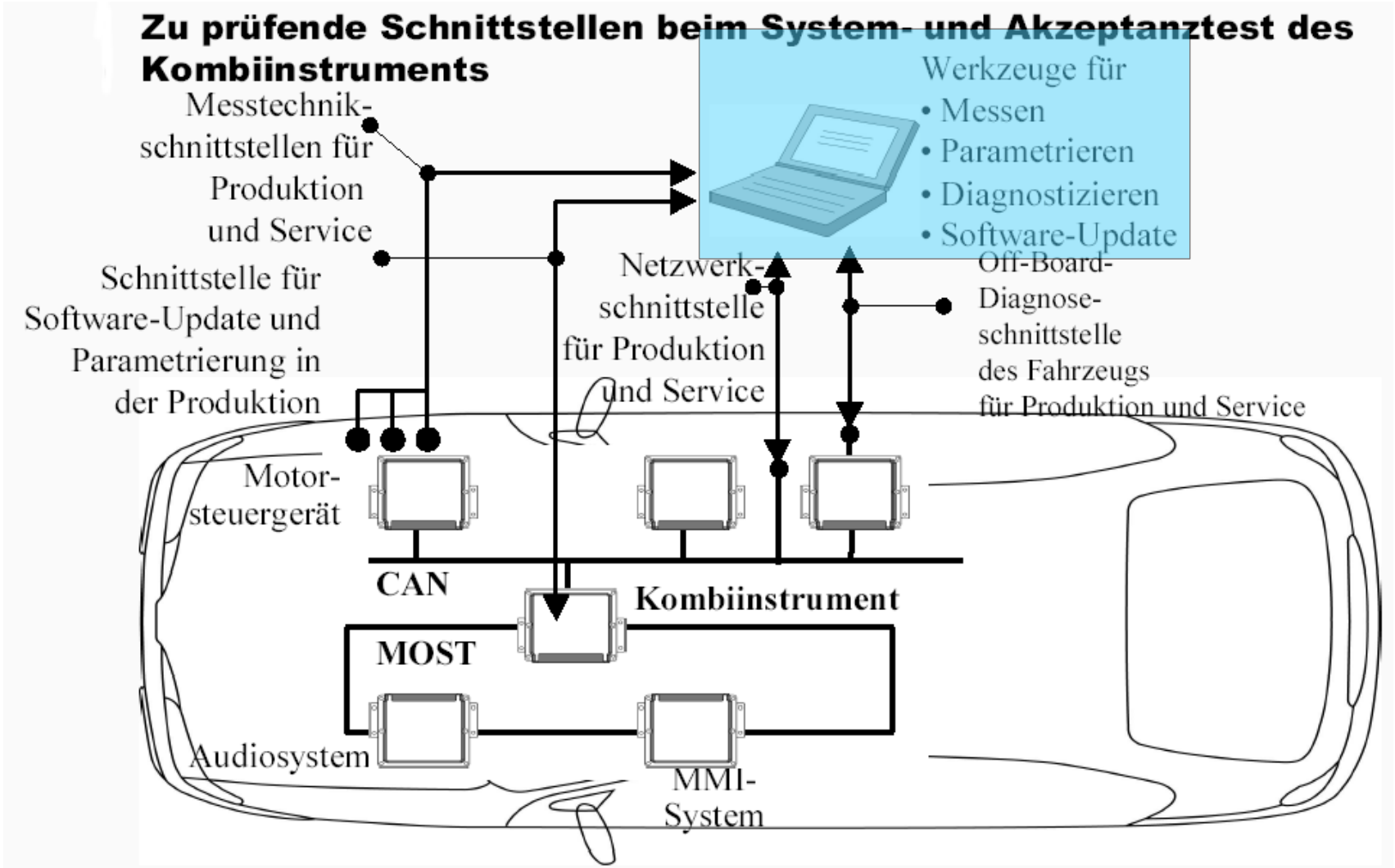


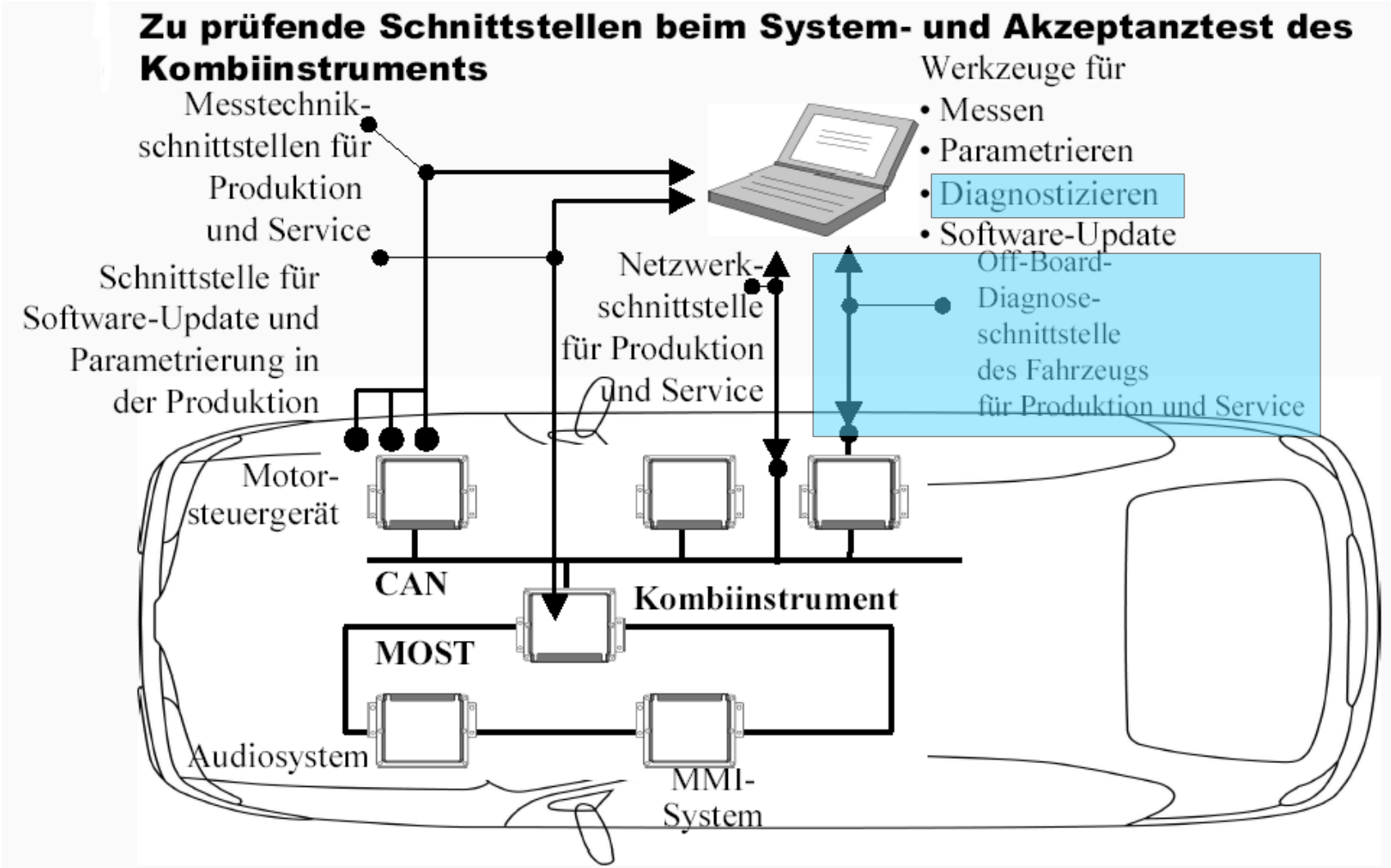
Prüfschritte im Fahrzeug

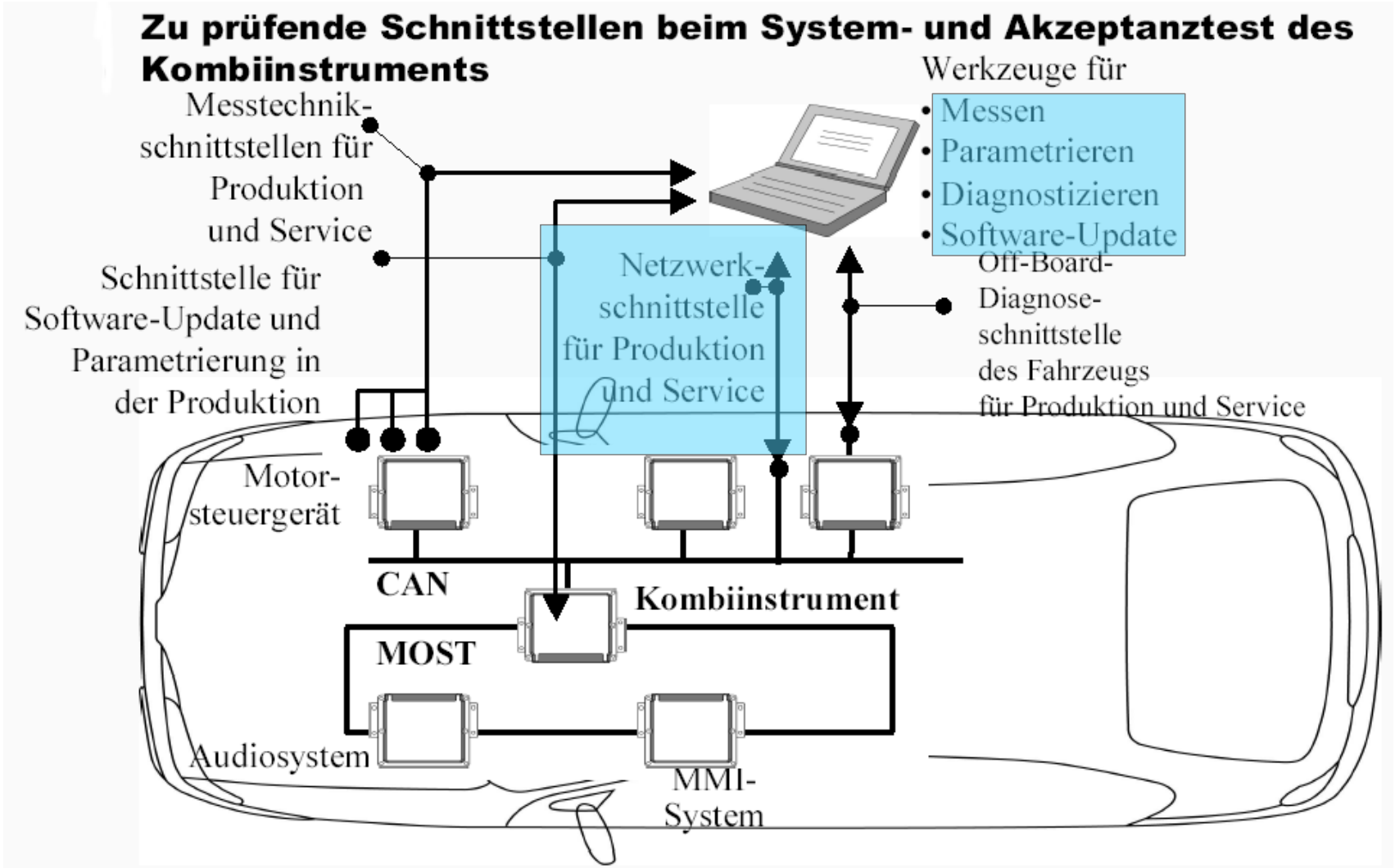
- Restrisiko durch Simulationsmodelle
- Systemtest im Fahrversuch:
  Akzeptanztest in der realen Betriebsumgebung
- fahrzeugtauglicher Zugang zu Steuergeräten und Netzwerken
- mobile Messtechnik

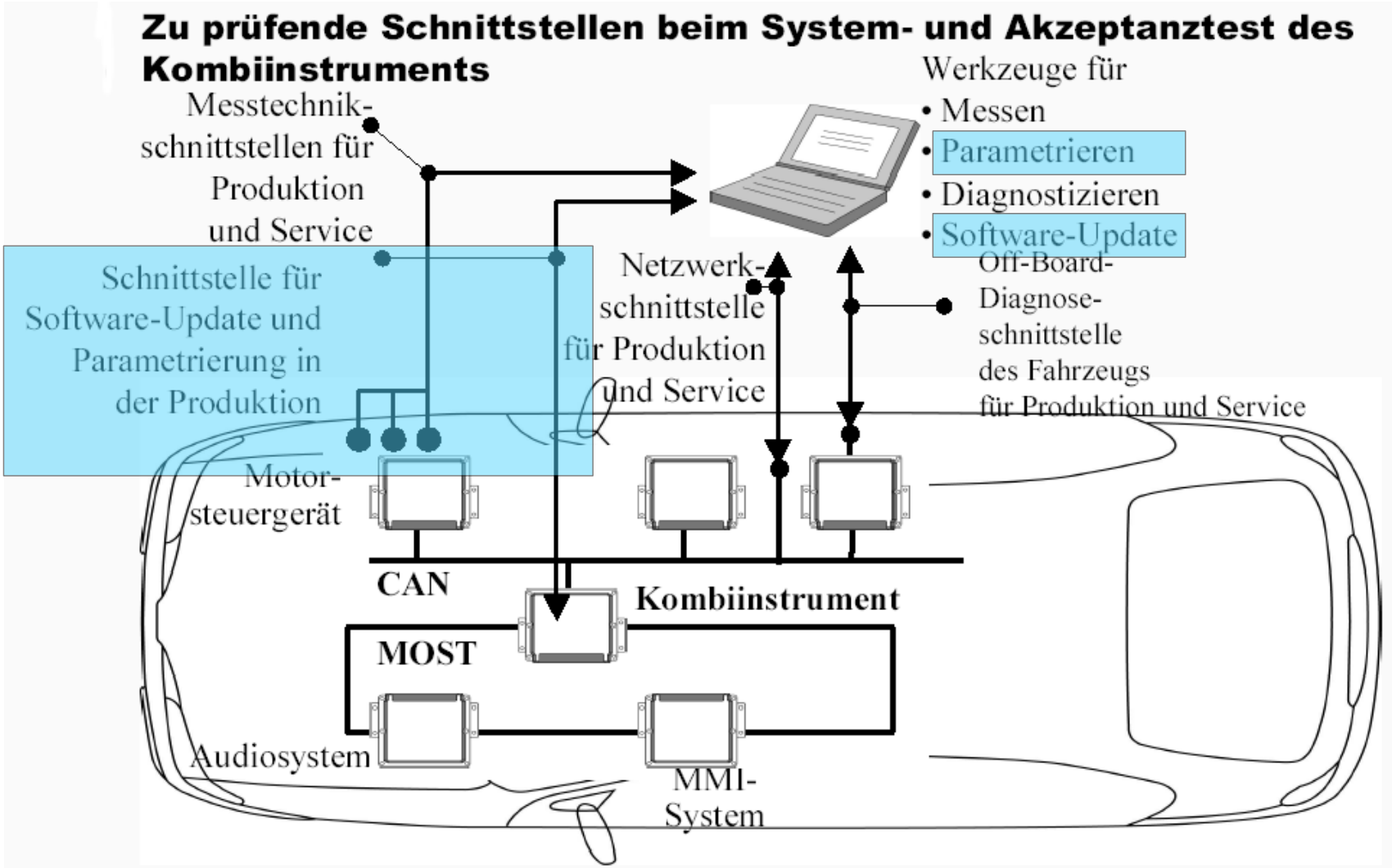


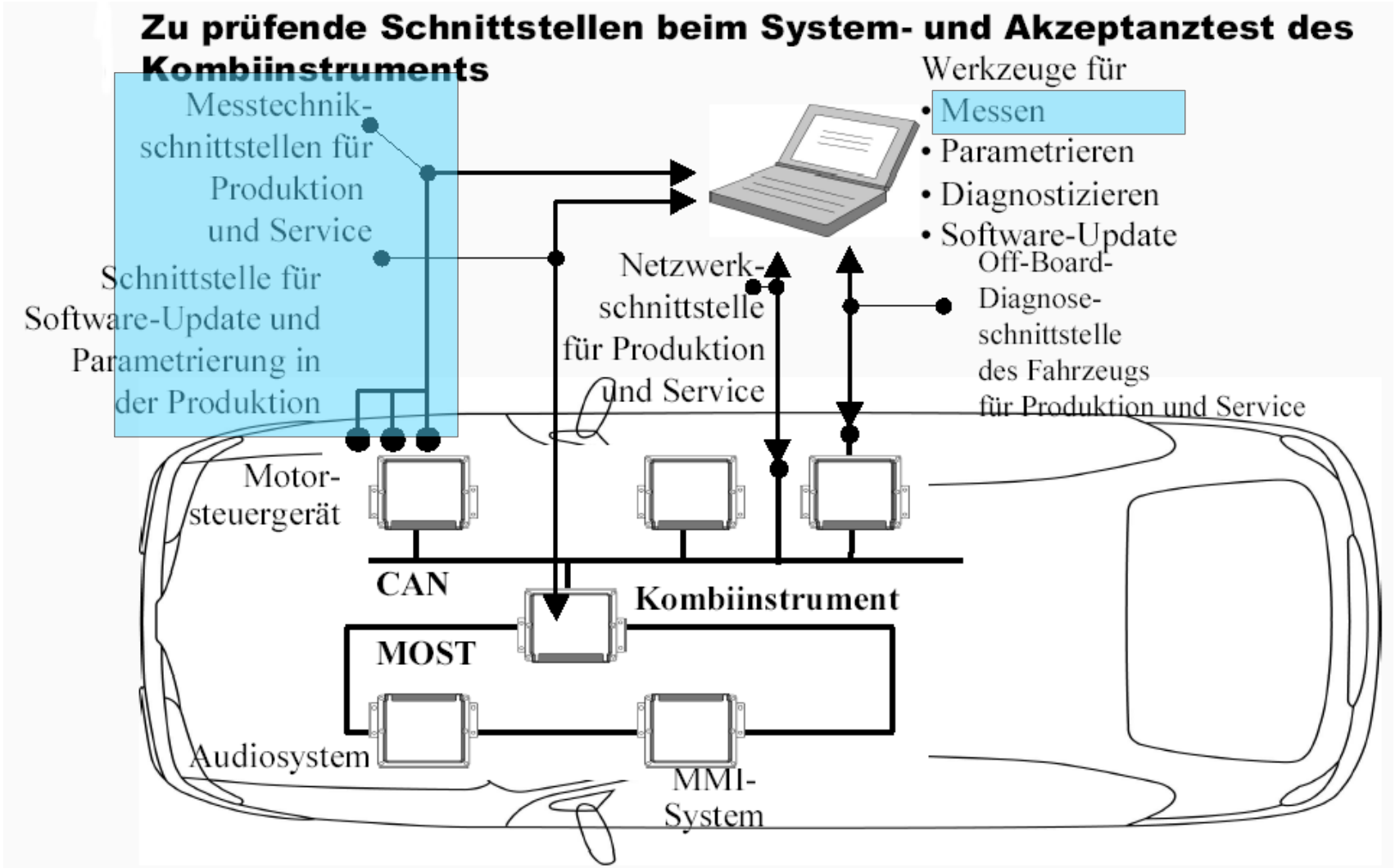

Zu prüfende Schnittstellen beim System- und Akzeptanztest des Kombiinstruments
Werkzeuge für
• Messen
• Parametrieren
• Diagnostizieren
• Software-Update
Messtechnik-schnittstellen für Produktion und Service
Schnittstelle für Software-Update und Parametrierung in der Produktion
Netzwerk-schnittstelle für Produktion und Service
Off-Board-Diagnose-schnittstelle des Fahrzeugs für Produktion und Service
Motor-steuergerät
CAN
Kombiinstrument
MOST
Audiosystem
MMI-System


Zu prüfende Schnittstellen beim System- und Akzeptanztest des Kombiinstruments
Werkzeuge für
• Messen
• Parametrieren
• Diagnostizieren
• Software-Update
Messtechnik-schnittstellen für Produktion und Service
Schnittstelle für Software-Update und Parametrierung in der Produktion
Netzwerk-schnittstelle für Produktion und Service
Off-Board-Diagnose-schnittstelle des Fahrzeugs für Produktion und Service
Motor-steuergerät
CAN
Kombiinstrument
MOST
Audiosystem
MMI-System


Zu prüfende Schnittstellen beim System- und Akzeptanztest des Kombiinstruments
Werkzeuge für
• Messen
• Parametrieren
• Diagnostizieren
• Software-Update
Messtechnik-schnittstellen für Produktion und Service
Schnittstelle für Software-Update und Parametrierung in der Produktion
Netzwerk-schnittstelle für Produktion und Service
Off-Board-Diagnose-schnittstelle des Fahrzeugs für Produktion und Service
Motor-steuergerät
CAN
Kombiinstrument
MOST
Audiosystem
MMI-System


Zu prüfende Schnittstellen beim System- und Akzeptanztest des Kombiinstruments
Werkzeuge für
Messen
Parametrieren
Diagnostizieren
Software-Update
Messtechnik-schnittstellen für Produktion und Service
Schnittstelle für Software-Update und Parametrierung in der Produktion
Netzwerk-schnittstelle für Produktion und Service
Off-Board-Diagnose-schnittstelle des Fahrzeugs für Produktion und Service
Motor-steuergerät
CAN
Kombiinstrument
MOST
Audiosystem
MMI-System


Zu prüfende Schnittstellen beim System- und Akzeptanztest des Kombiinstruments
Werkzeuge für
• Messen
• Parametrieren
• Diagnostizieren
• Software-Update
Messtechnik-schnittstellen für Produktion und Service
Schnittstelle für Software-Update und Parametrierung in der Produktion
Netzwerk-schnittstelle für Produktion und Service
Off-Board-Diagnose-schnittstelle des Fahrzeugs für Produktion und Service
Motor-steuergerät
CAN
Kombiinstrument
MOST
Audiosystem
MMI-System


Zu prüfende Schnittstellen beim System- und Akzeptanztest des Kombiinstruments
Messtechnik-schnittstellen für Produktion und Service
Schnittstelle für Software-Update und Parametrierung in der Produktion
Werkzeuge für
• Messen
• Parametrieren
• Diagnostizieren
• Software-Update
Netzwerk-schnittstelle für Produktion und Service
Off-Board-Diagnose-schnittstelle des Fahrzeugs für Produktion und Service
Motor-steuergerät
CAN
Kombiinstrument
MOST
Audiosystem
MMI-System

Mobile Messtechnik

Kernprozess zur Entwicklung von elektronischen Systemen und Software - Zusammenfassung

- Kein Anspruch auf Vollständigkeit, sondern Darstellung der Vorgehensweise mit Beispielen
- Anpassung des Kernprozesses für konkrete Projekte nötig
- Beispiele
- Motorsteuergeräte
  - Kalibrierung wichtig
  - Vielzahl verschiedener Funktionen
- Karosserieelektronik
  - Kalibrierung untergeordnet
  - Verteilte und vernetzte Systeme