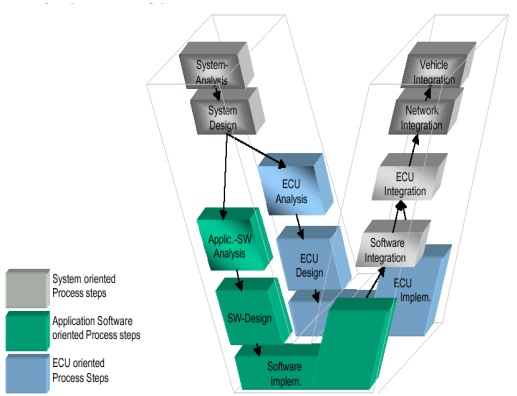
TECHNISCHE UNIVERSITÄT DRESDEN

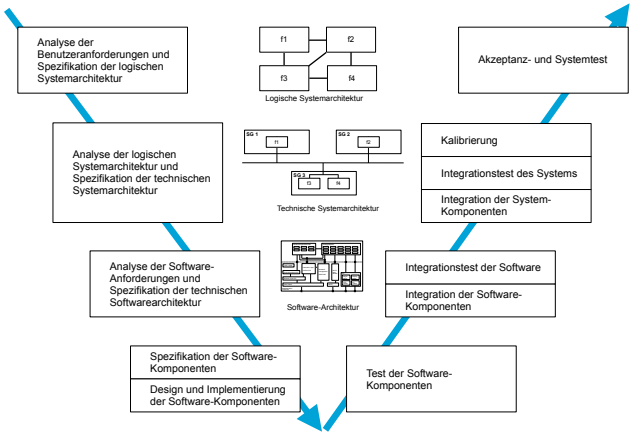
Vorlesung
Automotive Software Engineering
Teil 6 SW-Entwicklung (1)
Sommersemester 2015

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

Das V-Modell ist schon ziemlich alt ...

Das V-Modell ist schon ziemlich alt ...

Das V-Modell ist schon ziemlich alt ...

Das V-Modell ist schon ziemlich alt ...

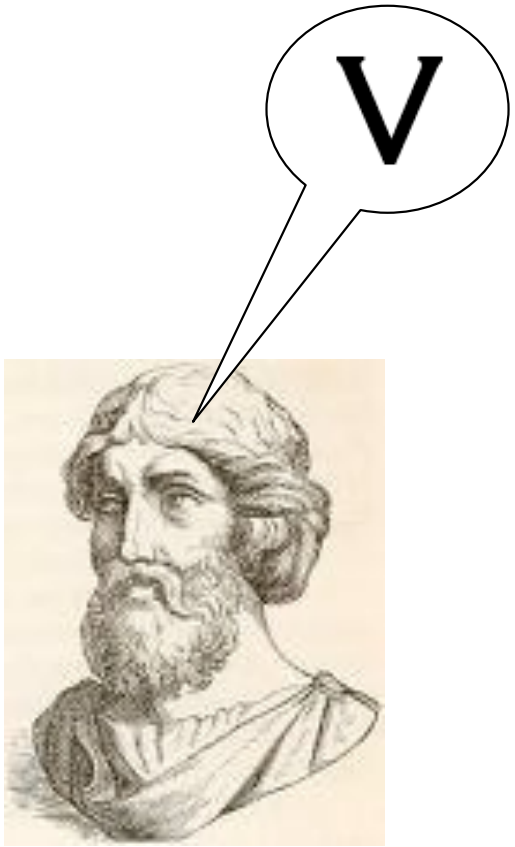
Das V-Modell ist schon ziemlich alt ...

Das V-Modell ist schon ziemlich alt ...

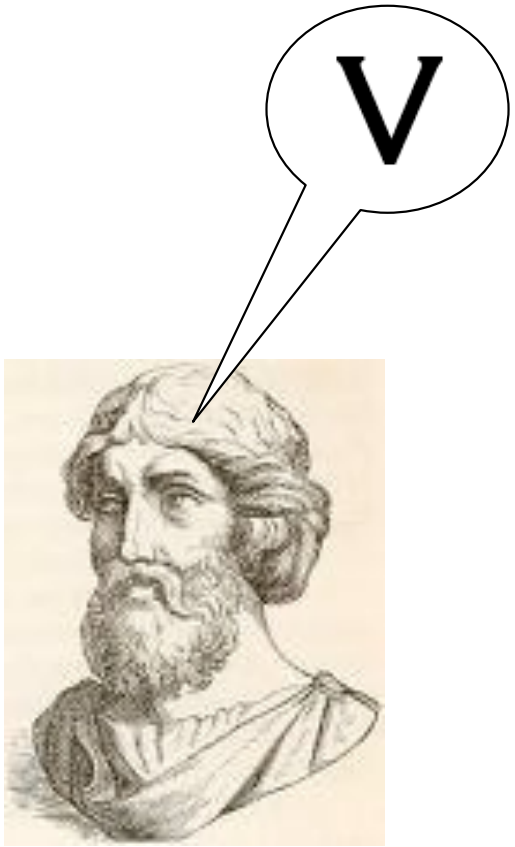
... und wurde weiterentwickelt und verfeinert

... und wurde weiterentwickelt und verfeinert

... und wurde weiterentwickelt und verfeinert

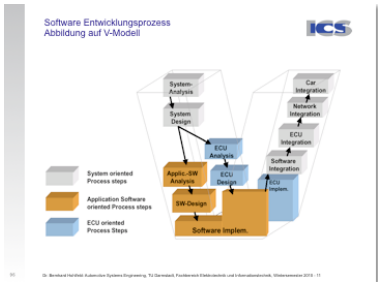


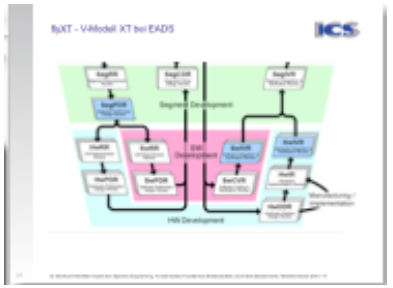
... und wurde weiterentwickelt und verfeinert



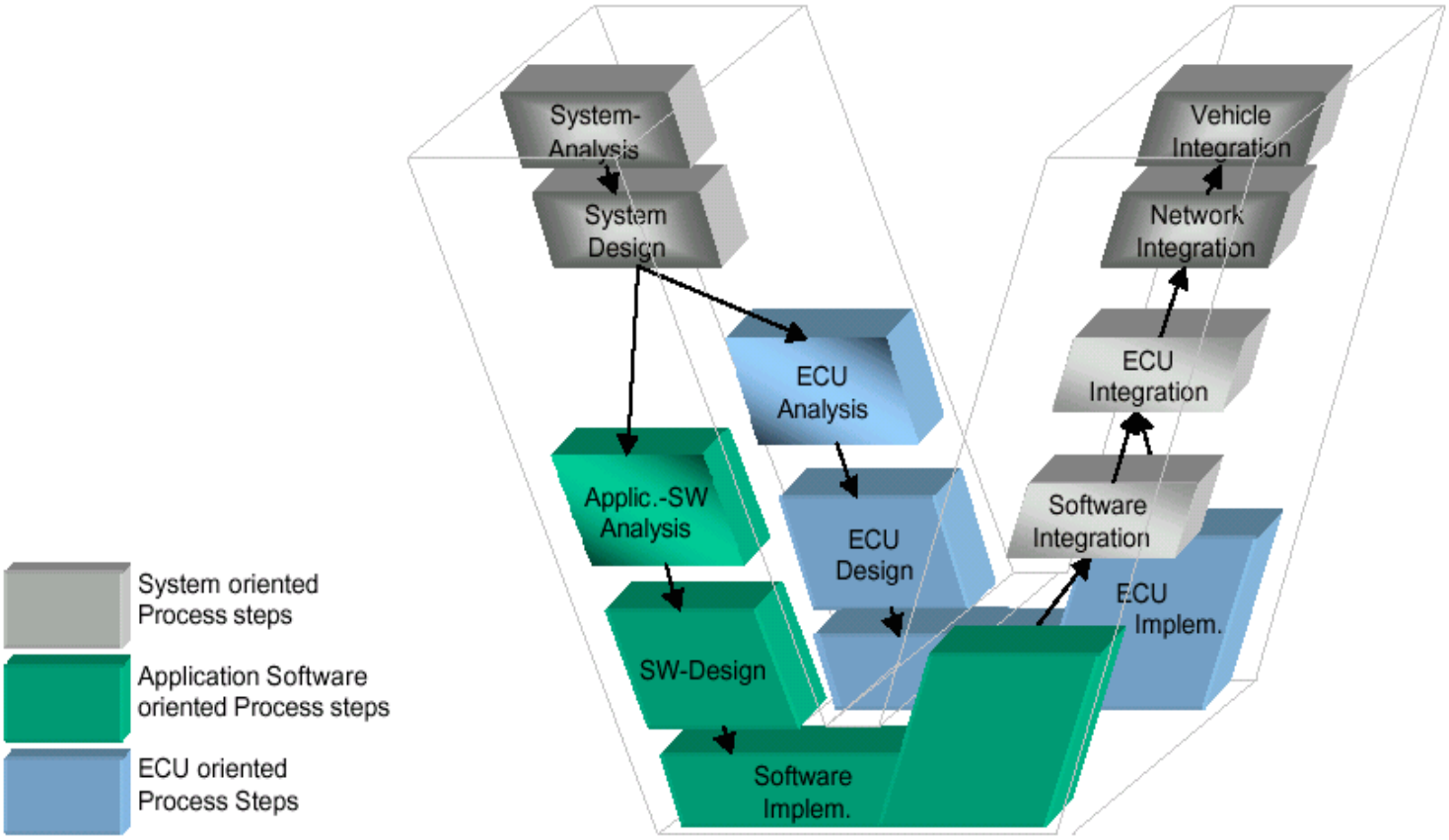
V-Modell(e)

- Es gibt nicht „das“ allgemein gültige V-Modell, sondern verschiedene Ausprägungen
- Charakteristisch für die SW-Entwicklung bei eingebetteten Systemen ist die Kopplung mit der System-Entwicklung und der HW-Entwicklung

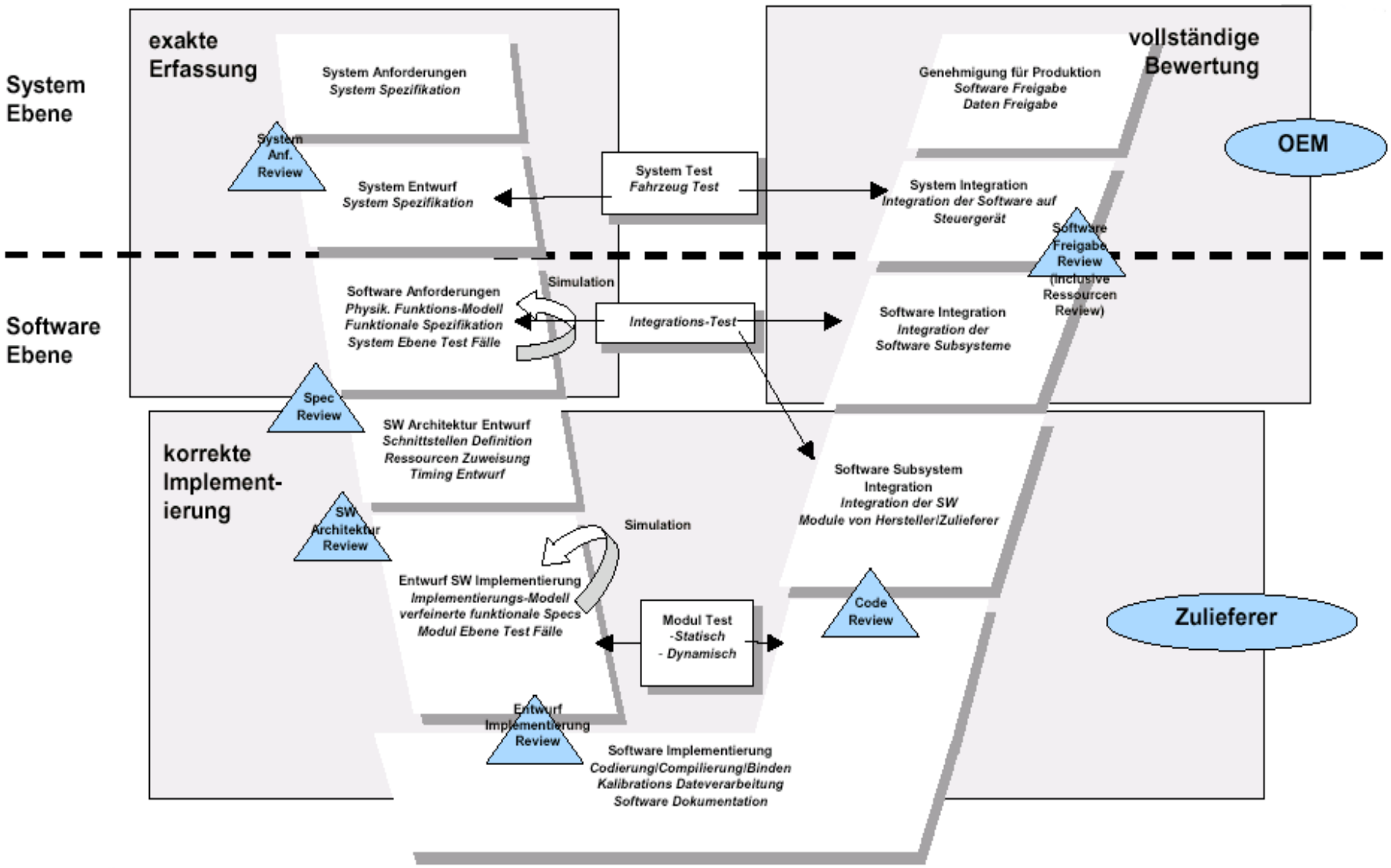


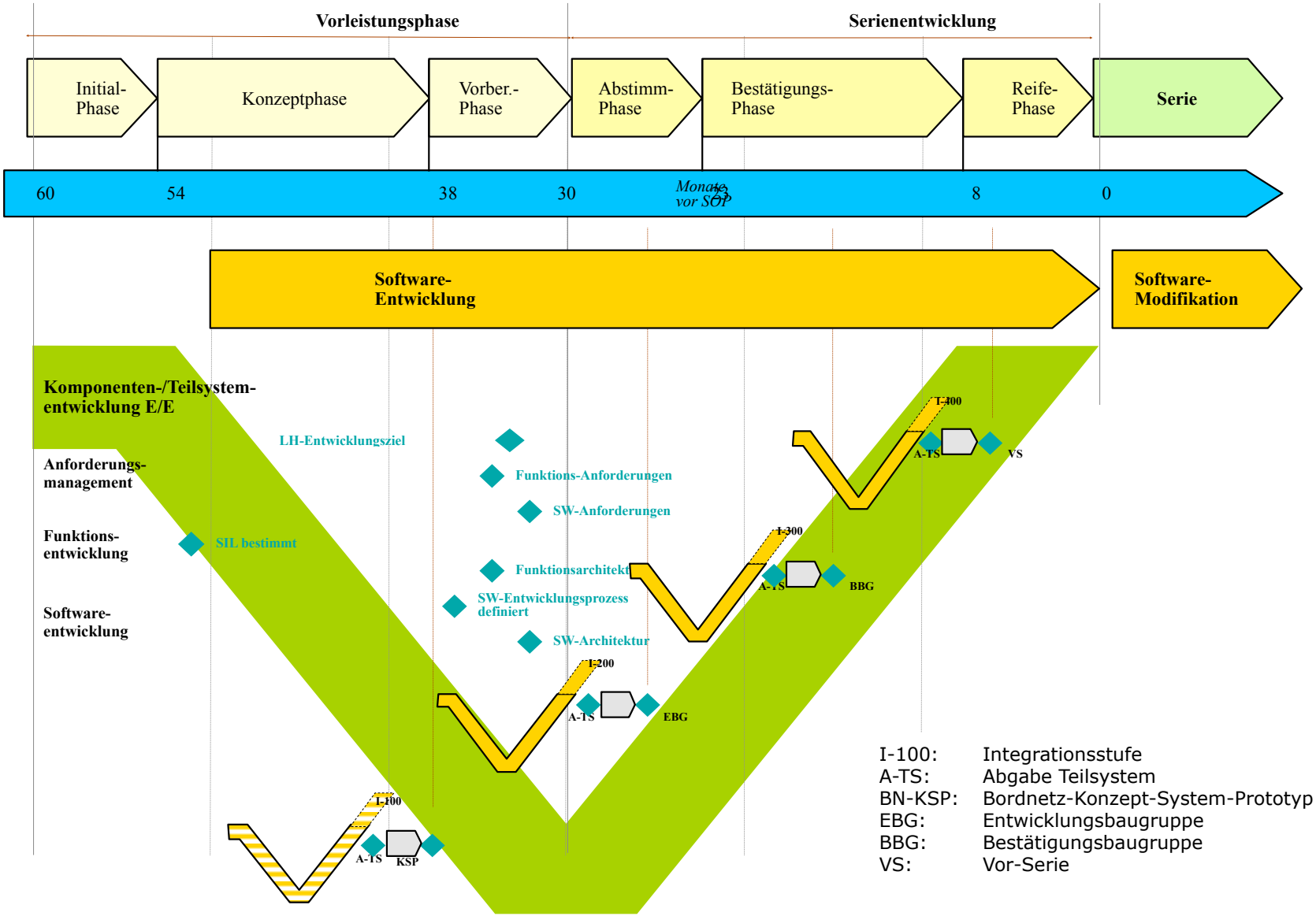


Vereinfachtes V-Modell

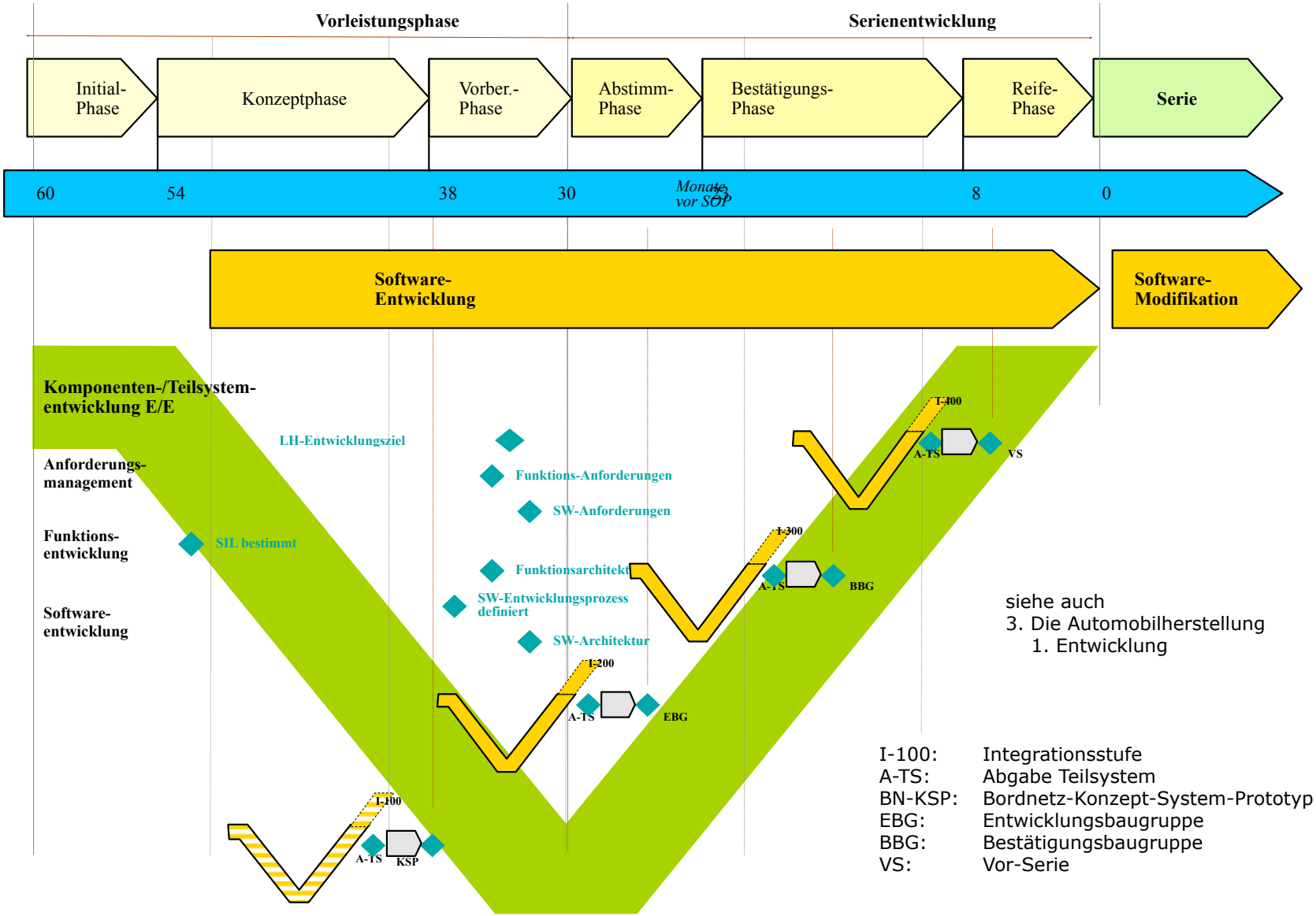


Detaillierung V-Modell





I-100: Integrationsstufe
A-TS: Abgabe Teilsystem
BN-KSP: Bordnetz-Konzept-System-Prototyp
EBG: Entwicklungsbaugruppe
BBG: Bestätigungsbaugruppe
VS: Vor-Serie

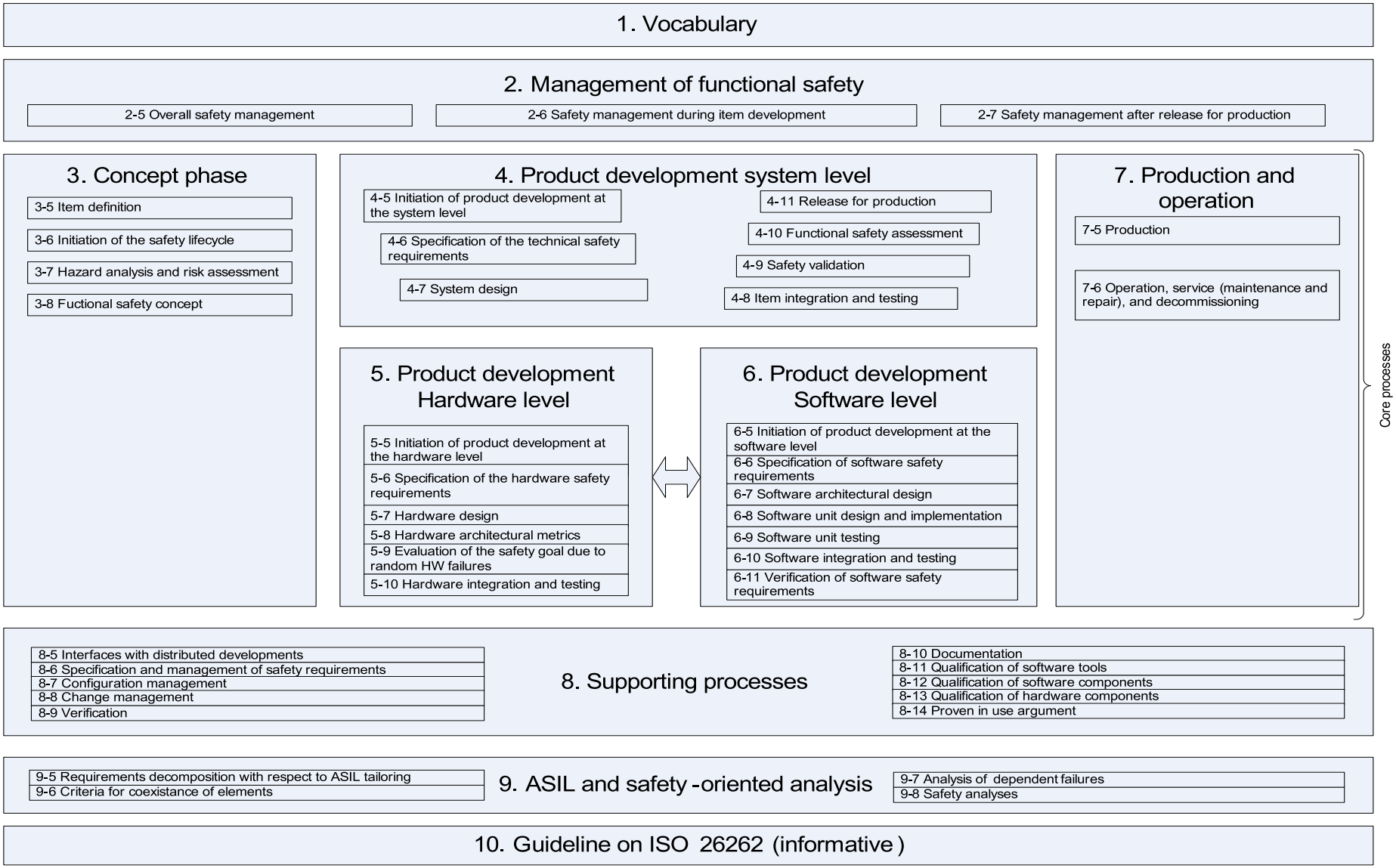


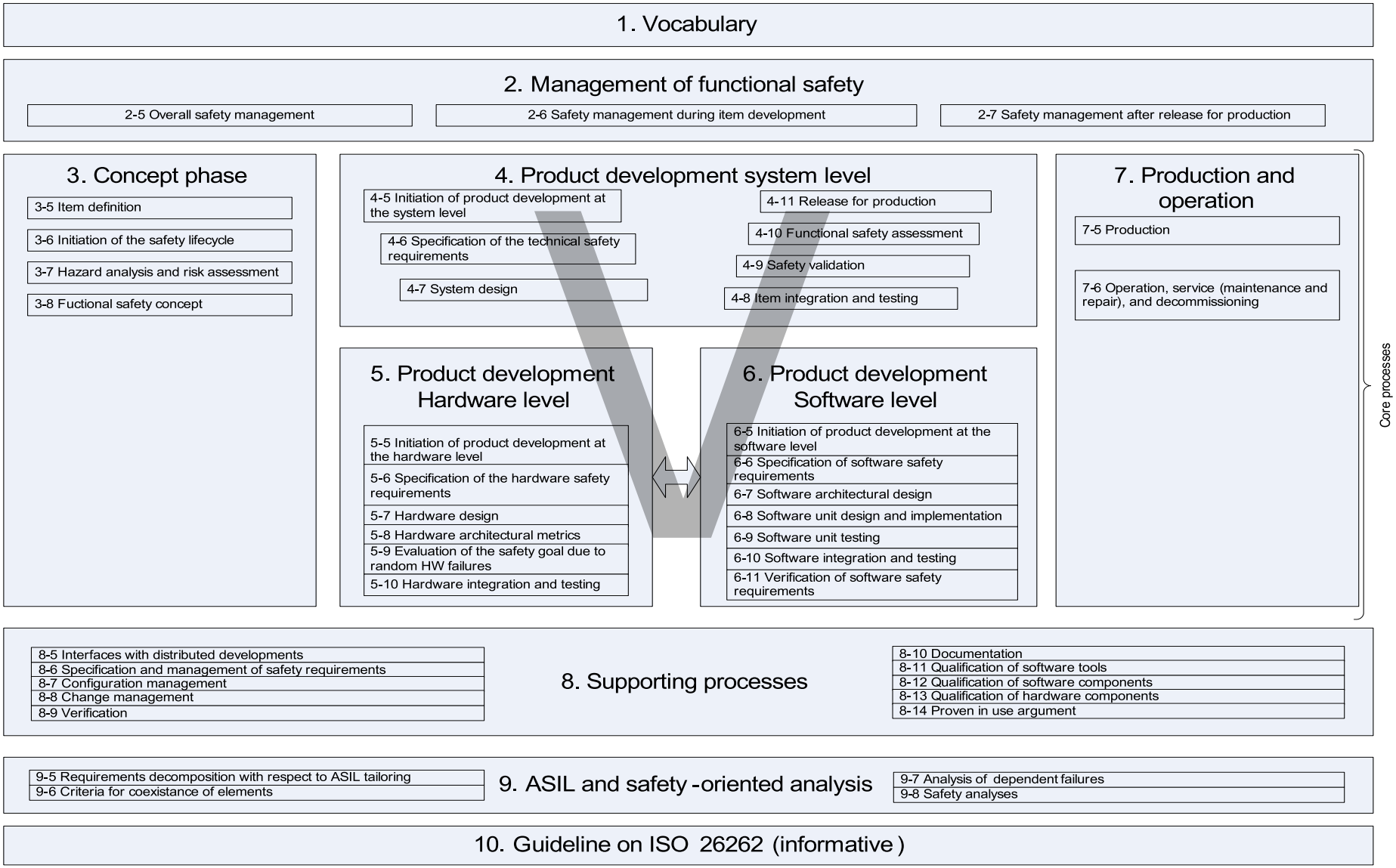
siehe auch
3. Die Automobilherstellung
   1. Entwicklung

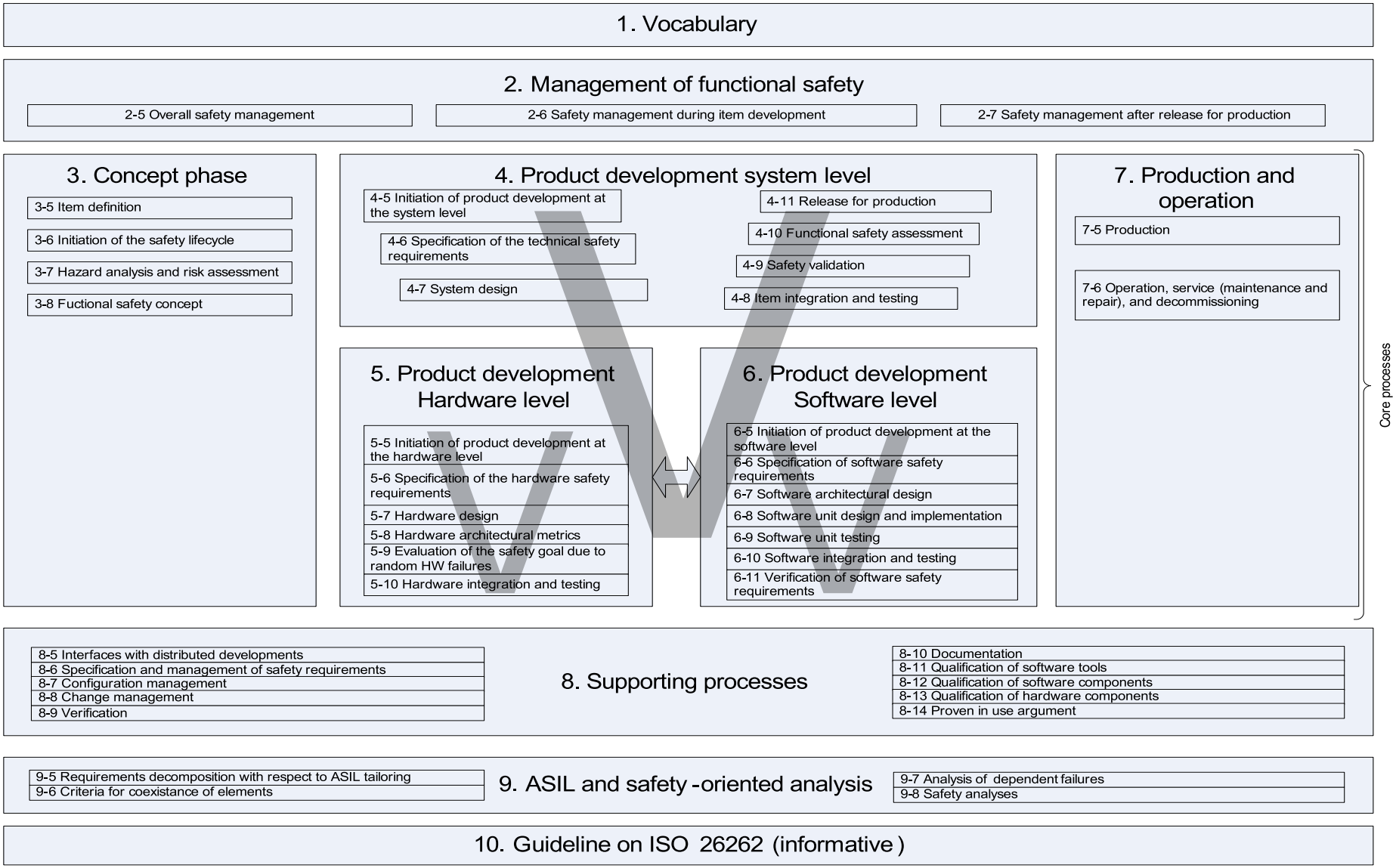
I-100:	Integrationsstufe
A-TS:	Abgabe Teilsystem
BN-KSP:	Bordnetz-Konzept-System-Prototyp
EBG:	Entwicklungsbaugruppe
BBG:	Bestätigungsbaugruppe
VS:	Vor-Serie

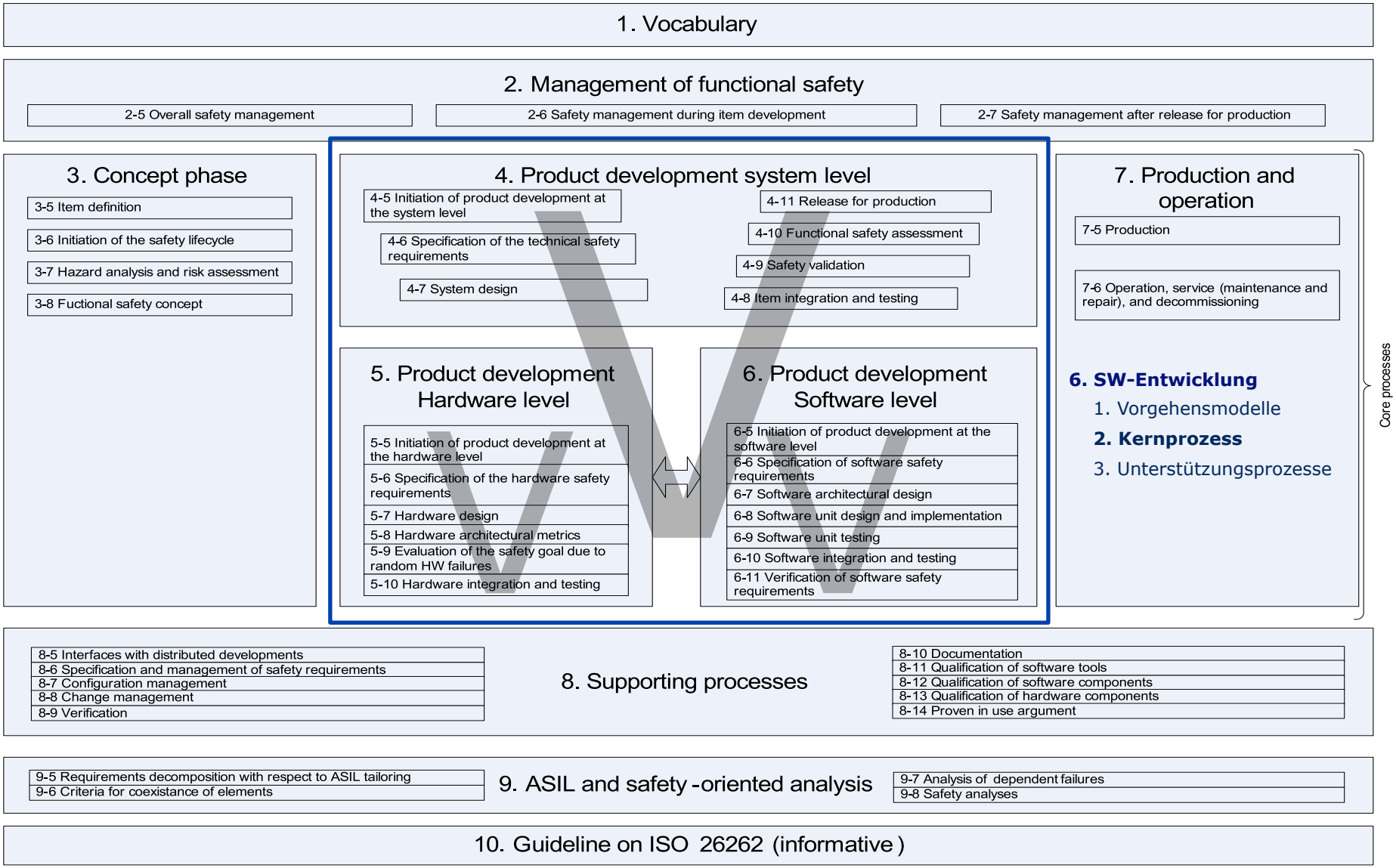
Software Entwicklungsprozess

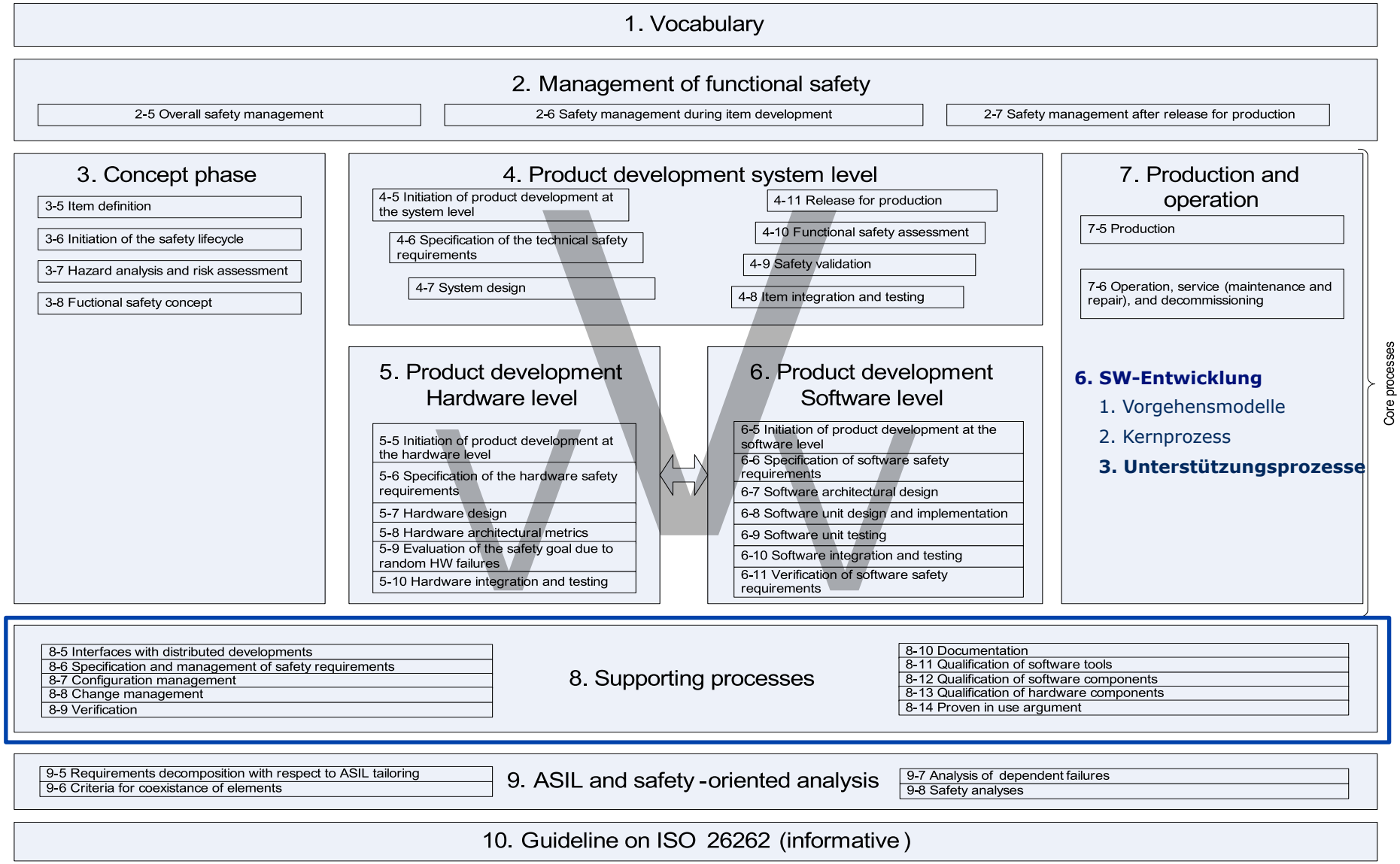
- Einordnung in Produktentstehungsprozess
  - Komponenten-/Teilsystementwicklung E/E
    - LH-Entwicklungsziel Lastenheft
    - Funktions-Anforderungen
    - SW-Anforderungen
  - Funktionsentwicklung
    - (A)SIL bestimmt (Automotive) Safety Integrity Level (Teil 7 Standards)
    - Funktionsarchitektur
  - Softwareentwicklung
    - SW-Entwicklungsprozess definiert
    - SW-Architektur
  - Integrationsstufen
    - I-100 BN-KSP: Bordnetz-Konzept-System-Prototyp
    - I-200 EBG: Entwicklungsbaugruppe
    - I-300 BBG: Bestätigungsbaugruppe
    - I-400 VS: Vorserie

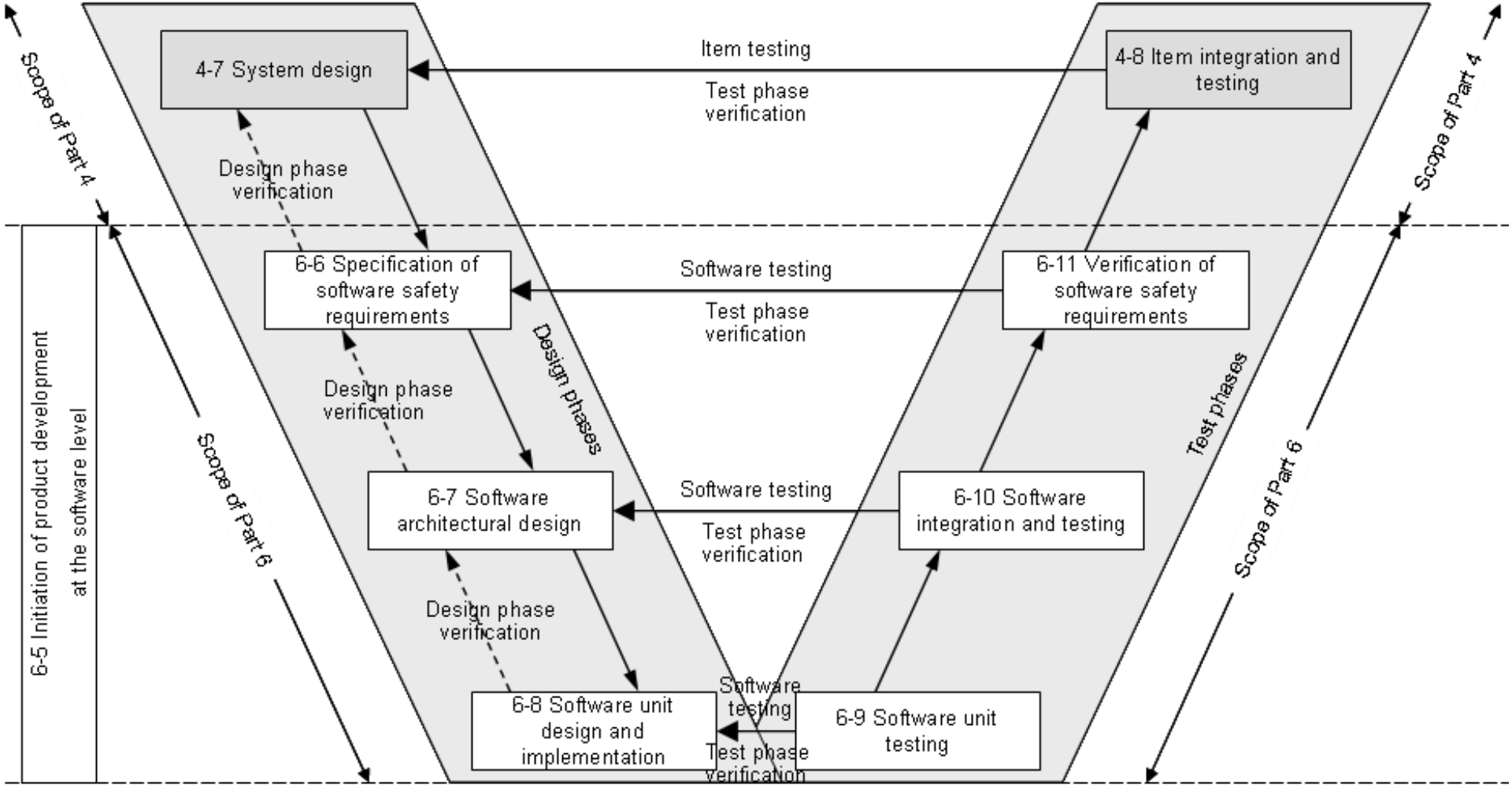

1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Fuctional safety concept
4. Product development system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
Core processes
5. Product development Hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of the hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development Software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
8-5 Interfaces with distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8. Supporting processes
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistance of elements
9. ASIL and safety-oriented analysis
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262 (informative)

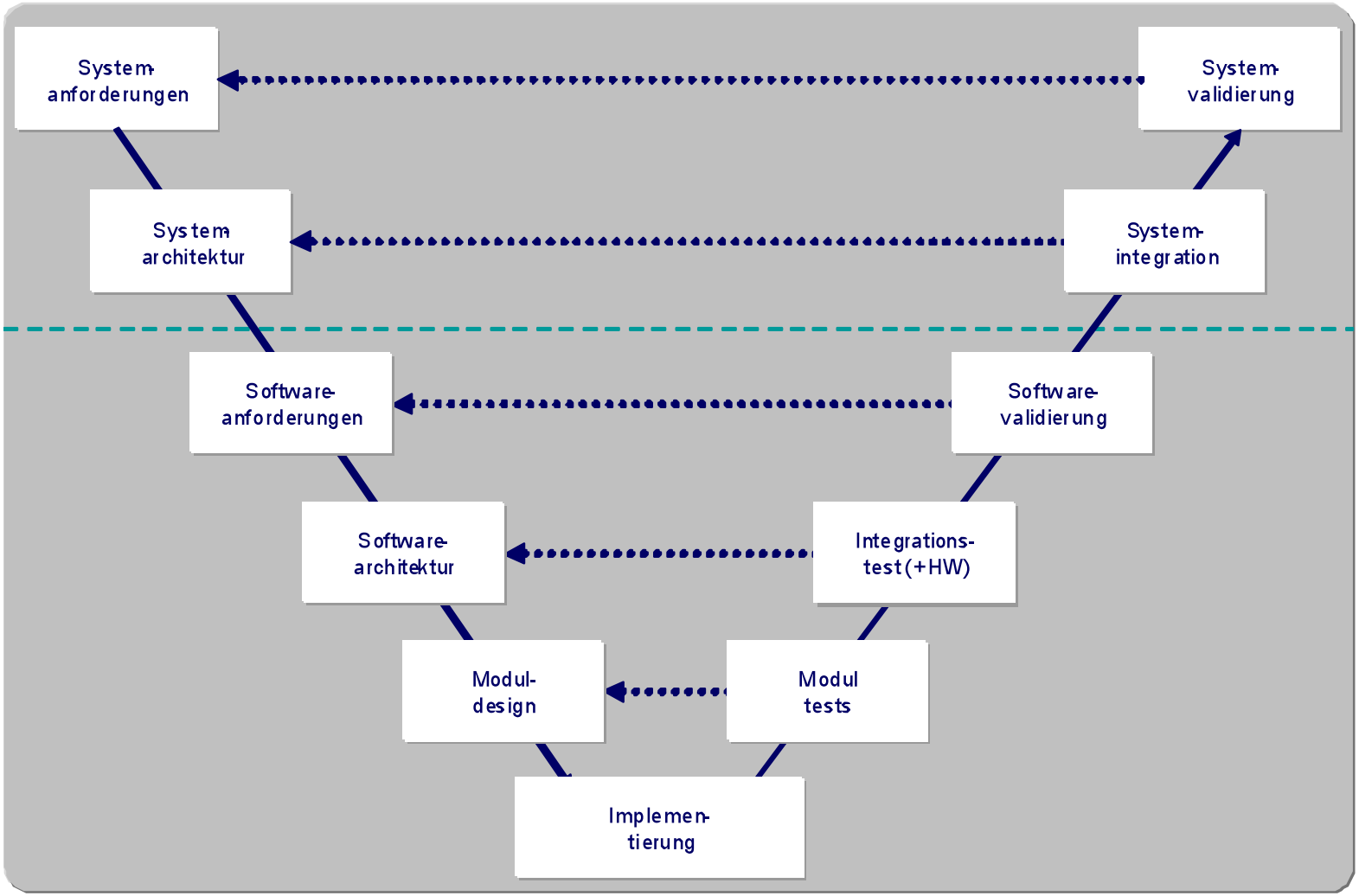

1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Fuctional safety concept
4. Product development system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
5. Product development Hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of the hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development Software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
Core processes
8-5 Interfaces with distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8. Supporting processes
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistance of elements
9. ASIL and safety-oriented analysis
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262 (informative)


1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Fuctional safety concept
4. Product development system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
5. Product development Hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of the hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development Software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
Core processes
8-5 Interfaces with distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8. Supporting processes
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistance of elements
9. ASIL and safety-oriented analysis
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262 (informative)

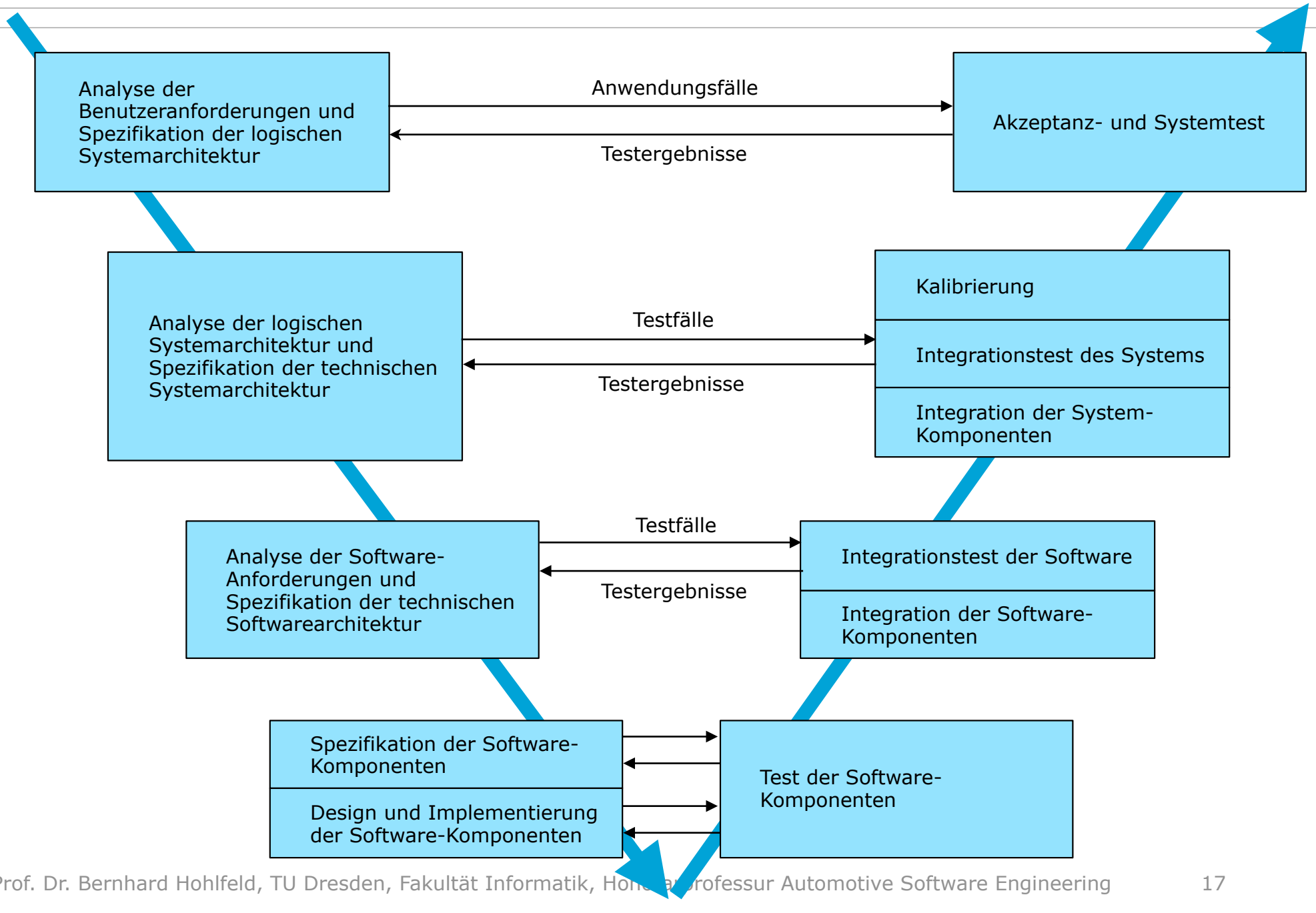

1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Fuctional safety concept
4. Product development system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
5. Product development Hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of the hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development Software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
6. SW-Entwicklung
1. Vorgehensmodelle
2. Kernprozess
3. Unterstützungsprozesse
Core processes
8-5 Interfaces with distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8. Supporting processes
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistance of elements
9. ASIL and safety-oriented analysis
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262 (informative)


1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Fuctional safety concept
4. Product development system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
5. Product development Hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of the hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development Software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
6. SW-Entwicklung
1. Vorgehensmodelle
2. Kernprozess
3. Unterstützungsprozesse
Core processes
8-5 Interfaces with distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8. Supporting processes
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistance of elements
9. ASIL and safety-oriented analysis
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262 (informative)

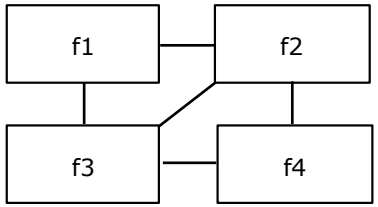

Scope of Part 4
4-7 System design
Item testing
Test phase verification
4-8 Item integration and testing
Scope of Part 4
Design phase verification
6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phases
Test phases
Design phase verification
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Scope of Part 6
Scope of Part 6
6-5 Initiation of product development at the software level
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing



- DIN EN 50128 Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme …


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

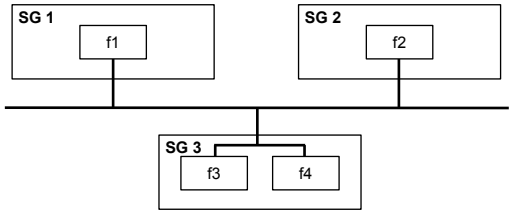
Analyse der Benutzeranforderungen und Spezifikation der logischen Systemarchitektur



Logische Systemarchitektur

Akzeptanz- und Systemtest

Analyse der logischen Systemarchitektur und Spezifikation der technischen Systemarchitektur



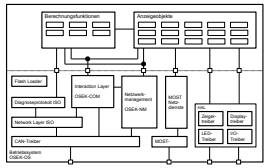
Technische Systemarchitektur

Kalibrierung

Integrationstest des Systems

Integration der System-Komponenten

Analyse der Software-Anforderungen und Spezifikation der technischen Softwarearchitektur



Integrationstest der Software

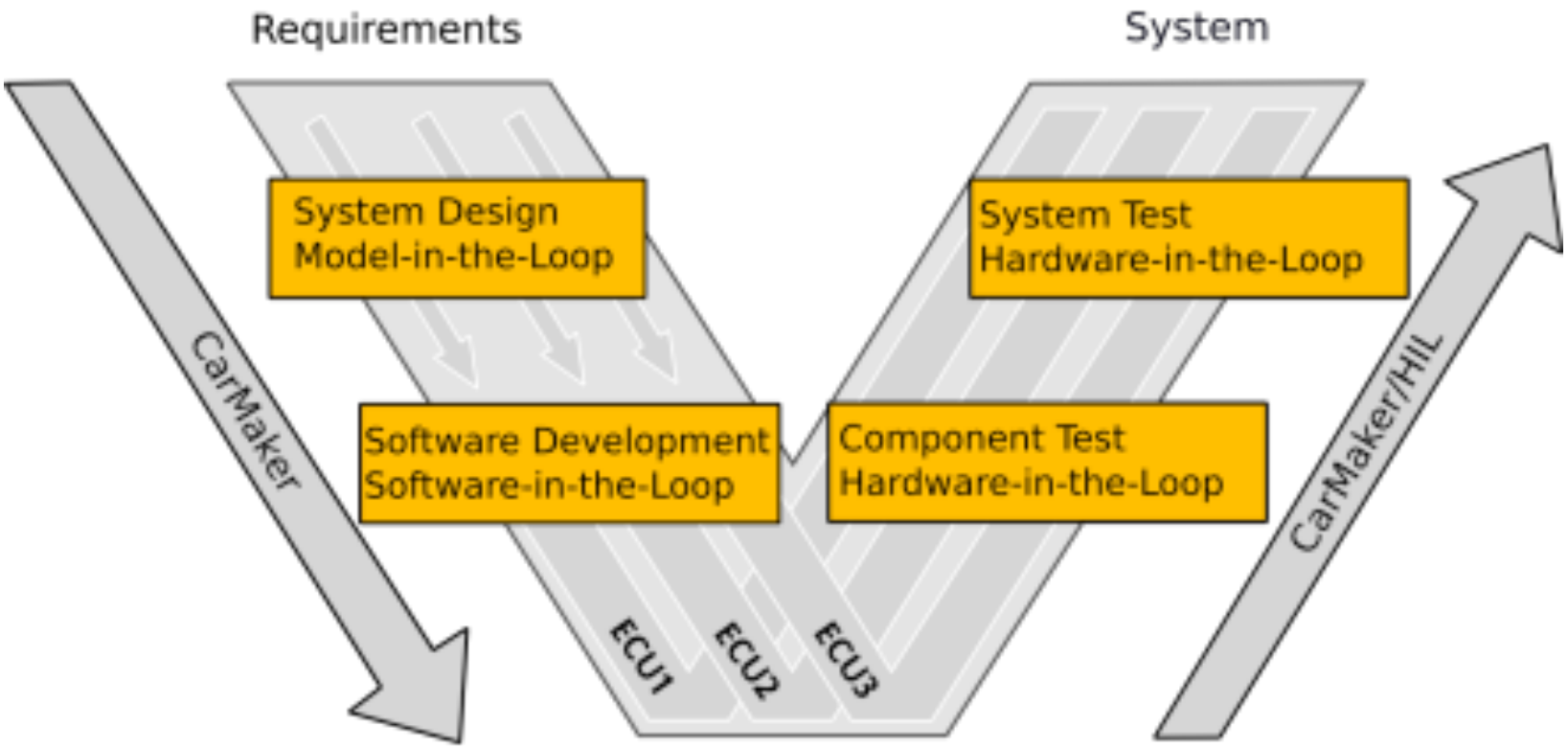
Integration der Software-Komponenten

Spezifikation der Software-Komponenten

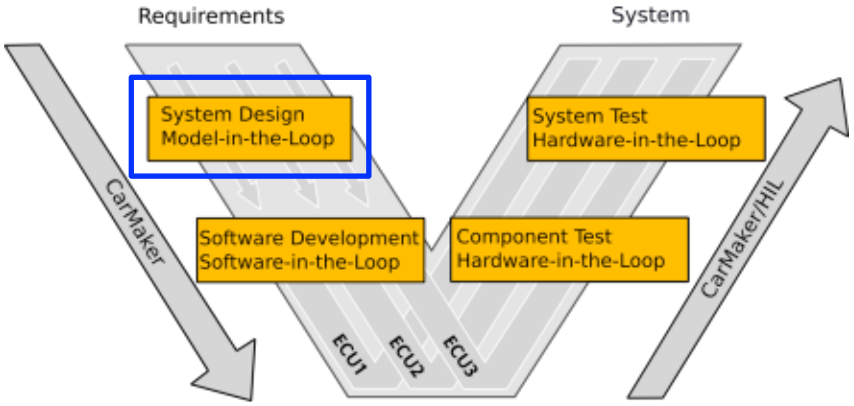
Design und Implementierung der Software-Komponenten

Test der Software-Komponenten

www.ipg.de


Requirements
System
System Design
Model-in-the-Loop
System Test
Hardware-in-the-Loop
Software Development
Software-in-the-Loop
Component Test
Hardware-in-the-Loop
CarMaker
CarMaker/HiL
ECU1
ECU2
ECU3

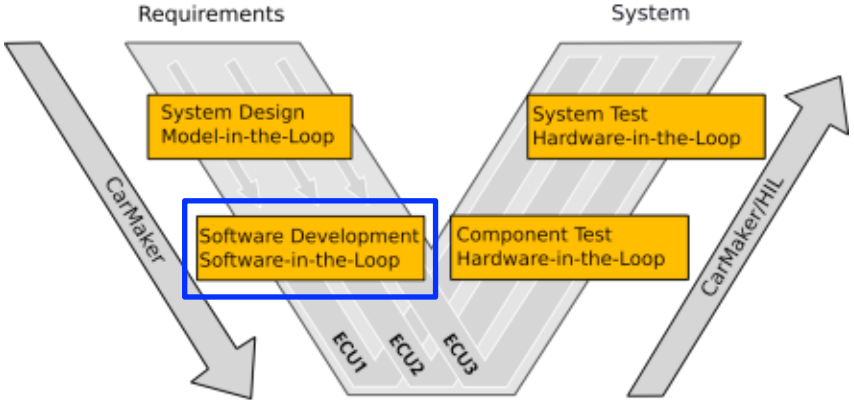
System Design (Quelle www.ipg.de)



- Ausgangspunkt der Entwicklung sind die Anforderungen, die in Form einer Systemspezifikation festgehalten werden.
- Die Funktionen der Systemspezifikation werden in Software-Modellen mit grafisch orientierten Programmier-systemen wie MATLAB/Simulink entwickelt. Diese Phase kann durch die Model-in-the-Loop (MIL)-Simulation unterstützt werden. Dabei werden die Software-Modelle mit CarMaker im virtuellen Fahrzeug validiert.
- Alle Modelle von CarMaker sind vollständig in Simulink integriert, so dass der Anwender problemlos Modifikationen an den Modellen vornehmen kann, indem er z.B. bestimmte Komponenten der Modelle durch eigene Simulink-Blöcke ersetzt.

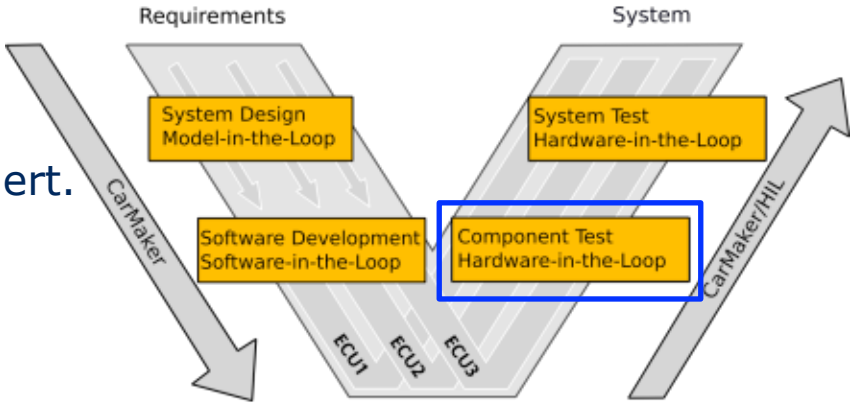
Software Entwicklung (Quelle www.ipg.de)

- Das in der MIL-Simulation getestete Software-Modell wird durch die Software des Reglers ersetzt und in die Simulations-umgebung von CarMaker eingebunden (Software-in-the-Loop SIL). Somit wird der spätere Seriencode in einer virtuellen Fahrzeugumgebung ausgeführt, um das Verhalten zu untersuchen und Fehler in der Implementierung aufzudecken. Die Besonderheit von CarMaker ist, dass auch die SIL-Tests auf einem Echtzeitsystem erfolgen können, um die Software unter zeitlich kritischen Randbedingung zu prüfen. Damit muss die Software nicht wie in vielen anderen Anwendungen für den Test extra angepasst werden.



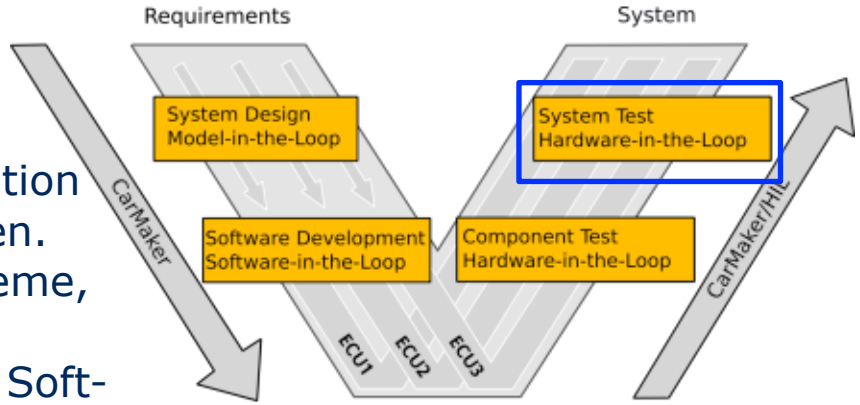
Component Test (Quelle www.ipg.de)

- Die Komponenten-Tests sollen sicherstellen, dass das ausführbare Programm im Zusammenspiel mit der Steuergeräte-Hardware und dem Betriebssystem funktioniert. Das reale Steuergerät wird hierzu an einen CarMaker/HIL-Prüfstand angebunden, der in Echtzeit das Fahrzeug simuliert und die Signale zur Ansteuerung des Steuergerätes generiert (Hardware-in-the-Loop).



- Die Bandbreite der Tests am HIL-Prüfstand reicht von Performance-Tests bis hin zu Sicherheitstest, bei denen das Verhalten des Fahrzeugs in Folge von elektrischen Fehlern geprüft wird. Für diese Tests bietet IPG zusätzlich den FailSafeTester an, mit dem elektrische Fehler am HIL-Prüfstand automatisch generiert werden können. Dank der Unterstützung von CCP (CAN Calibration Protocol) und XCP (Uiversal Measurement and Calibration Protocol) können mit CarMaker auch Steuergeräte-interne Messgrößen erfasst und analysiert werden.

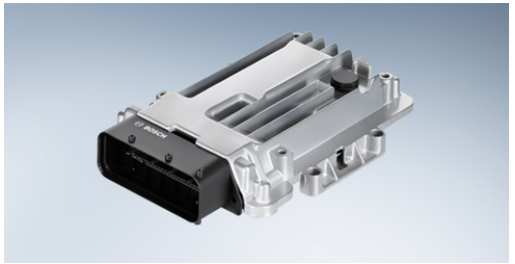
System Test (Quelle www.ipg.de)

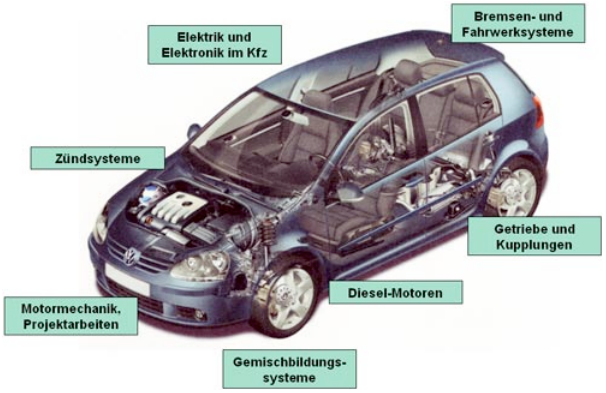


- Der Automobilhersteller hat die Verantwortung für das Fahrzeug als Gesamtsystem. Er muss sicherstellen, dass die einzelnen Steuergeräte nach der Integration im Fahrzeug optimal miteinander interagieren. Um dies zu gewährleisten, müssen die Systeme, die häufig von unterschiedlichen Zulieferern stammen, im Verbund getestet werden. Die Soft- und Hardware der CarMaker/HIL-Systeme ist daher modular aufgebaut, so dass einzelne Steuergeräte flexibel zu- oder abgeschaltet werden können.
- Die Funktionalität des Gesamtverbundes kann damit in allen Fahrsituationen (z.B. Kurvenfahrten, Vollbremsung etc.) untersucht werden. Eine wichtige Rolle spielt bei den Verbundtests die Überwachung der Kommunikation zwischen den Steuergeräten (via CAN, FlexRay etc.). Neben dem Normalbetrieb muss auch hier das Verhalten im Fehlerfall geprüft werden. Je nach Ziel der Untersuchung kann der Prüfling neben den Steuergeräten auch mechanische Komponenten (z.B. Dämpfer, Lenksystem, Motor) enthalten. Die Anzahl der realen Komponenten, die in den Prüfstand integriert werden, kann je nach Anwendung stark variieren.

Begriffsdefinitionen

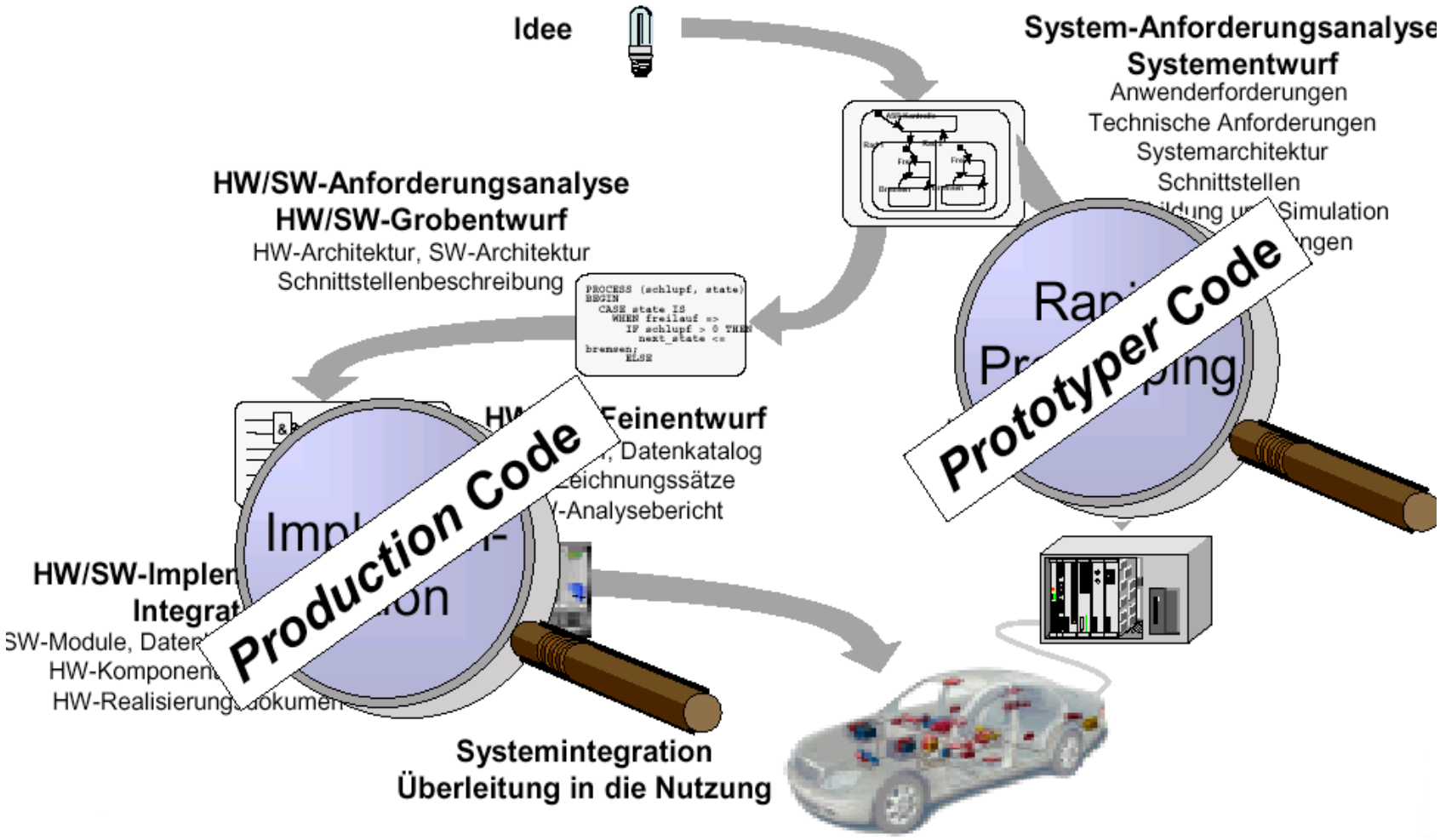
- MIL Model in the Loop, SIL Software in the Loop
  - Modell / SG-Software läuft auf simuliertem SG, das von simulierter Fahrzeugumgebung (rechnergenerierten Sensor- und Bussignalen) gespeist wird.
- HIL Hardware in the Loop
  - Reale SG-Hardware wird von simulierter Fahrzeugumgebung gespeist
- Prototyp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



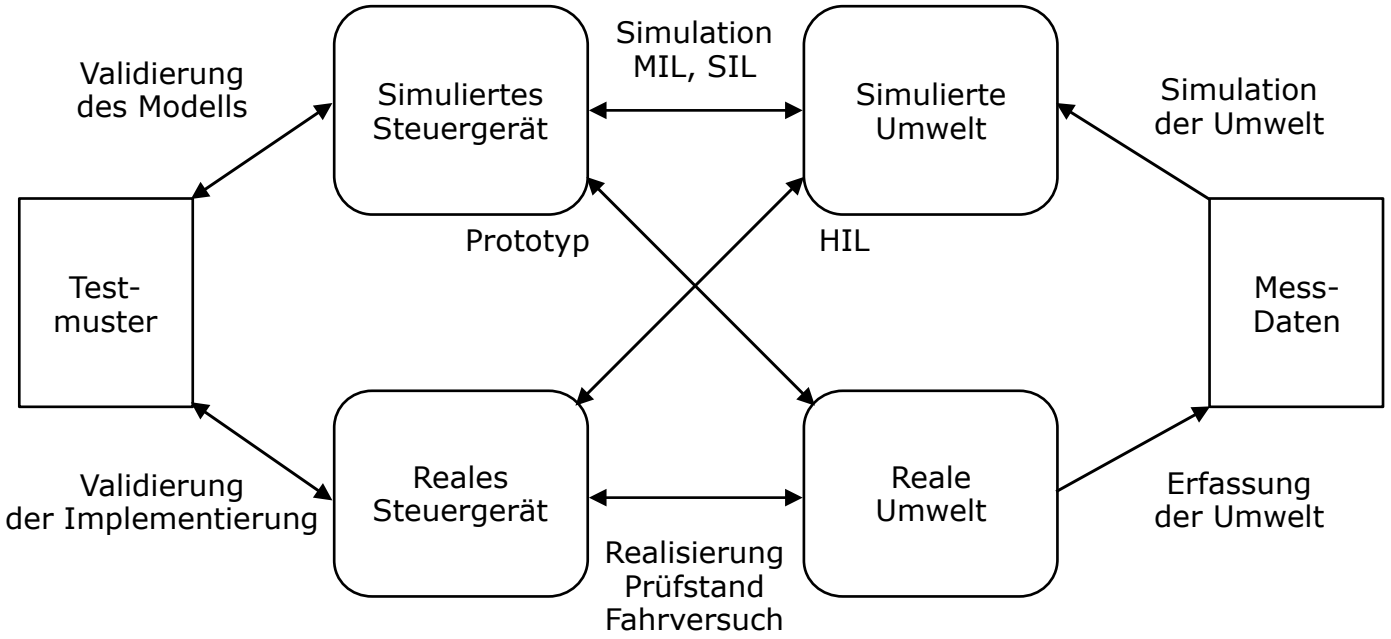


	Steuergerät	
Umgebung, Fahrzeug	simuliert	real
simuliert	MIL, SIL	HIL
real	Prototyp	Prüfstand, Fahrversuch

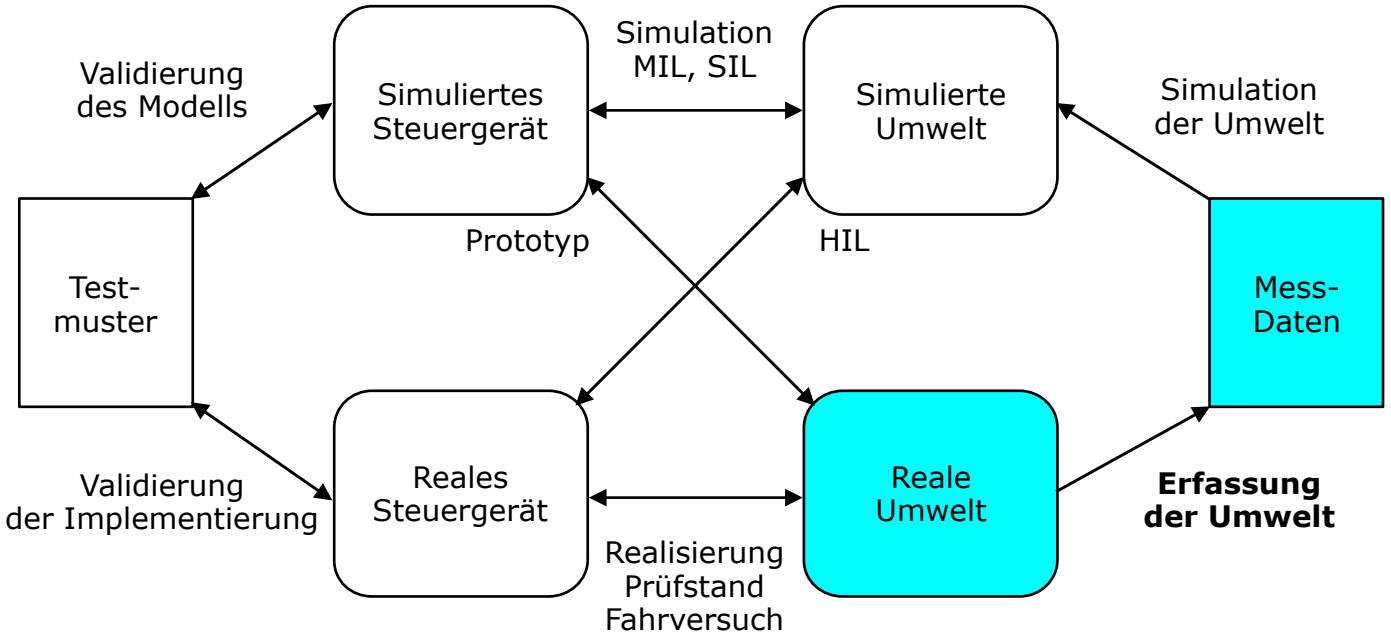
Prototyp und Produkt



Simulation, Rapid Prototyping und Test

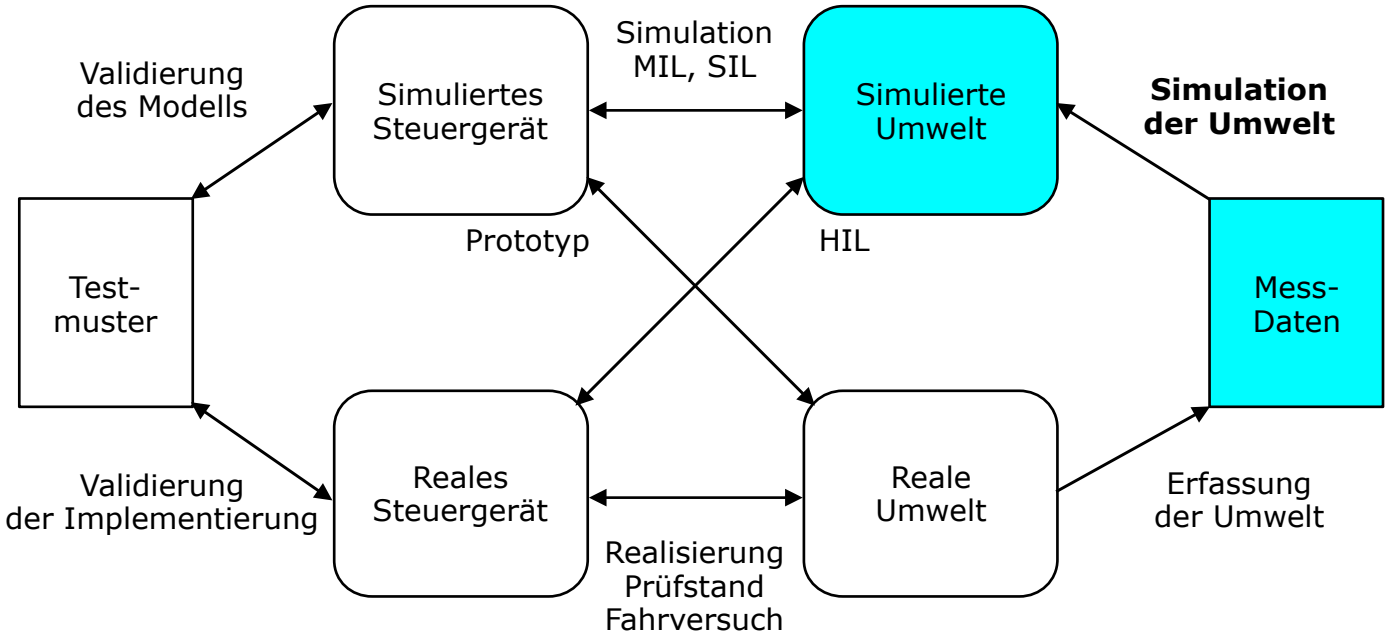


Simulation, Rapid Prototyping und Test



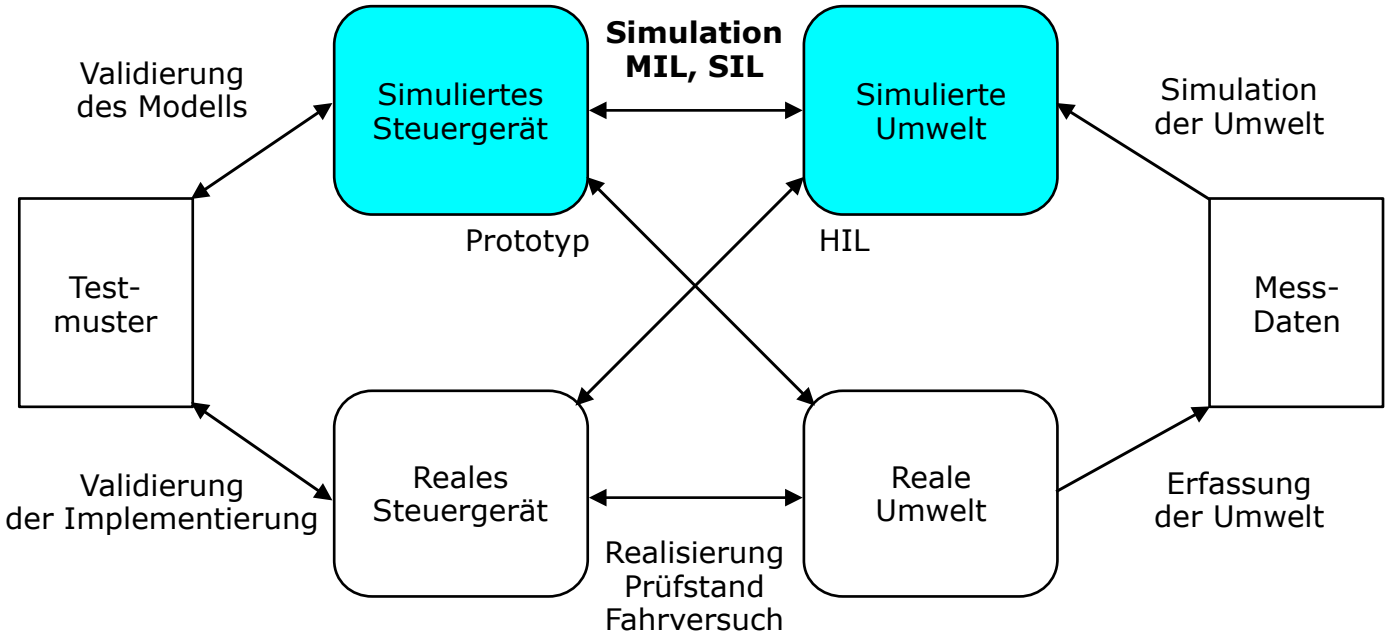
- Erfassung von Messdaten zur Simulation der Umwelt

Simulation, Rapid Prototyping und Test



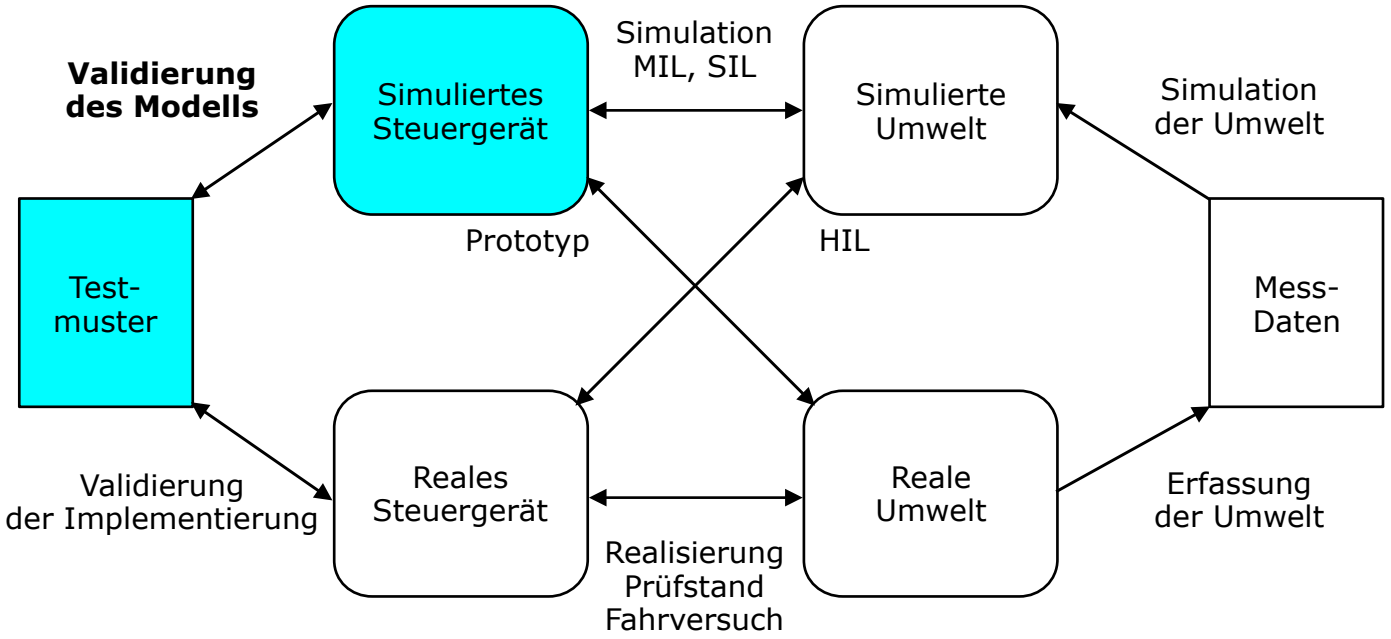
- Simulation der Umwelt

Simulation, Rapid Prototyping und Test



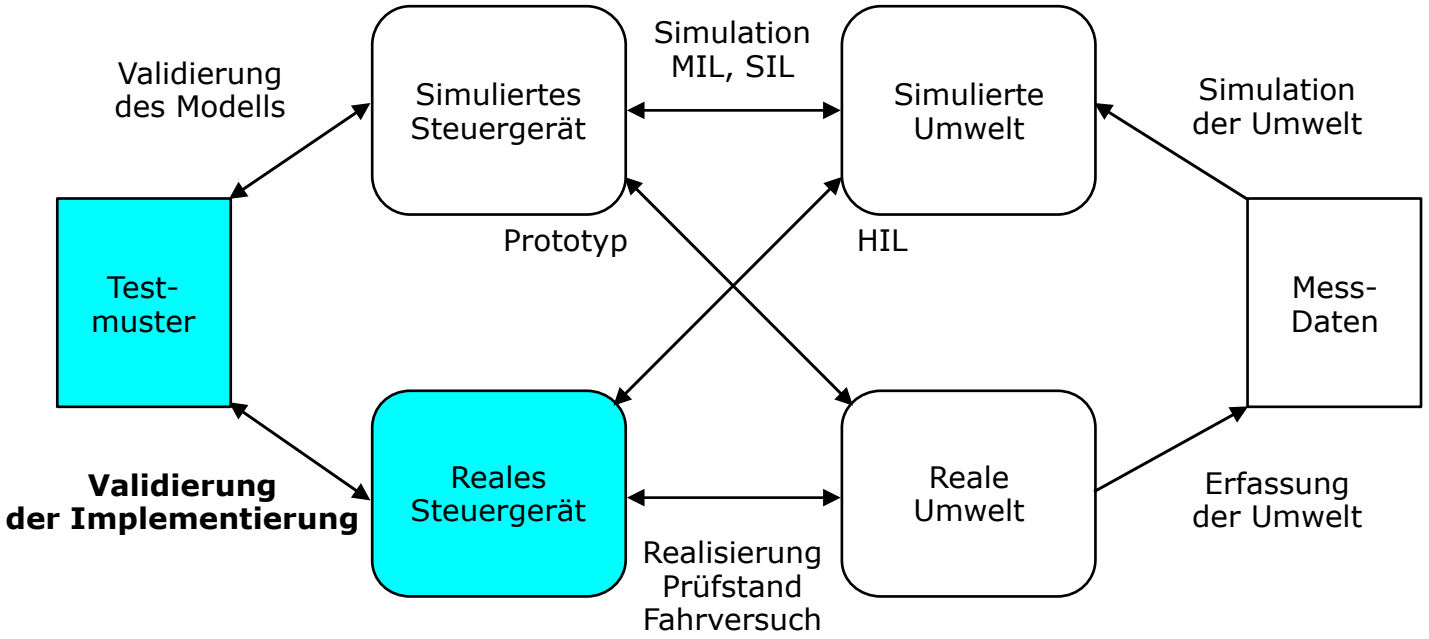
- Simulation von Steuergerät und Umwelt:
  Model in the Loop (MIL), Software in the Loop (SIL)

Simulation, Rapid Prototyping und Test



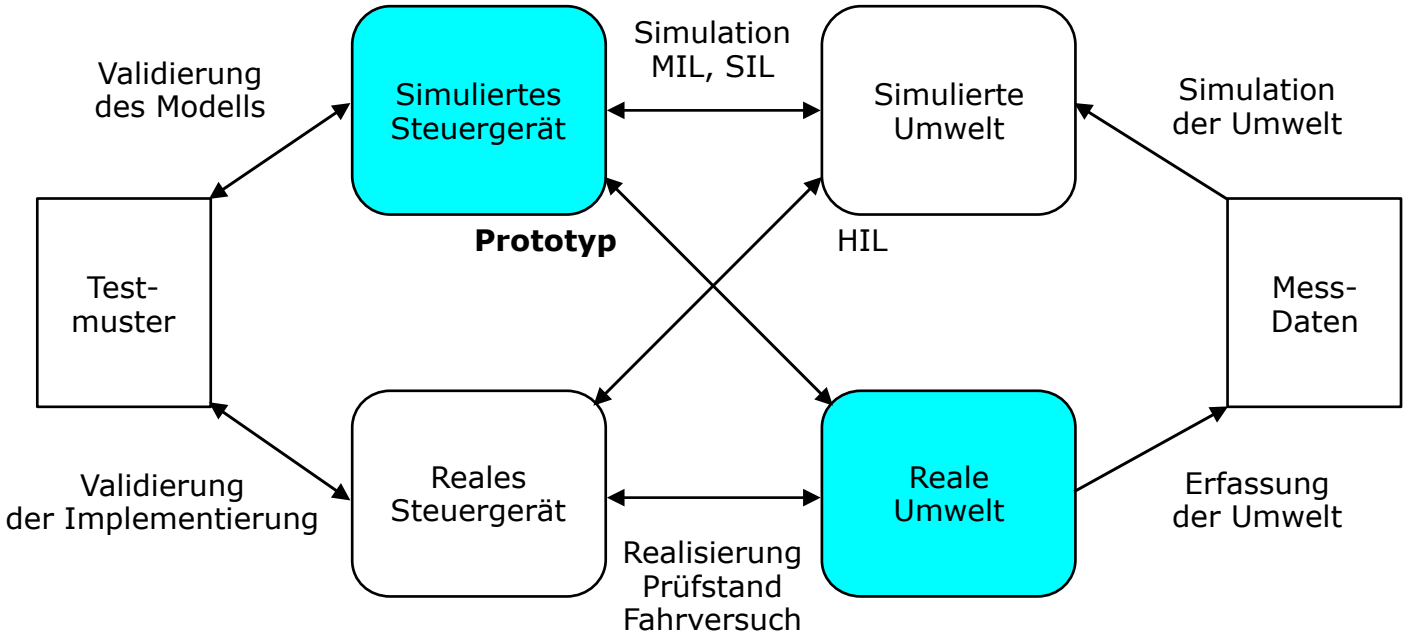
- Validierung des Modells: Test des Simulierten Steuergerätes

Simulation, Rapid Prototyping und Test



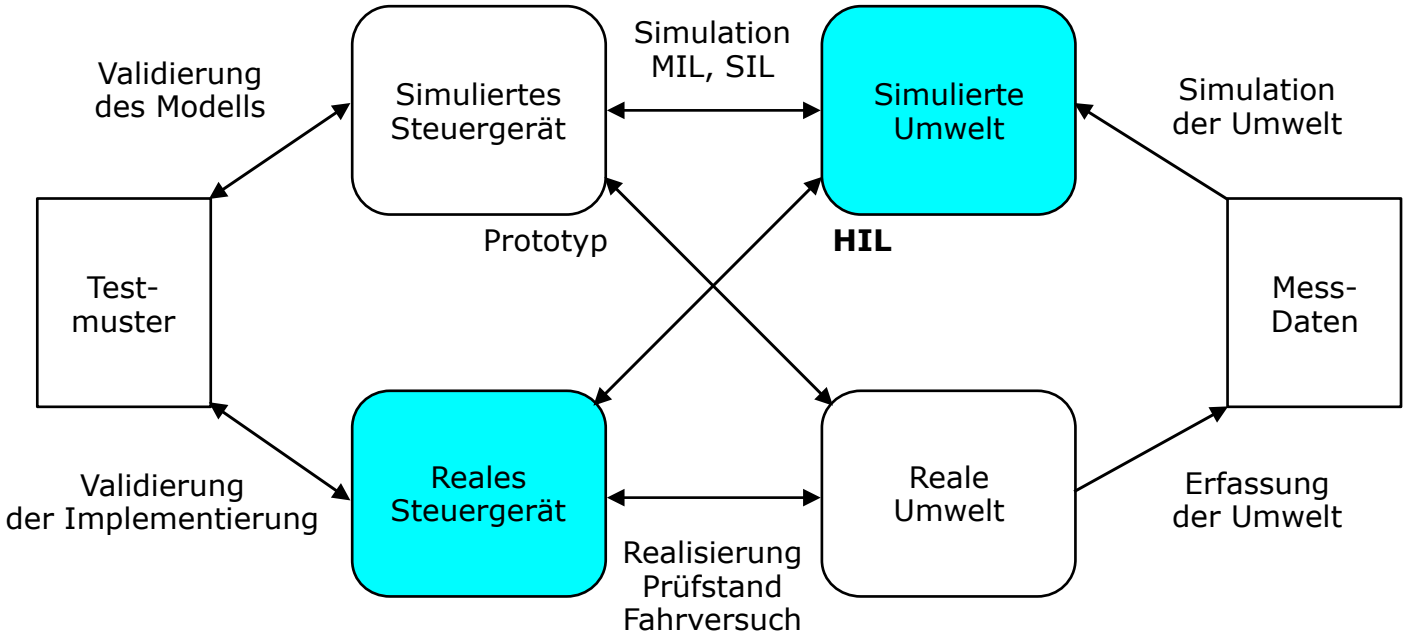
- Validierung der Implementierung: Test des Realen Steuergerätes
- Gleiche Testdaten wie bei Validierung des Modells

Simulation, Rapid Prototyping und Test



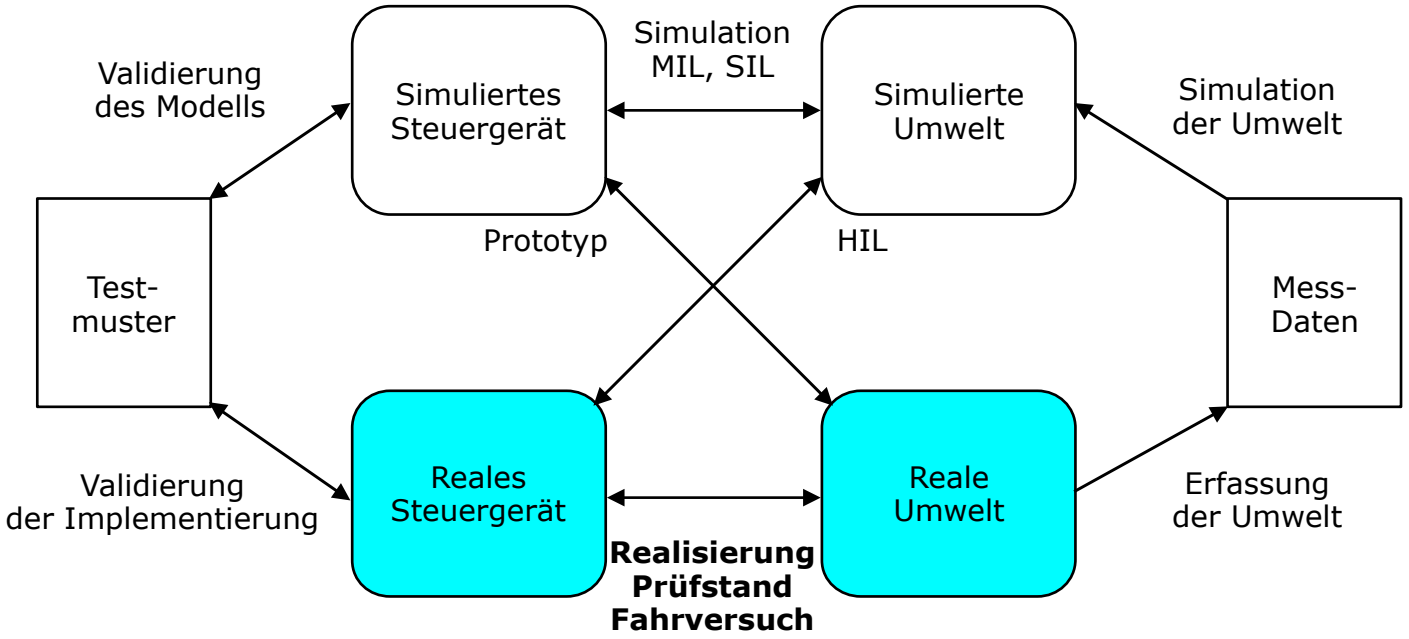
- Prototyp / Rapid Prototyping: Frühe Tests in realer Umgebung

Simulation, Rapid Prototyping und Test



- Umfangreiche Tests des Steuergerätes in simulierter Umgebung: Hardware in the Loop (HIL)

Simulation, Rapid Prototyping und Test



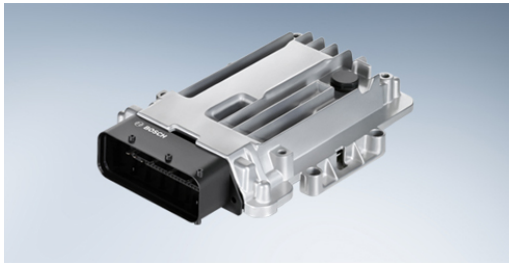
- Realisierung: Prüfstand undFahrversuch

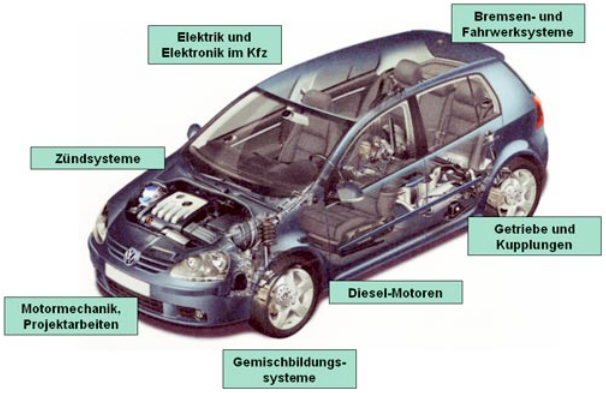
Testvarianten im Integrationsprozess

- MIL Model in the Loop
- SIL Software in the Loop
- PIL Prozessor in the Loop
- HIL Hardware in the Loop
  - K-HIL Komponente
  - S-HIL System
- VIL Vehicle in the Loop
- EFP Erweiterte Funktionsprüfung

Begriffsdefinitionen

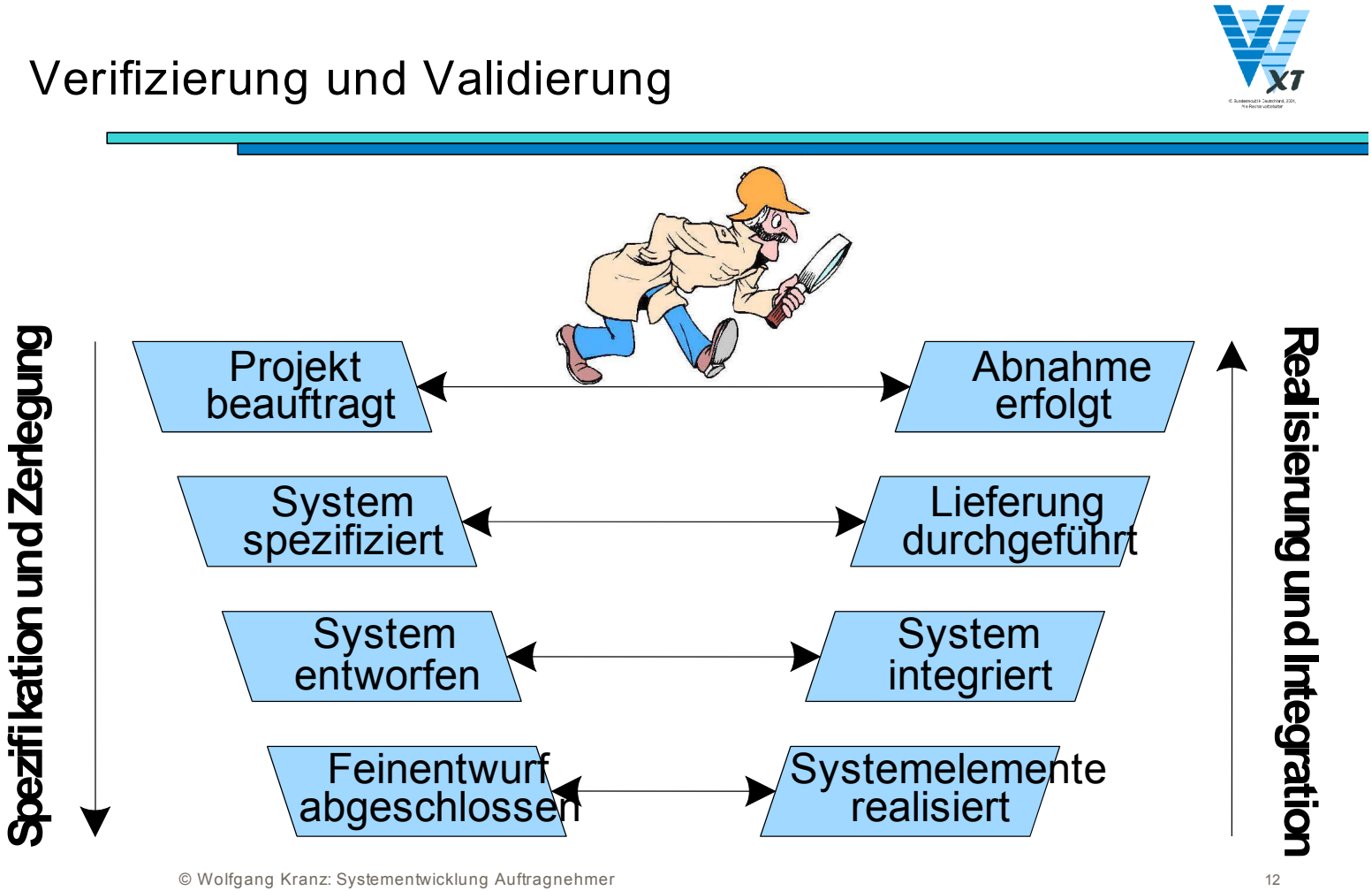
- MIL Model in the Loop, SIL Software in the Loop
  - Modell / SG-Software läuft auf simuliertem SG, das von simulierter Fahrzeugumgebung (rechnergenerierten Sensor- und Bussignalen) gespeist wird.
- HIL Hardware in the Loop
  - Reale SG-Hardware wird von simulierter Fahrzeugumgebung gespeist
- Prototyp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



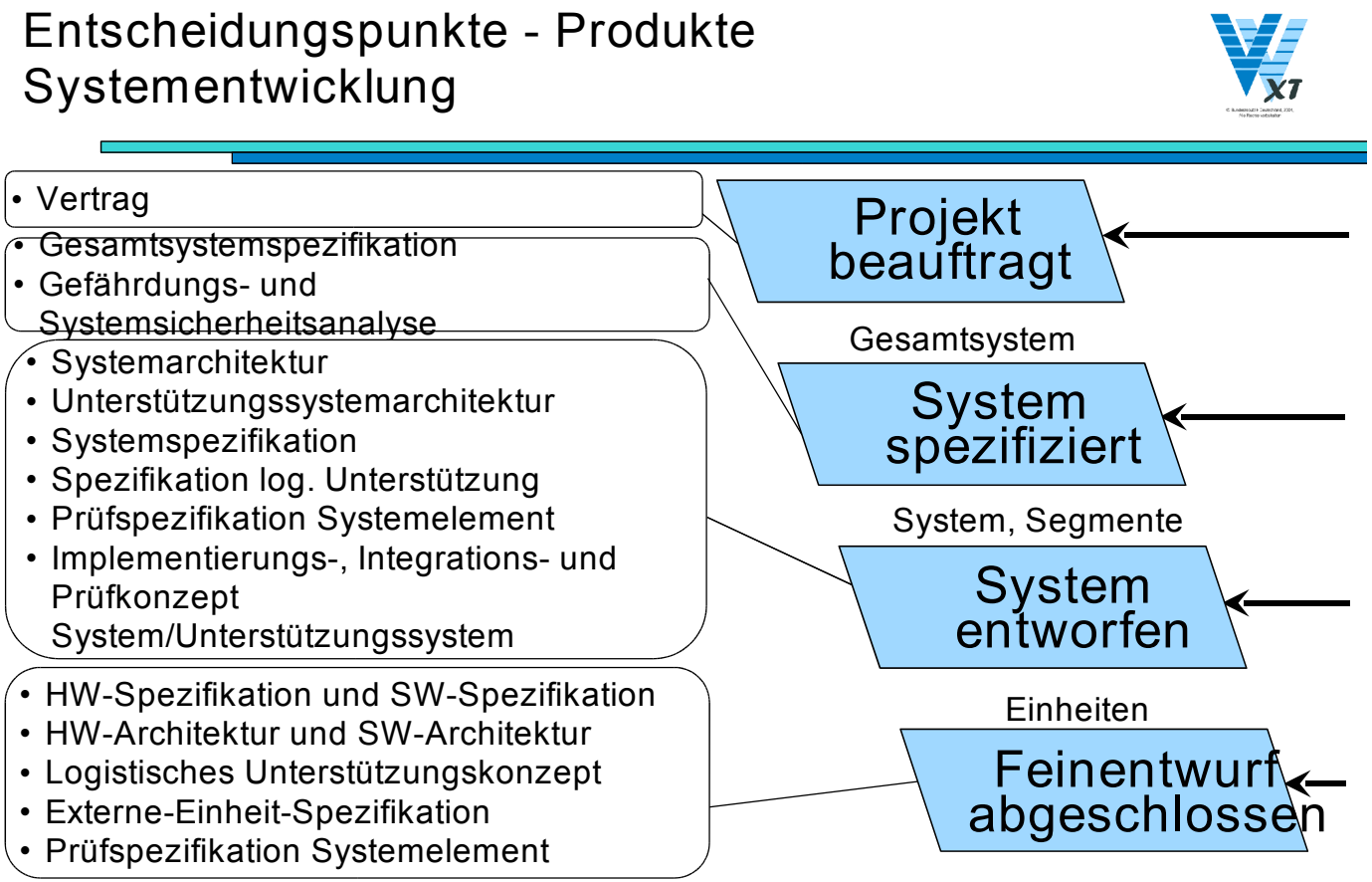


	Steuergerät	
Umgebung, Fahrzeug	simuliert	real
simuliert	MIL, SIL	PIL, HIL, VIL
real	Prototyp	Prüfstand, Fahrversuch

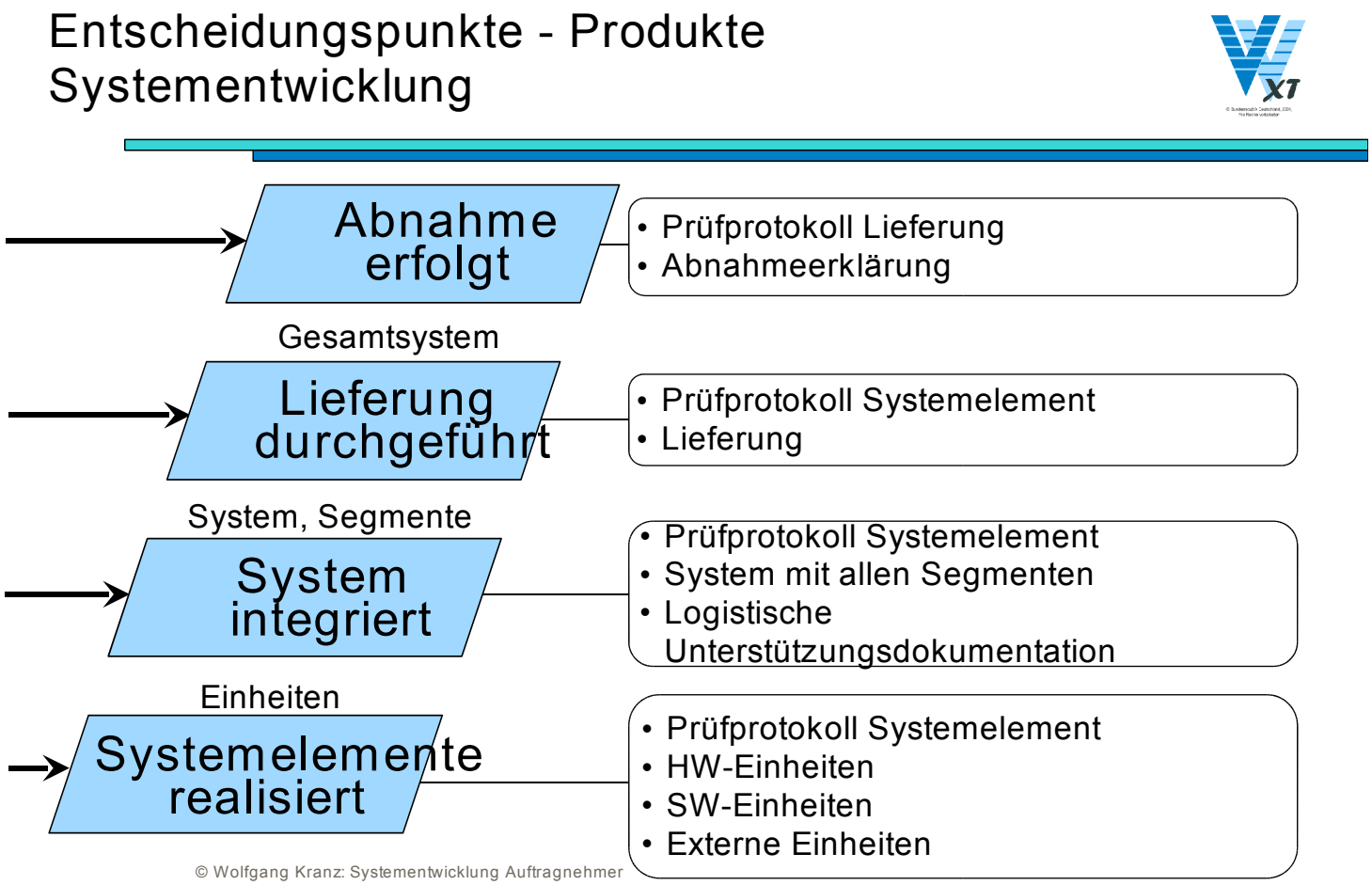
V-Modell XT (1)


Verifizierung und Validierung
Projekt beauftragt
Abnahme erfolgt
System spezifiziert
Lieferung durchgeführt
System entworfen
System integriert
Feinentwurf abgeschlossen
Systemelemente realisiert
Spezifikation und Zerlegung
Realisierung und Integration
© Wolfgang Kranz: Systementwicklung Auftragnehmer
12

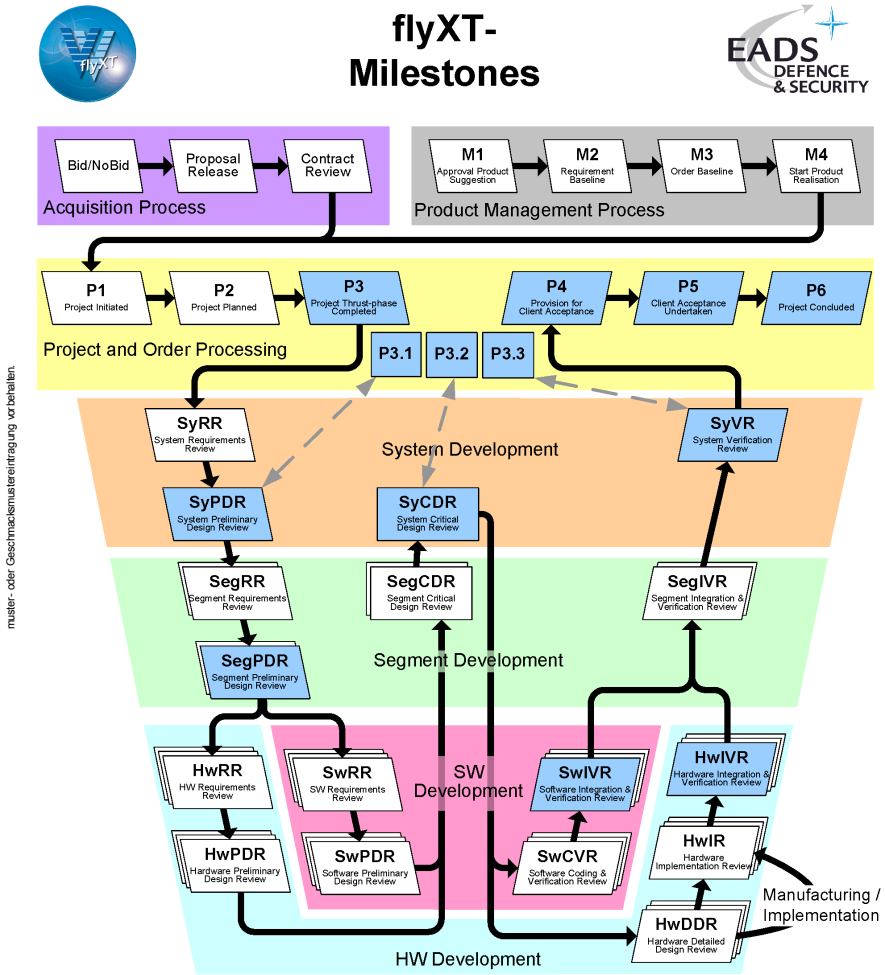
V-Modell XT (2)


Entscheidungspunkte - Produkte Systementwicklung
• Vertrag
• Gesamtsystemspezifikation
• Gefährdungs- und Systemsicherheitsanalyse
• Systemarchitektur
• Unterstützungssystemarchitektur
• Systemspezifikation
• Spezifikation log. Unterstützung
• Prüfspezifikation Systemelement
• Implementierungs-, Integrations- und Prüfkonzept System/Unterstützungssystem
• HW-Spezifikation und SW-Spezifikation
• HW-Architektur und SW-Architektur
• Logistisches Unterstützungskonzept
• Externe-Einheit-Spezifikation
• Prüfspezifikation Systemelement
Projekt beauftragt
Gesamtsystem
System spezifiziert
System, Segmente
System entworfen
Einheiten
Feinentwurf abgeschlossen
© Wolfgang Kranz: Systementwicklung Auftragnehmer
10

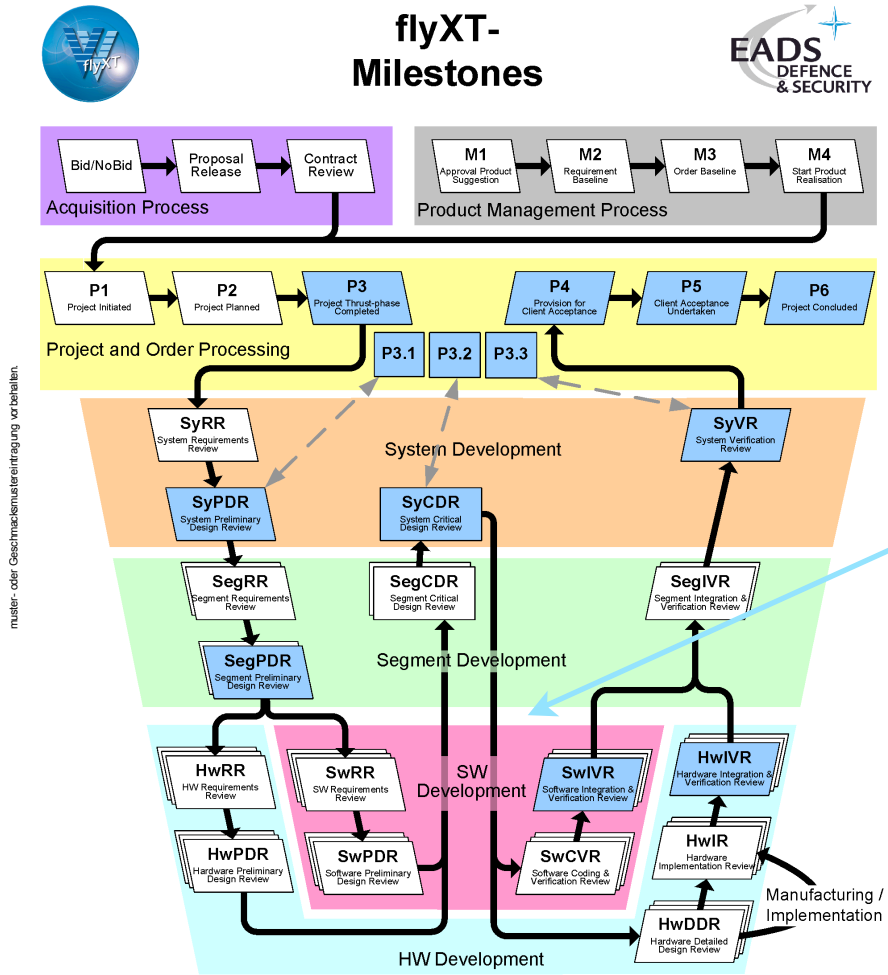
V-Modell XT (3)


Entscheidungspunkte - Produkte Systementwicklung
Abnahme erfolgt
• Prüfprotokoll Lieferung
• Abnahmeerklärung
Gesamtsystem
Lieferung durchgeführt
• Prüfprotokoll Systemelement
• Lieferung
System, Segmente
System integriert
• Prüfprotokoll Systemelement
• System mit allen Segmenten
• Logistische Unterstützungsdokumentation
Einheiten
Systemelemente realisiert
• Prüfprotokoll Systemelement
• HW-Einheiten
• SW-Einheiten
• Externe Einheiten
© Wolfgang Kranz: Systementwicklung Auftragnehmer

flyXT - EADS-Ausprägung des V-Modell XT



flyXT - EADS-Ausprägung des V-Modell XT



Software Entwicklung

Softwaretechnik, Prof. Partsch, Universität Ulm

V-Modell (5/7)

Aktueller Stand: V-Modell®XT (seit 2005)

Bereits eingesetzt: Adressverwaltung des Bundestags, Inpol-neu, Bordradar Eurofighter, ...

- Produktzentrierter Ansatz („Integriertes Produktmodell“)
  - **Produkte** (= Ergebnisse oder Zwischenergebnisse eines Projekts)
    - stehen im Mittelpunkt (> 100 Produkte in Produktgruppen geordnet)
    - haben definierte Abhängigkeiten
  - **Aktivitäten**
    - dienen dazu, Produkte zu erzeugen oder zu bearbeiten
      (90 Aktivitäten in 13 Aktivitätsgruppen, die entsprechenden Produktgruppen zugeordnet sind)
  - **Rollen**
    - beschreiben zusammengehörige Aufgaben, Verantwortlichkeit + benötigte Fähigkeiten
- Submodelle, Abstraktionsebenen und Erzeugnisstruktur i.w. wie bei V-Modell 97
  (zusätzliches Submodell: Problem- und Änderungsmanagement)

Früher: Teil des Konfigurationsmanagements

Softwaretechnik, Prof. Ilka Philippow, TU Ilmenau

Lernziele

1. Einführung in die Softwaretechnik (Zitat aus Skript)
- Einführung und Klärung von Grundbegriffen
- den Unterschied von Softwareprodukten zu anderen Produkten verstehen und erklären können
- den Softwarelebenszyklus kennen
  - Phasen, Aktivitäten, Produkte
  - Methoden und Konzepte

2. Vorgehensmodelle (Zitat aus Skript)
- die prinzipiellen Phasen und Aktivitäten der Softwareentwicklung und ihre Produkte verstehen und erklären können
- die Unterschiede, Vor- und Nachteile verschiedener Vorgehensmodelle kennen
- die Schritte zur Vorgabe eines Prozessmodells kennen

Vorgehensmodell

Allgemeiner Vorgehensrahmen

- benennt und beschreibt allgemein die Art und Anordnung von Aktivitäten im Rahmen der SW-Entwicklung
- man unterscheidet
  - sequenzielle, streng phasenorientierte Modelle
  - nichtlineare evolutionäre Modelle
- wird in der Praxis für konkrete Projekte und Unternehmen näher spezifiziert

→ konkretes Prozessmodell festlegen

Prozessmodell

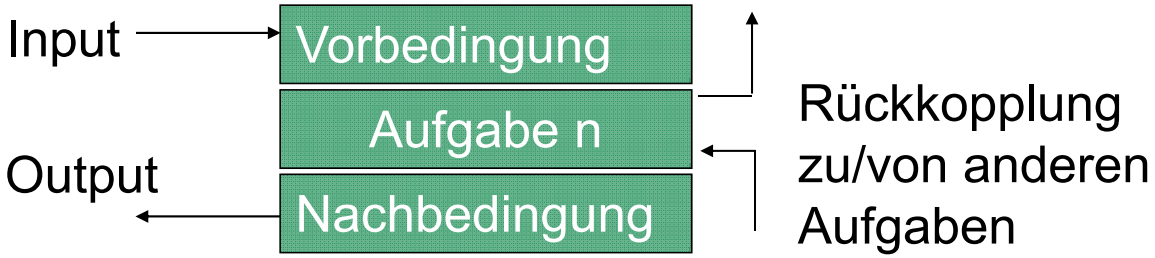
Konkrete Instanz eines Vorgehensmodells

- definiert die konkreten Aktivitäten, ihre Reihenfolge und ihre Produkte für die Durchführung eines Projektes
- Definition ist abhängig von
  - Projekttyp und
  - Unternehmenspolitik und -kompetenz
- die Aktivitäten müssen durch Methoden und Werkzeuge unterstützt werden

→ abhängig vom Vorgehensparadigma

Festlegung von Prozessmodellen

1. Definition der Aktivitäten (wer, was, wie, wann)



2. Auswahl von Methoden und Werkzeugen

→ erzeugen Produktmodelle

- strukturierte Methoden und Konzepte
- objektorientierte Methoden und Konzepte

Vorgehensmodelle im Software Engineering

- Dr. Marco Kuhrmann, TU München

- TU München - Fakultät für Informatik Lehrstuhl IV: Software & Systems Engineering (Prof. Manfred Broy)
- Vorlesung | Wintersemester 2010/2011 (2+1, 5 ETCS-Punkte)
- http://www4.in.tum.de/lehre/vorlesungen/vgmse/ws1011/
- Inhalt

  Die Vorlesung vermittelt die grundlegenden Techniken und Methoden der Projektorganisation und des Projektmanagements für die Entwicklung großer Softwaresysteme.

Vorgehensmodelle im Software Engineering

- Grundlagen Was ist ein Vorgehensmodell?
  - Einordnung von Vorgehensmodellen in Prozessladschaften
- Übersicht Vorgehensmodelle
  - Klassische Ansätze
  - Agile Modelle/Methoden
  - Strukturierte Modelle
  - Weitere Modelle
  - Werkzeugunterstützung
- Lebenszyklus von Vorgehensmodellen Phasenübersicht (im Folgenden vertieft)
- Analyse
  - Analyse und Optimierung von Prozessen
  - Stakeholder und Rollen
  - Vorgänge/Abläufe
  - Artefakte/Produkte
  - Strukturierung von Prozessen

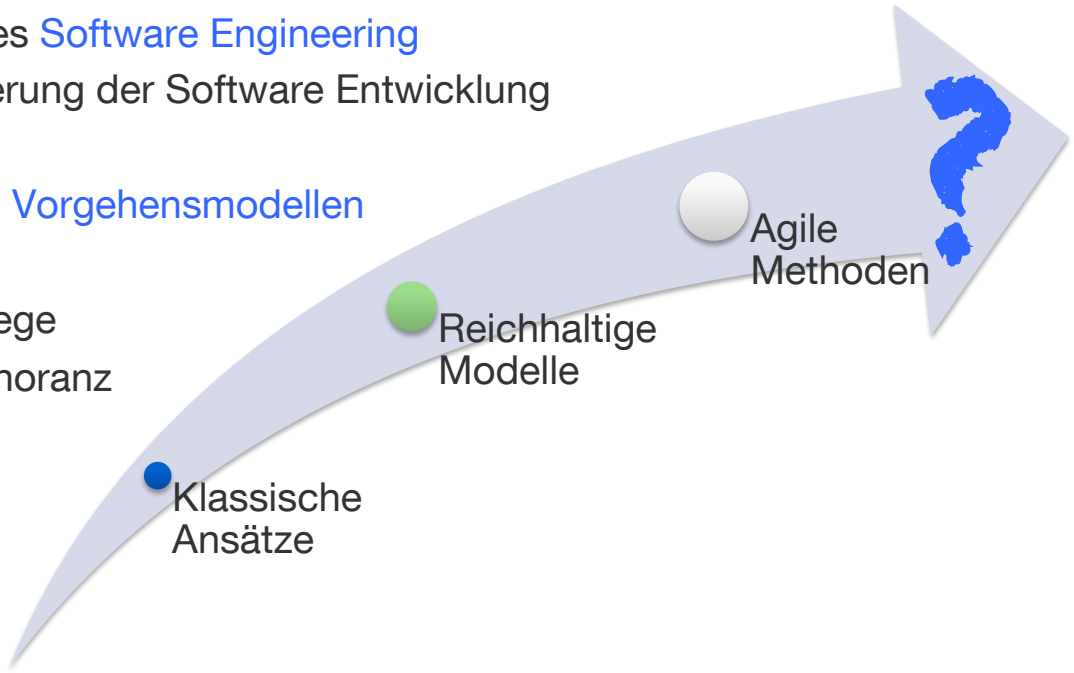
Vorgehensmodelle im Software Engineering

- Konstruktion und Anpassung
  - Konstruktionsoptionen
  - Vorgehensmetamodelle
  - Umsetzung und Bereitstellung
- Einführung
  - Einführungsstrategien
  - Planung und Maßnahmen
  - Schulungen
  - Etablieren von Feedbackschleifen
- Weiterentwicklung und Pflege
  - Prozessreife
  - Audits, Assessments und Zertifizierung
  - Planung von Prozessverbesserungen
  - Ausgewählte Modelle und Methoden

Inhalte und Ziele der Vorlesung (1)

- Vorgehensmodelle existieren seit den frühen 70‘ern
  - Ziel: „Raus aus der Software Krise“
    → Geburt des Software Engineering
  - Systematisierung der Software Entwicklung

- Entwicklung von Vorgehensmodellen
  - Inflation
  - Glaubenskriege
  - Trend zur Ignoranz



Inhalte und Ziele der Vorlesung (2)

- Ziele
  - Organisations- und Projektprozesse verstehen
  - Vorgehensmodelle verstehen
  - Prozesse planen, abbilden, definieren und umsetzen
  - Handhabung von Werkzeugen und Infrastrukturen für Prozessanpassungen erlernen

- Praktischer Anwendungskontext
  - Prozessoptimierendes Projektmanagement
    - Projekt- und Teamorganisation
    - Effektivitätsfeststellung und Verbesserung
  - Prozessanalyse und Beratung

Inhalte und Ziele der Vorlesung (3)

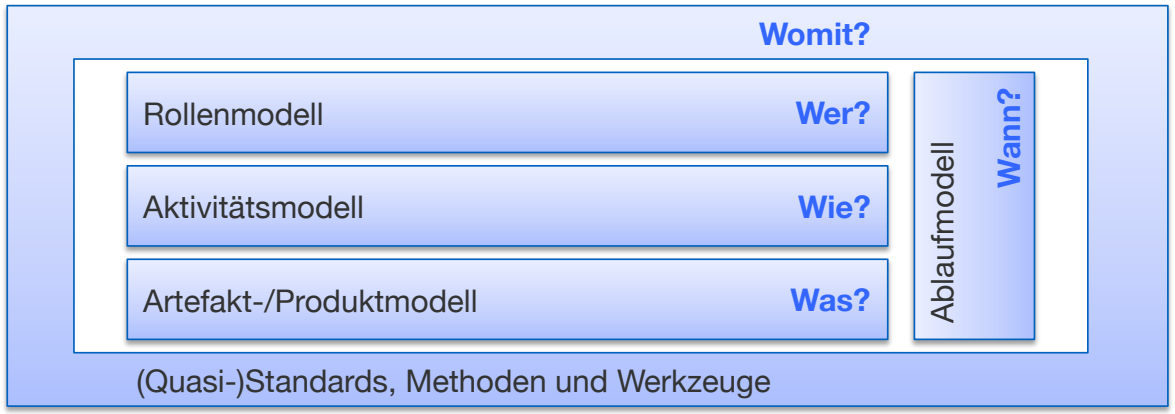
1. Grundlagen
   → *Begriffsklärung und Einordnung*
2. Übersicht Vorgehensmodelle
   → *Aufstellung/Bewertung existierender Ansätze*
3. Lebenszyklus von Vorgehensmodellen
   → *Überblick über die Phasen (Vertiefung im Anschluss)*
4. Analyse
   → *Analyse und Optimierung von Prozessen, Stakeholder, Modelle etc.*
5. Konstruktion und Anpassung
   → *Metamodellierung und Modellierung von Vorgehensmodellen*
6. Einführung
   → *Einführungsmaßnahmen, Planung, Schulung etc.*
7. Weiterentwicklung und Pflege
   → *Prozessreife, Planung, Übersicht Reifegradmodelle*

Was ist ein Vorgehensmodell?

Definition

Ein Vorgehensmodell beschreibt systematische, ingenieurmäßige und quantifizierbare Vorgehensweisen, um Aufgaben einer bestimmten Klasse wiederholbar zu lösen.

Submodelle



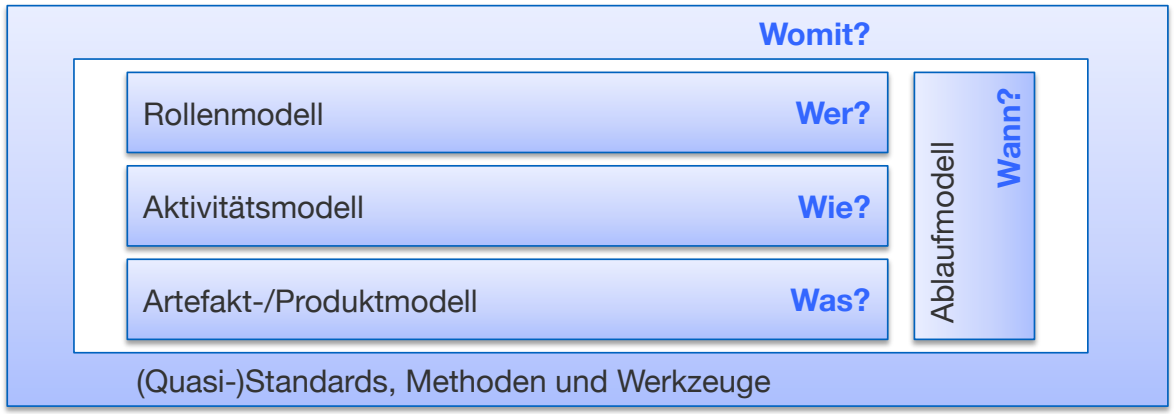
1. Definition der Aktivitäten (wer, was, wie, wann)

Was ist ein Vorgehensmodell?

Definition

Ein Vorgehensmodell beschreibt systematische, ingenieurmäßige und quantifizierbare Vorgehensweisen, um Aufgaben einer bestimmten Klasse wiederholbar zu lösen.

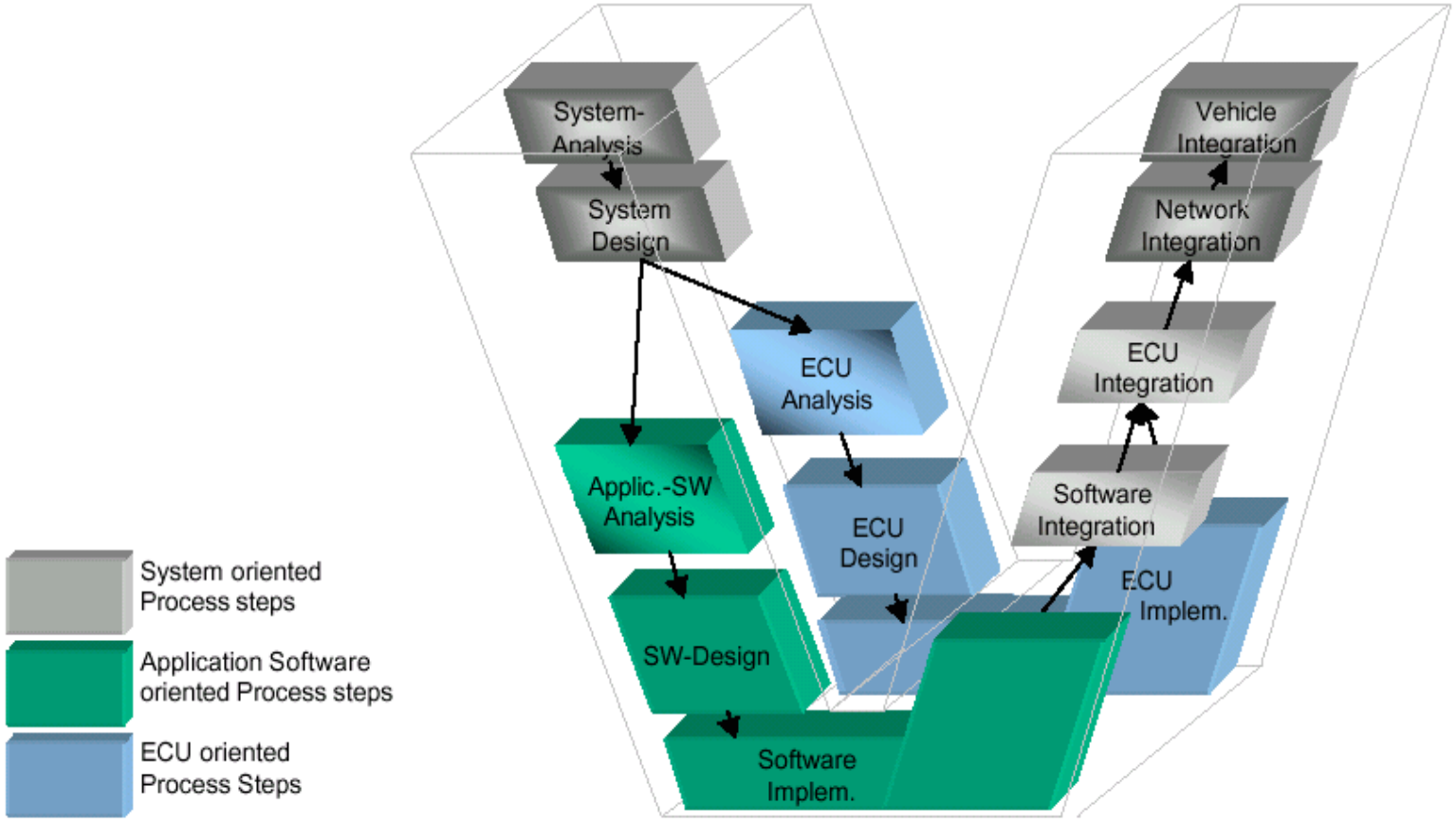
Submodelle



Sichten auf das Vorgehensmodell

- Wer?
  Rollenorientierte Sicht:
  Wer ist für welche Arbeitsergebnisse (Produkte, Work Products, Artefakte) verantwortlich?
- Was?
  Produktorientierte Sicht:
  Was sind die zu erarbeitenden Produkte?
- Wie?
  Aktivitätenorientierte Sicht:
  Wie werden die Produkte erarbeitet?
- Wann?
  Phasenorientierte Sicht:
  Wann werden welche Arbeitsschritte durchgeführt?
  Meilensteinorientierte Sicht:
  Wann werden welche Produkte fertiggestellt und geprüft?
- Womit?
  Methoden- und werkzeugorientierte Sicht:
  Womit (Methoden und Werkzeuge) werden die Produkte erarbeitet?

Vereinfachtes V-Modell



Sichten auf das Vorgehensmodell - Beispiel

- Wer?
  Rollenorientierte Sicht:
  Der SW-Architekt ist für die SW-Architektur verantwortlich
- Was?
  Produktorientierte Sicht:
  Die SW-Architektur beschreibt die Zerlegung in Komponenten, deren Schnittstellen und Wechselwirkungen
- Wie?
  Aktivitätenorientierte Sicht:
  Die SW-Architektur wird nach funktionalen Gesichtspunkten unter Berücksichtigung von Modularität und Kapselung entworfen
- Wann?
  Phasenorientierte Sicht:
  Die SW-Architektur wird in der Phase SW-Design entworfen.
  Meilensteinorientierte Sicht:
  Die SW-Architektur wird zum Meilenstein SW-Design Review fertiggestellt und geprüft

Sichten auf das Vorgehensmodell - Beispiel

- Womit?
  Methoden- und werkzeugorientierte Sicht:
  Die SW-Architektur wird entsprechend den firmeninternen Entwurfsrichtlinien mit unterstützenden Werkzeugen erstellt.