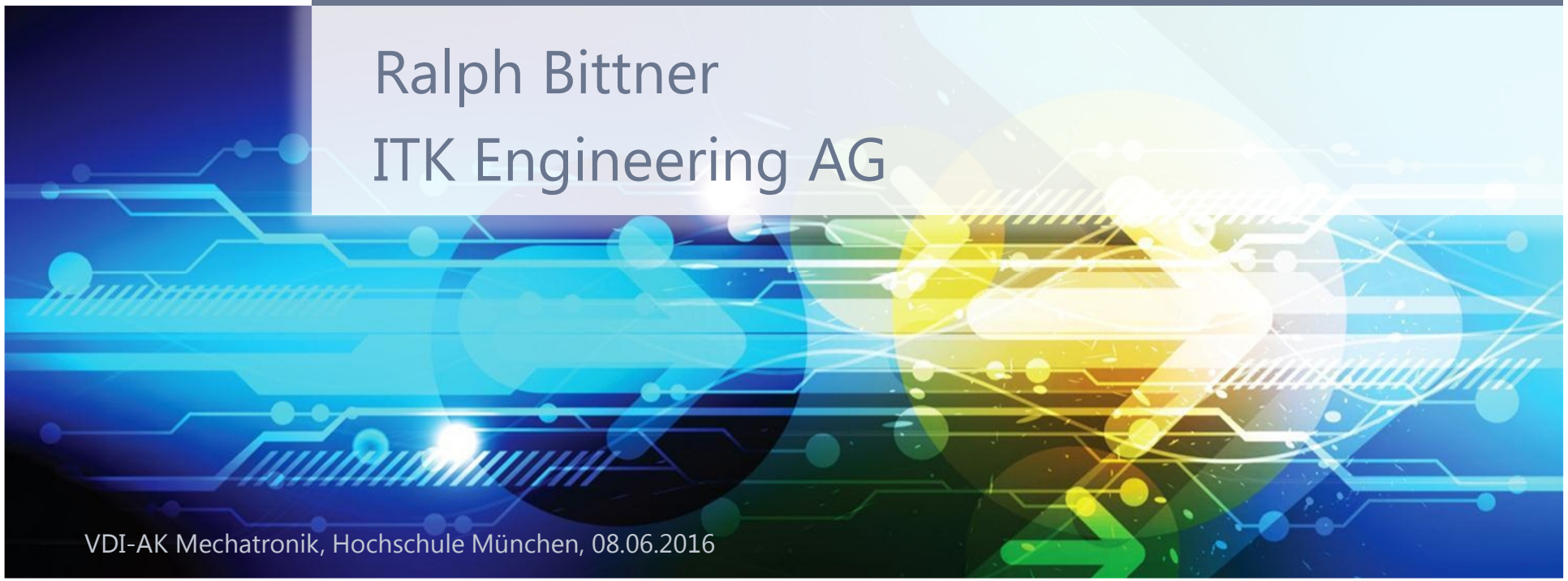
Modellbasierte Software-Entwicklung und HiL-Testing

Ralph Bittner
ITK Engineering AG



VDI-AK Mechatronik, Hochschule München, 08.06.2016

Herausforderungen in der Softwareentwicklung

Hohe Systemkomplexität

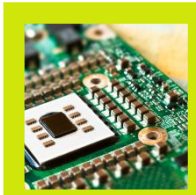
Hoher Automatisierungsgrad
Normgerechte Traceability

Kürzere Entwicklungszeiten

Simulation & Codegenerierung
Parallele Entwicklung von Software & Hardware

Steigende Variantenvielfalt

Wiederverwendbarkeit von Modellen
Bessere Wartbarkeit



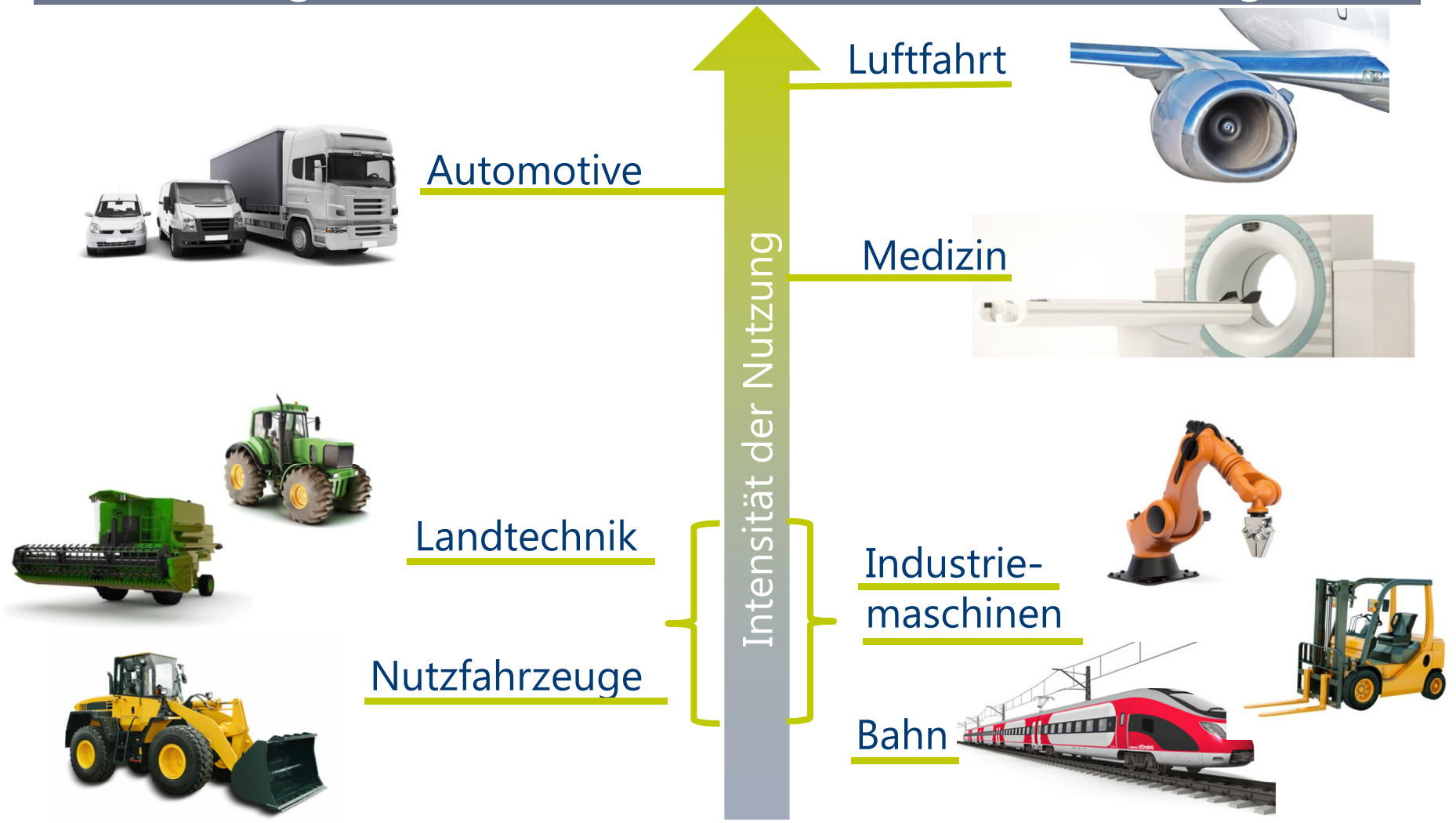
Senken der Entwicklungskosten

Frühe Fehlererkennung
Hohe Durchgängigkeit

… und wie modellbasierte Methoden Abhilfe schaffen.

Wo wird Modellbasierte SW-Entwicklung eingesetzt?

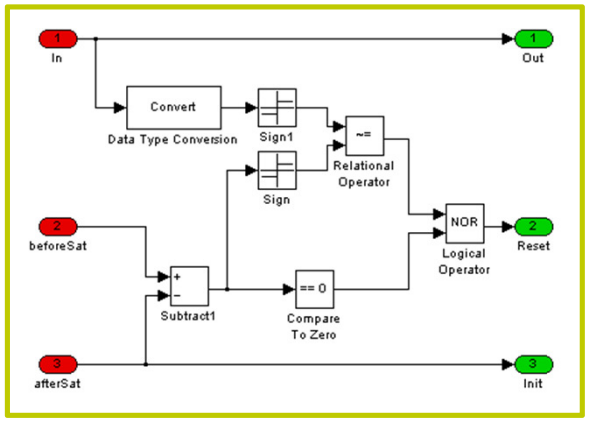
Anwendung nach Branchen (Simulation, Virtualisierung)



Was ist Modellbasierte SW-Entwicklung?

Eine neue Abstraktionsebene

Modell



Hochsprache

```
if ((real_T)rtb_Abs < 0.0) {
      rtb_Buffer = -1;}
else {
      rtb_Buffer =
      (real_T)rtb_Abs > 0.0 ?
      1 : 0;}
rtb_minxy =
      (int16_T)(rtb_Sum1 >> 16);
```

Codegenerator

Maschinensprache

```
mov     di,num1+digits-1
mov     si,num2+digits-1
mov     cx,digits
call    AddNumbers
mov     bp,num2
call    calc_p
dec     dword [term]
jz      .done
```

Compiler

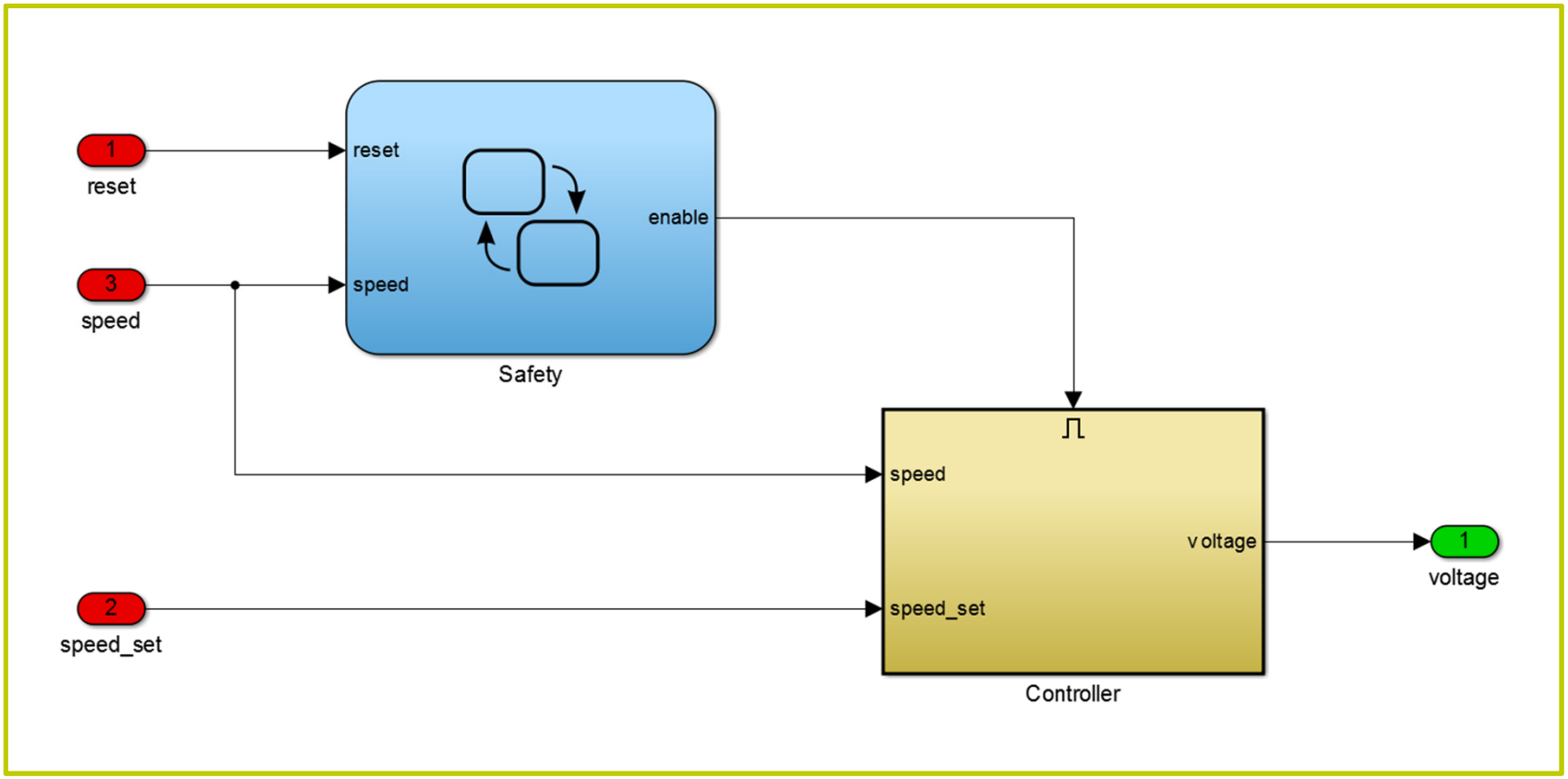
Modellbasiert vs. klassisch

SW-Entwicklung	Modellbasiert	Klassisch
Spezifikation	▪ Grafisch, ausführbar ▪ Single Source	Textform
Simulation	Simulation von Komponenten und Gesamtsystemen (Model-in-the-Loop, MiL)	n/a
Code	Automatische Generierung	Manuelle Implementierung
Test	▪ Frühzeitig auf Spezifikationsebene ▪ Monitoring implementierter Funktionen in der Simulation	Nach Implementierung aller abhängigen Software-Module

Ausführbare Spezifikation

Architektur

- Beziehungen und Hierarchie großer Softwareeinheiten

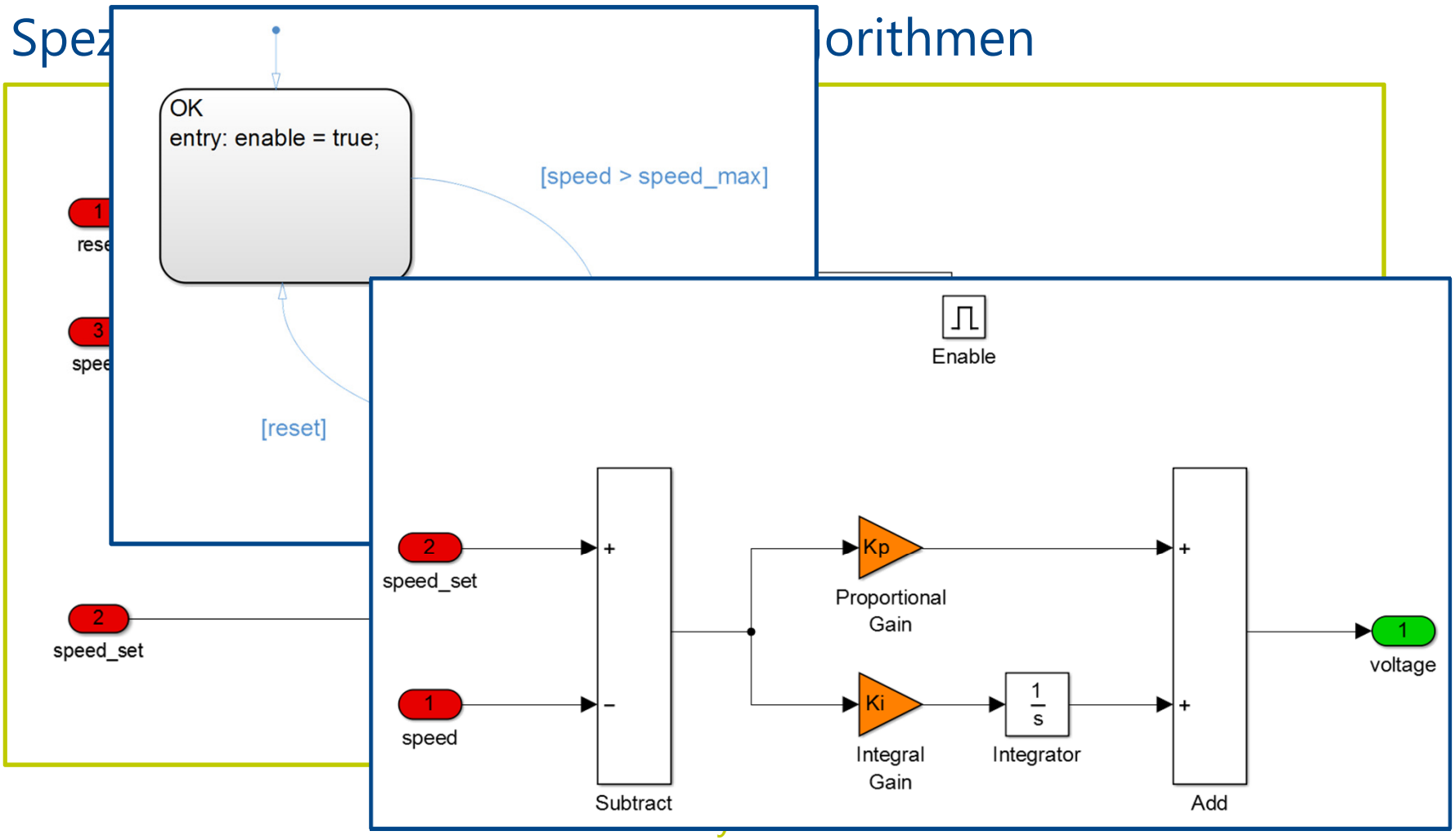


Control System

Ausführbare Spezifikation

Software-Design

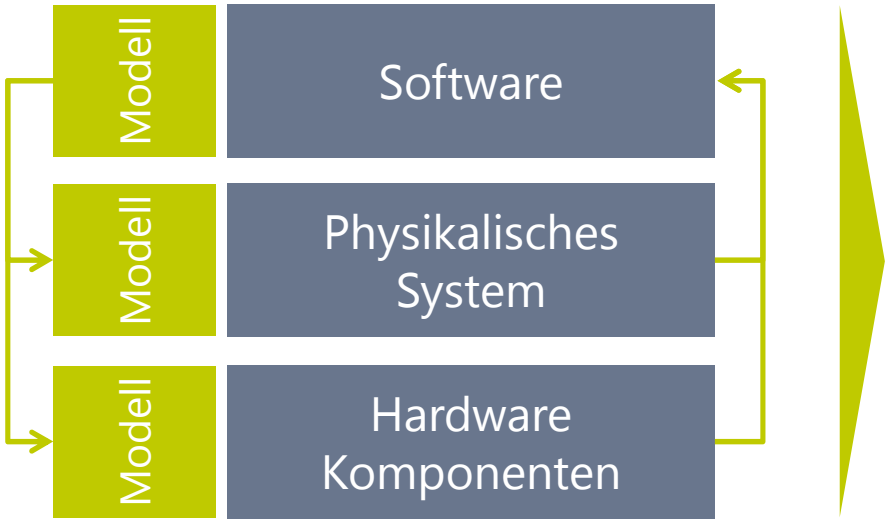
- Spez orithmen



Simulation (Model-in-the-Loop, MiL)

„Simulation ist der Prozess, ein **Modell eines realen Systems** zu entwerfen und damit Experimente durchzuführen, die entweder dazu dienen, das Systemverhalten und die **zugrundeliegenden Prinzipien zu verstehen** oder verschiedene Entwürfe eines künstlichen Systems oder von **Strategien zum Betrieb des Systems zu bewerten**."

Robert E. Shannon. Systems Simulation: The Art and Science. (1975)



Großes Einsparpotenzial an Zeit/Kosten

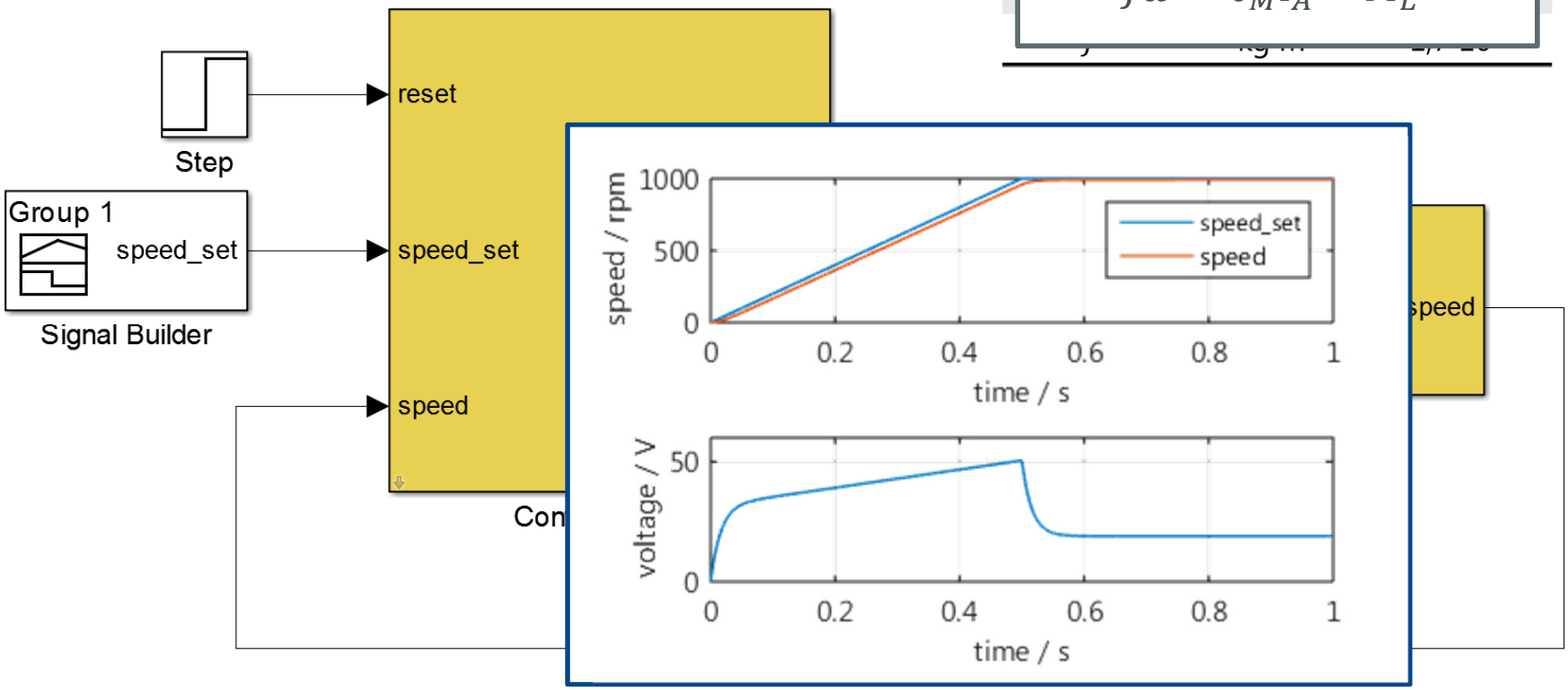
- Schnelle Fehlereingrenzung
- Parallele Entwicklung Hardware/Software

Simulation (Model-in-the-Loop, MiL)

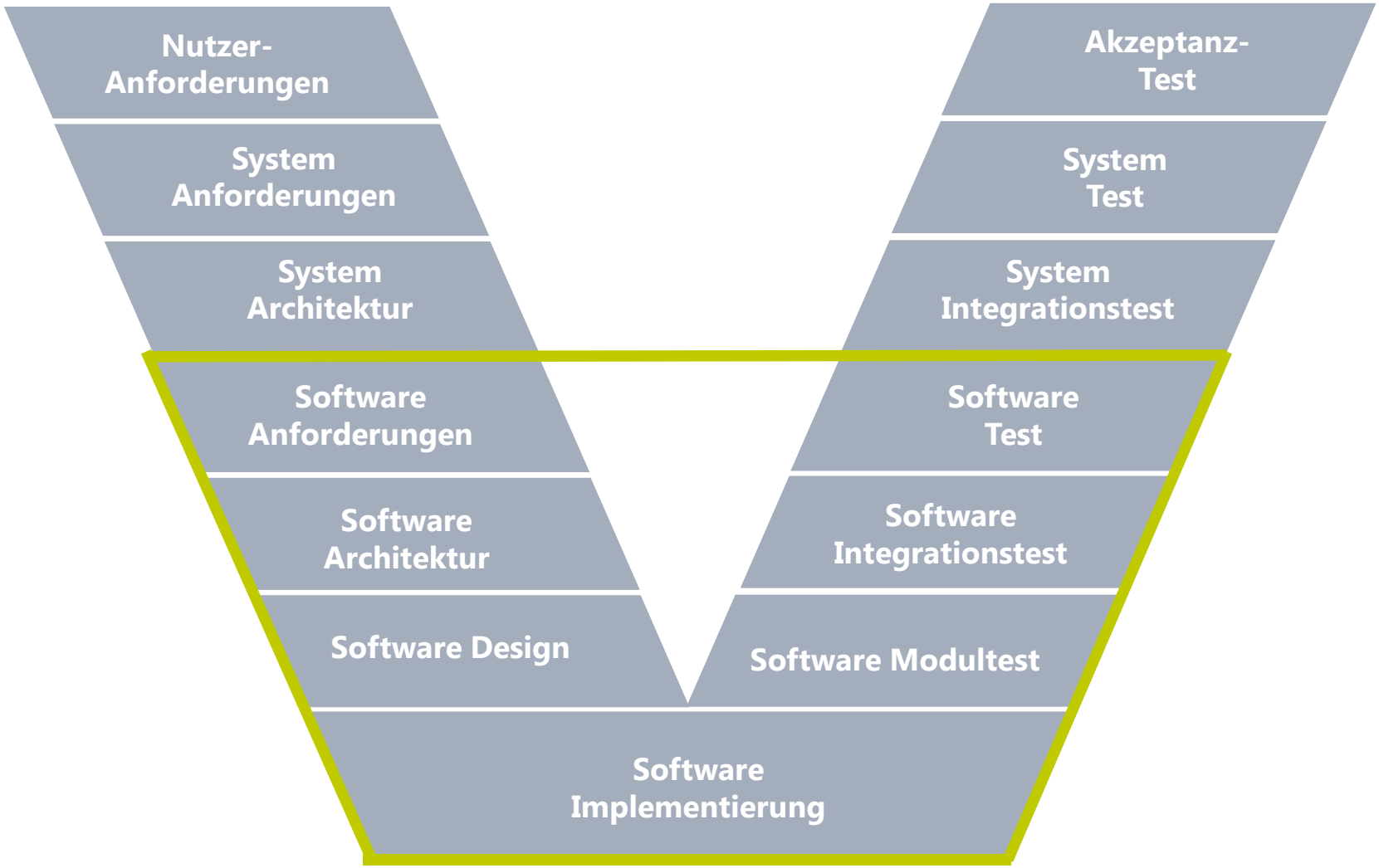
Beispiel

Parameter	Einheit	Wert

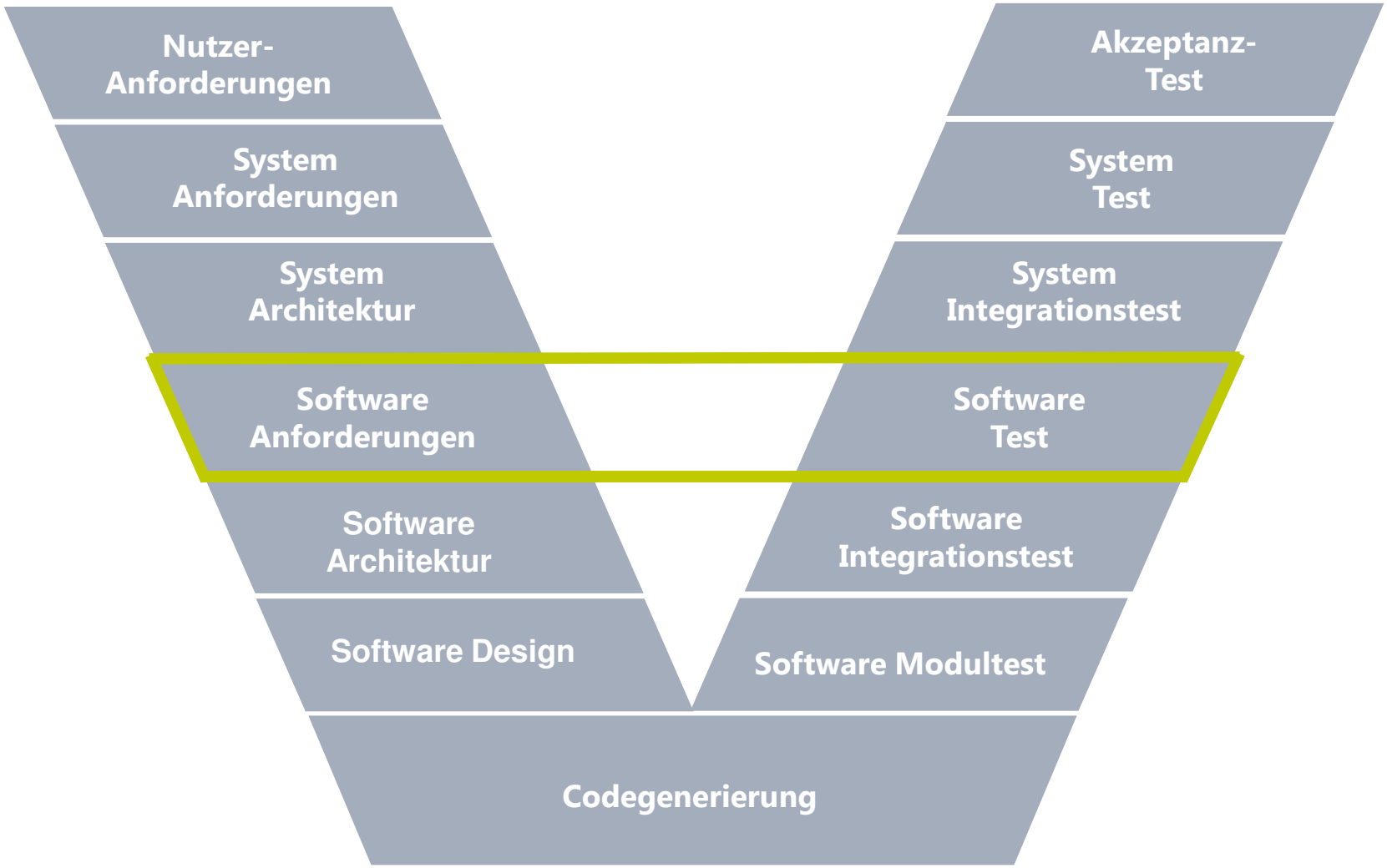
$$L_A\dot{I}_A = U_A - R_A I_A - c_M\omega$$
$$J\dot{\omega} = c_M I_A - M_L$$



Klassischer Entwicklungsprozess



Modellbasierter Entwicklungsprozess



Was bieten Tools?

Framework

- Normgerechtes Testen
  - Überwachung der Testabdeckung
  - Formale Analysen
    (Korrektheit des Codes, keine Laufzeitfehler)
- Automatische Testfallgenerierung
  - Test der Software gegen Modell

Test-Tools

- Testautomatisierung
- Testmodi
  - True Double Override (TDO)
  - Software-in-the-Loop (SiL)
  - Processor-in-the-Loop (PiL)

Testmodi

True Double Override (TDO)

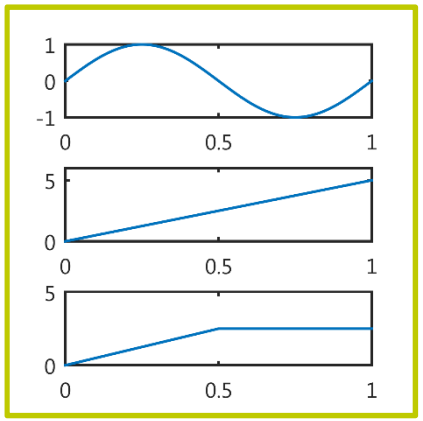
Zahlenformat	Eigenschaften	Gefahren
Festkomma	Begrenzter Wertebereich	Überläufe
	Begrenzte Präzision	Informationsverlust
Gleitkomma (einfache Präzision)	Großer Wertebereich	Überläufe (seltener)
	Kleine Zahlen präziser als große	Informationsverlust wird unterschätzt

TDO ersetzt Zahlen durch präzisestes Zahlenformat, Fehler werden sichtbar.

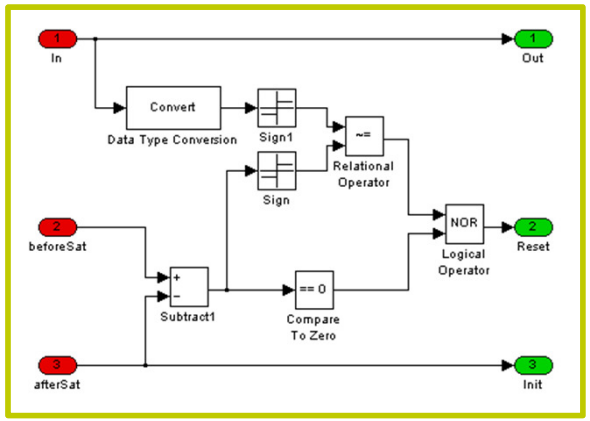
Test-Modi

Software-in-the-Loop (SiL)

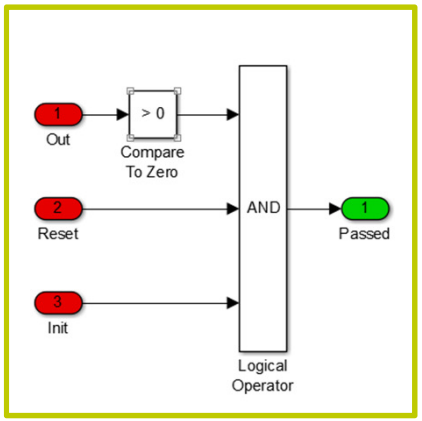
- Test des generierten Codes statt des Modells
- Kein zusätzlicher Aufwand(!)



Stimuli



Generiertes Modell Code



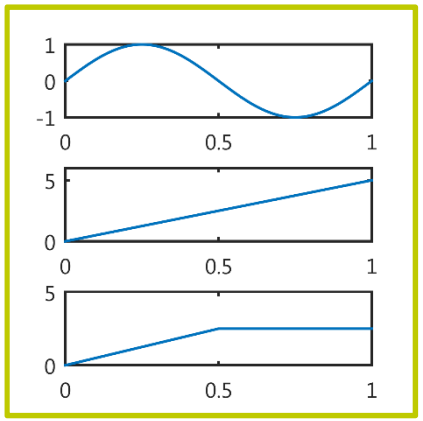
Auswertung

SiL testet generierten Code, Fehler bei Codegenerierung werden sichtbar.

Test-Modi

Processor-in-the-Loop (PiL)

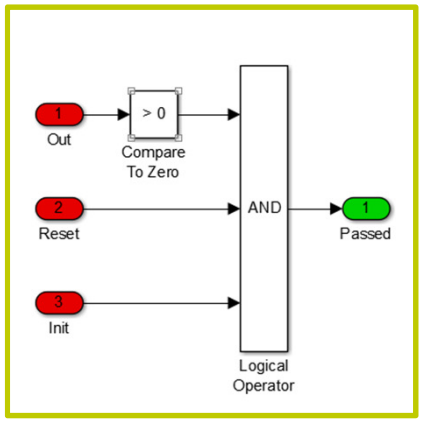
- Test des compilierten Codes
- Kein zusätzlicher Aufwand(!)



Stimuli

```
mov     di,num1+digits-1
mov     si,num2+digits-1
mov     cx,digits
call    AddNumbers
mov     bp,num2
call    calc_p
dec     dword [term]
jz      .done
```

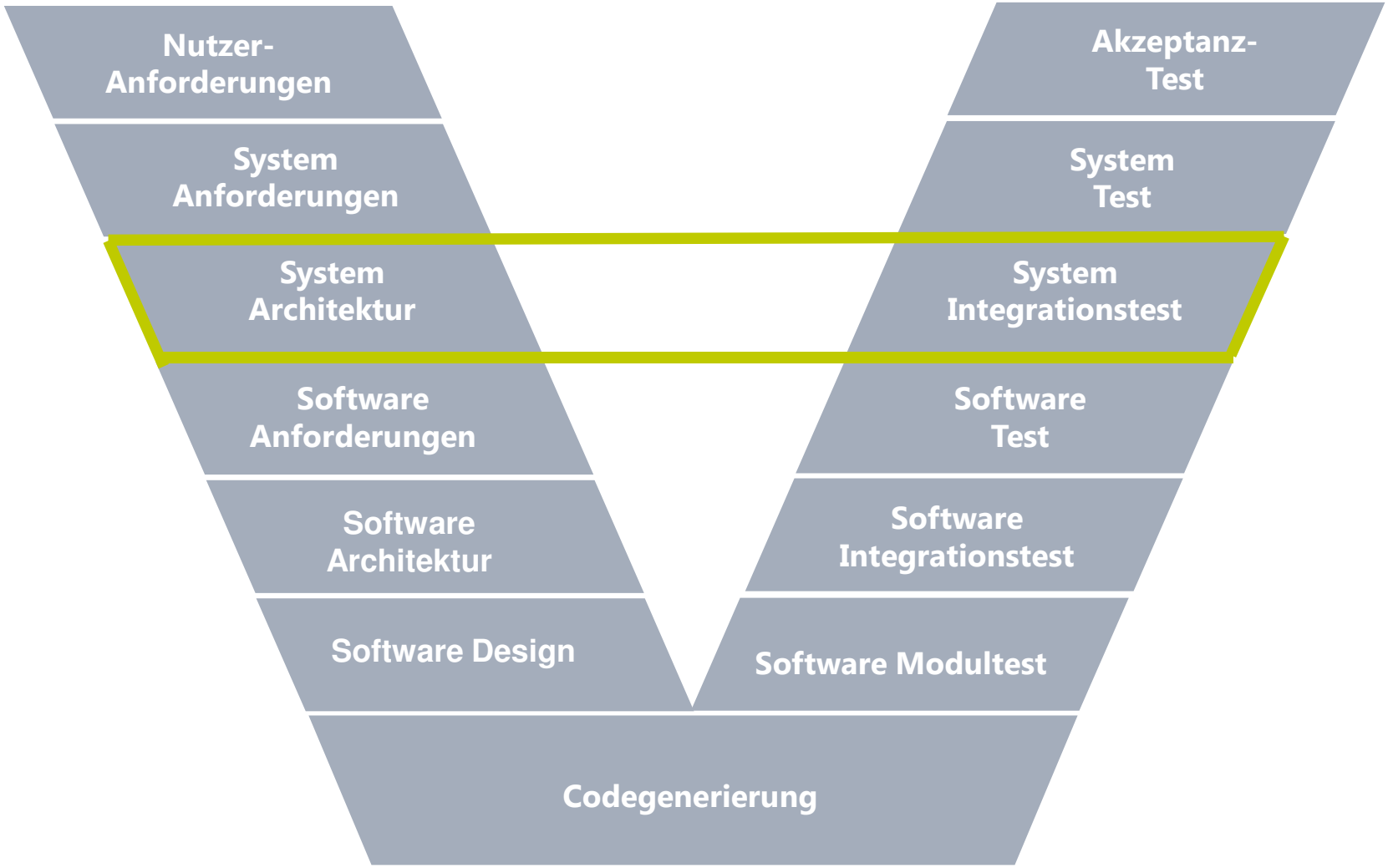
Compilierter Code



Auswertung

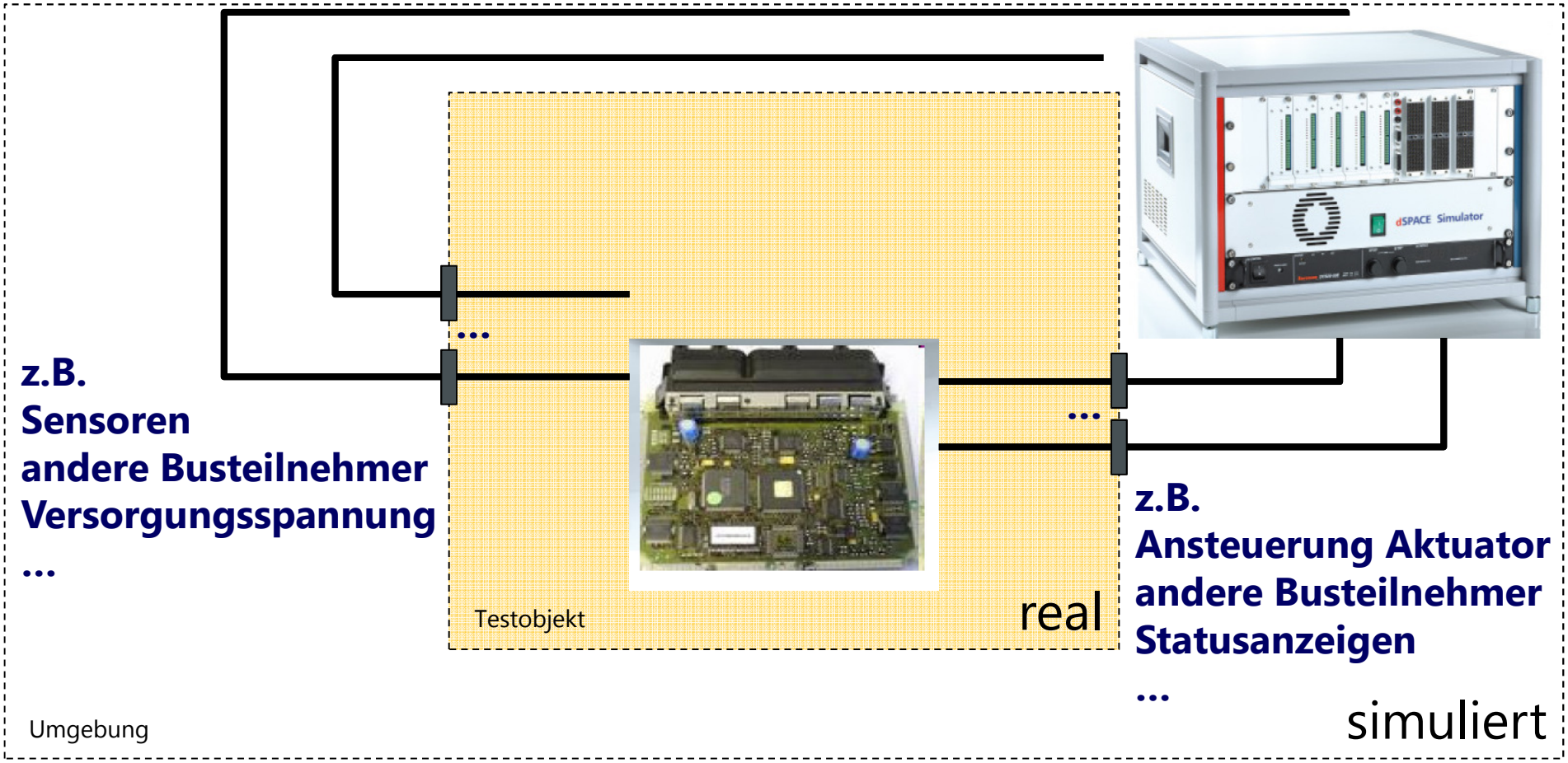
PiL testet Maschinencode, Einfluss von Compiler und realer Hardware werden sichtbar.

HiL-Testing



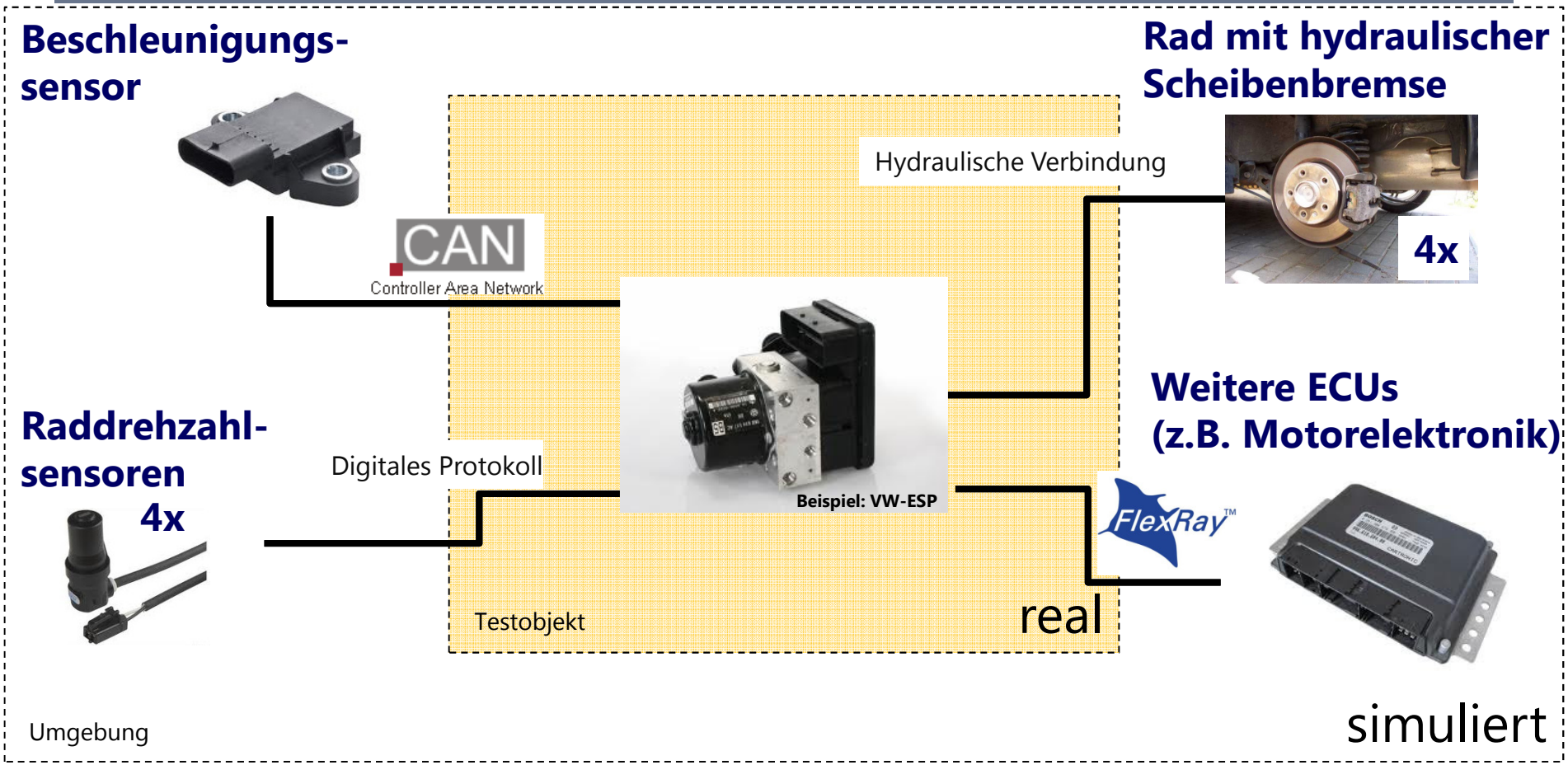
Hardware-in-the-Loop: Was ist das?

Schnittstellen eines HiL-Systems: Überblick



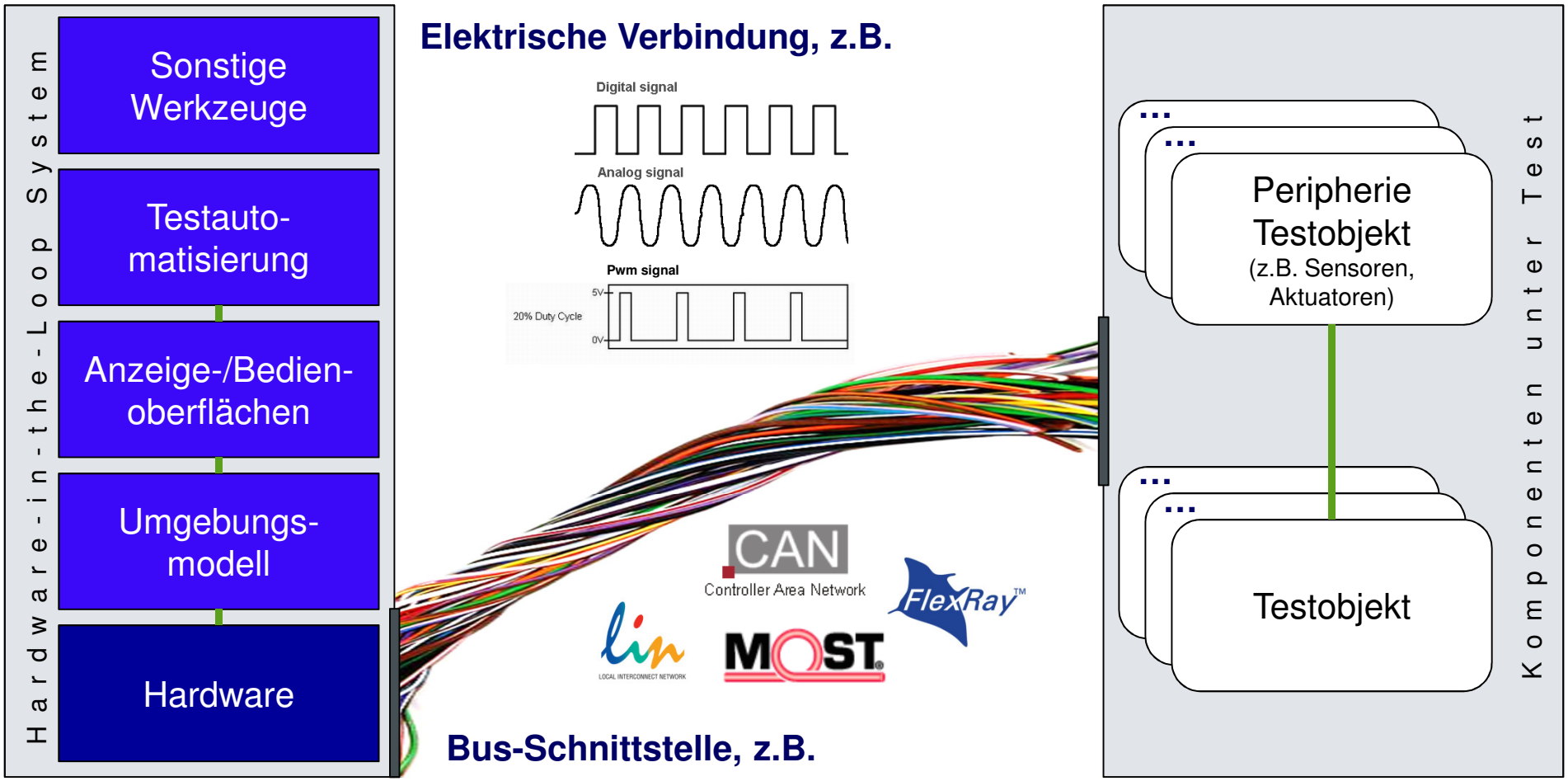
Automotive Softwareentwicklung

Schnittstellen eines HiL-Systems, Beispiel 1: ESP-Steuergerät



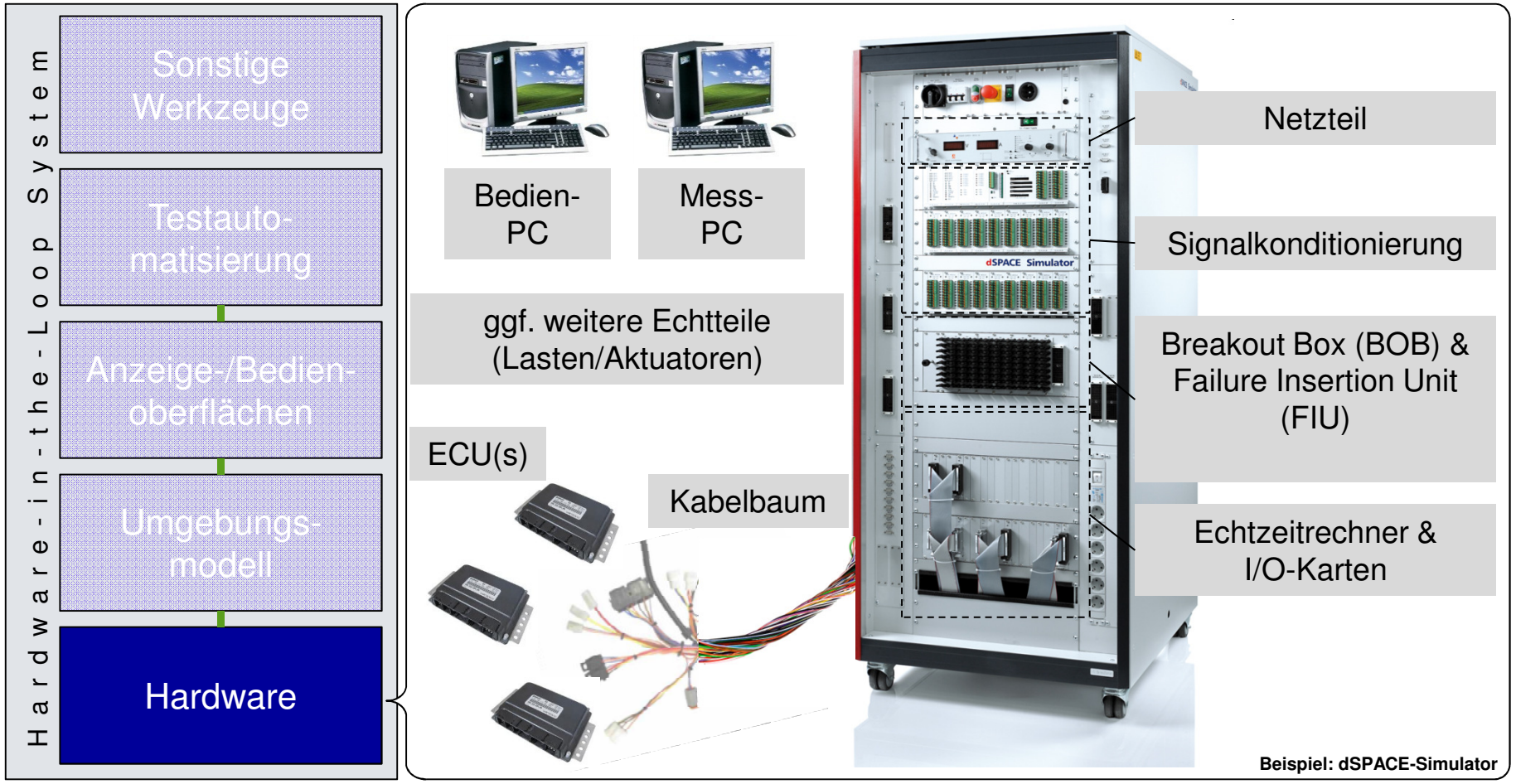
Automotive Softwareentwicklung

Teilkomponenten eines HiL-Systems: Überblick



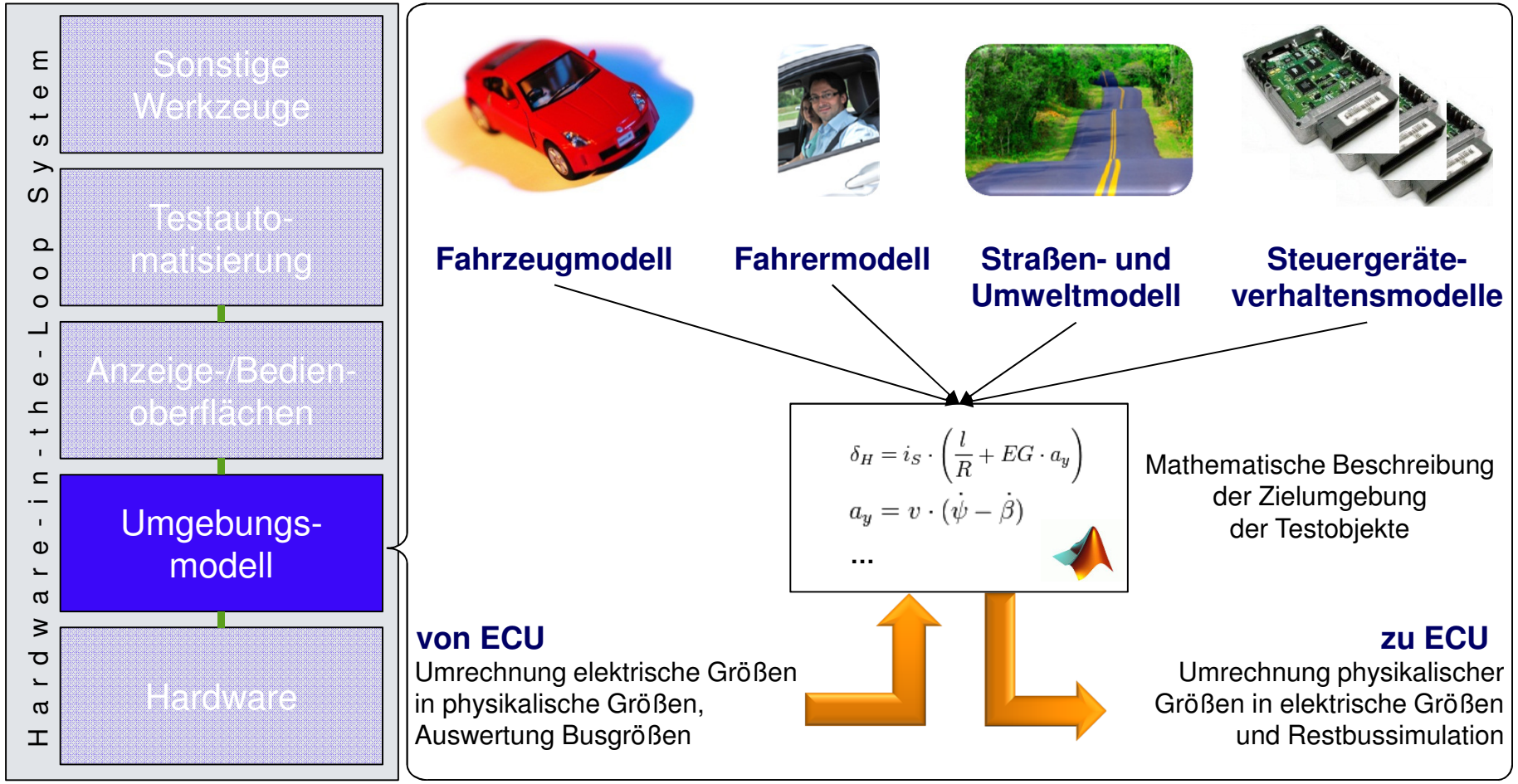
Automotive Softwareentwicklung

Teilkomponenten eines HiL-Systems: Hardware



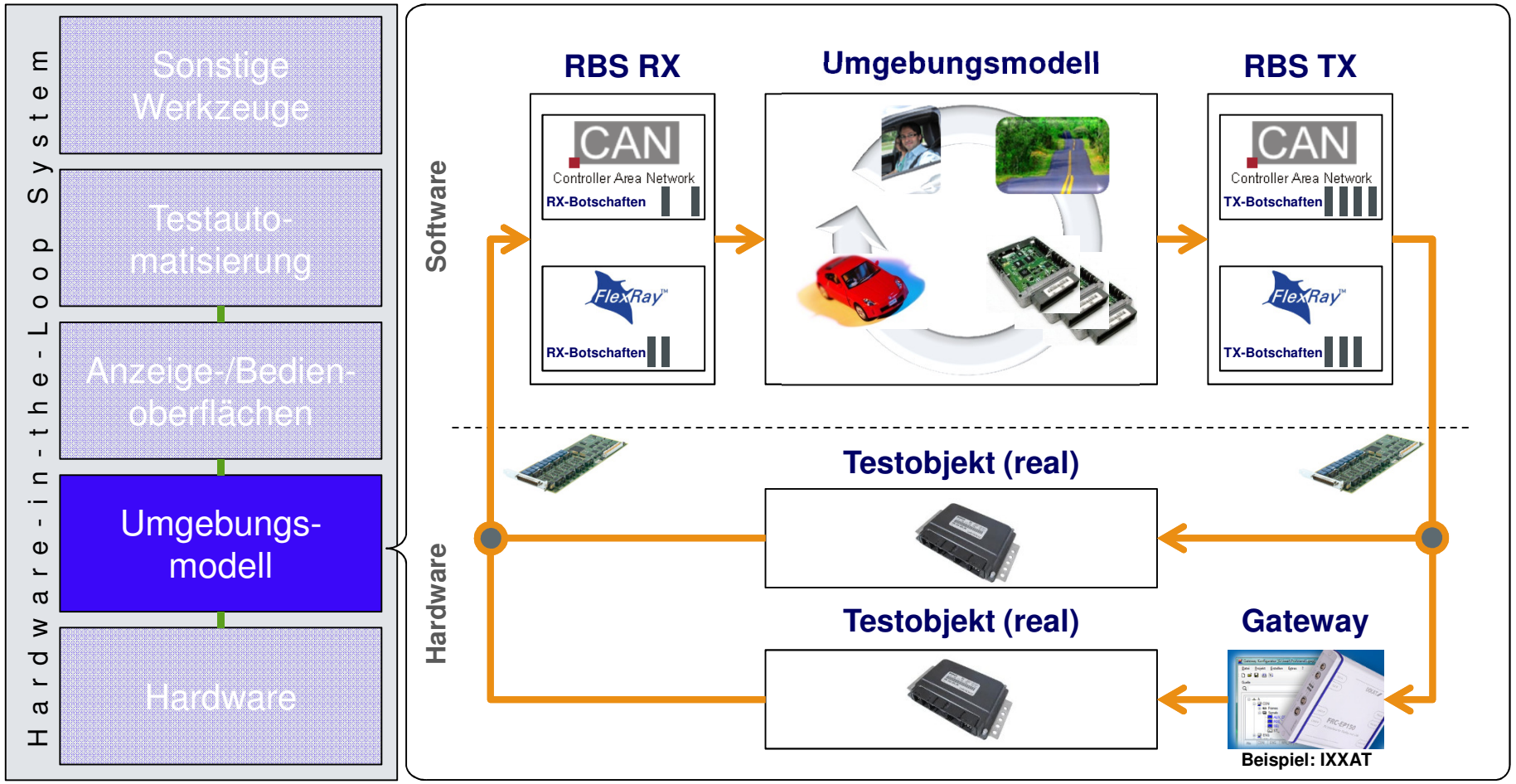
Automotive Softwareentwicklung

Teilkomponenten eines HiL-Systems: Umgebungsmodell



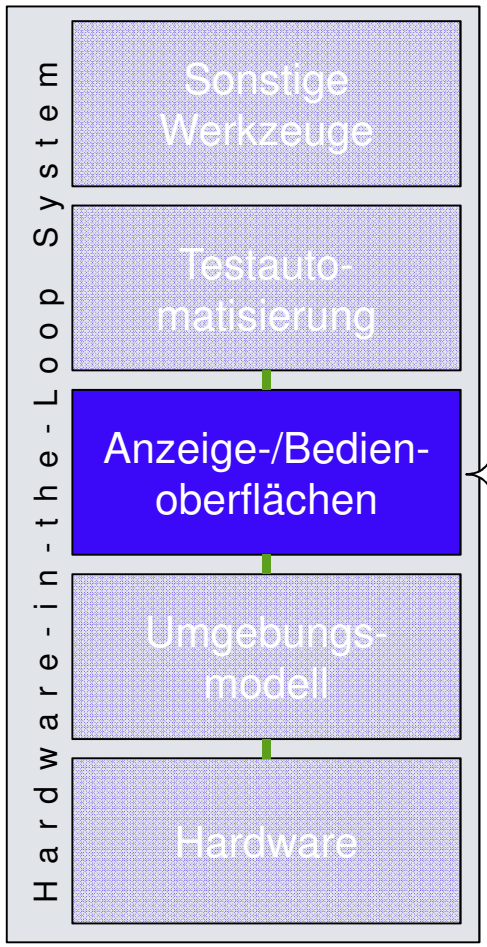
Automotive Softwareentwicklung

Teilkomponente eines HiL-Systems: Restbussimulation

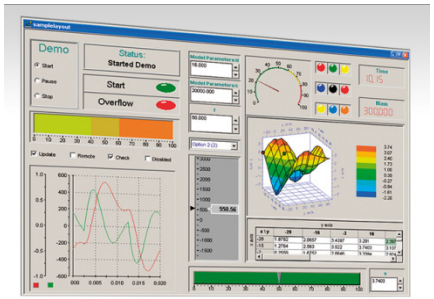


Automotive Softwareentwicklung

Teilkomponenten eines HiL-Systems: Anzeige- und Bedienoberflächen

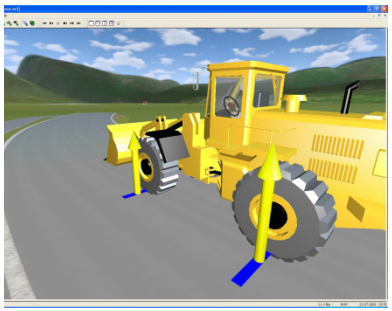


- **Grafische und numerische Anzeige von Signalen des Echtzeitmodells und der Restbussimulation**
- **Logging von Signalen des Echtzeitmodells**
- **Manipulation von Modellparametern während der Laufzeit des Echtzeitmodells**
- **Teilweise auch 3d-Animation**



dSPACE ControlDesk

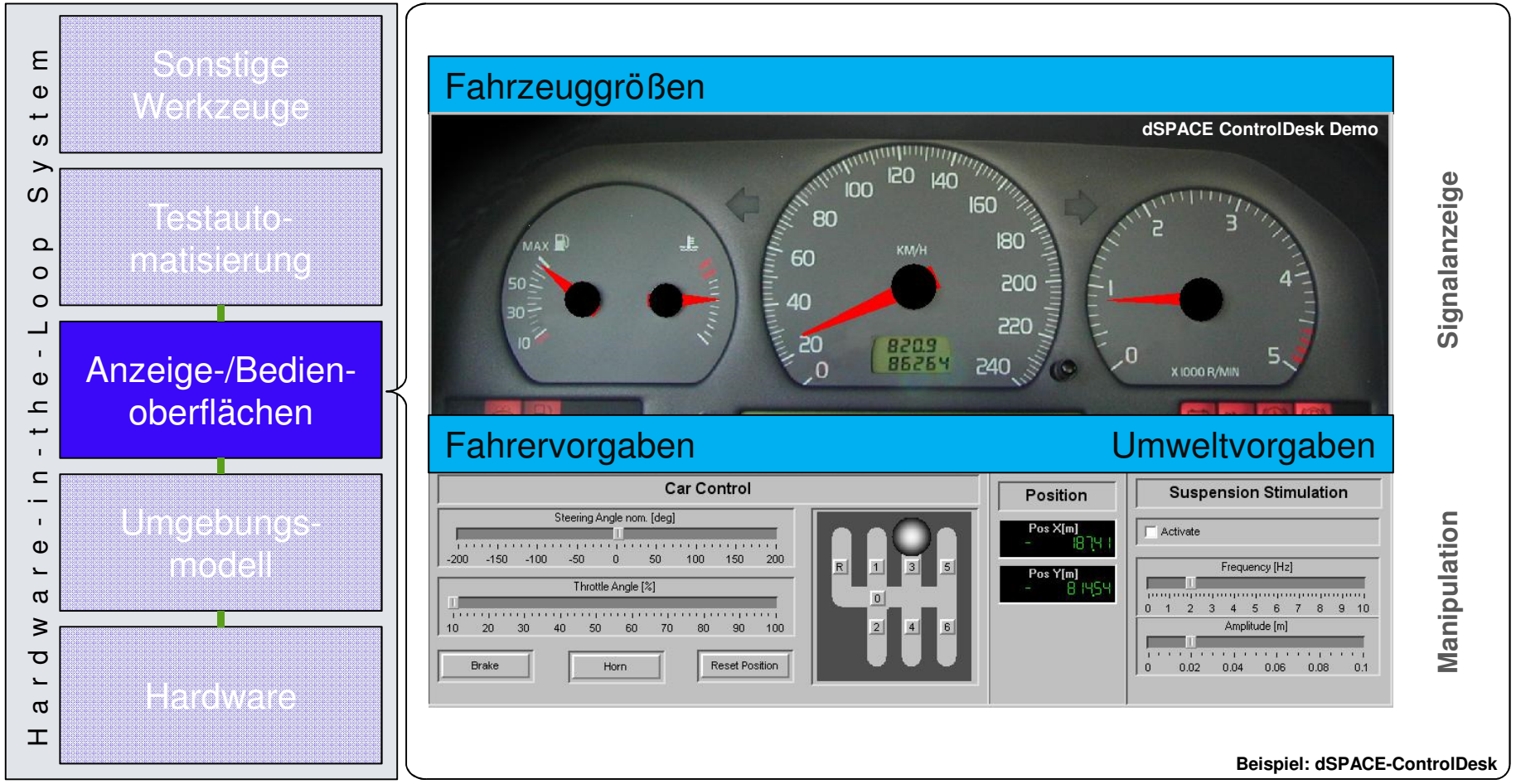
ETAS LABCAR OPERATOR



dSPACE MotionDesk

Automotive Softwareentwicklung

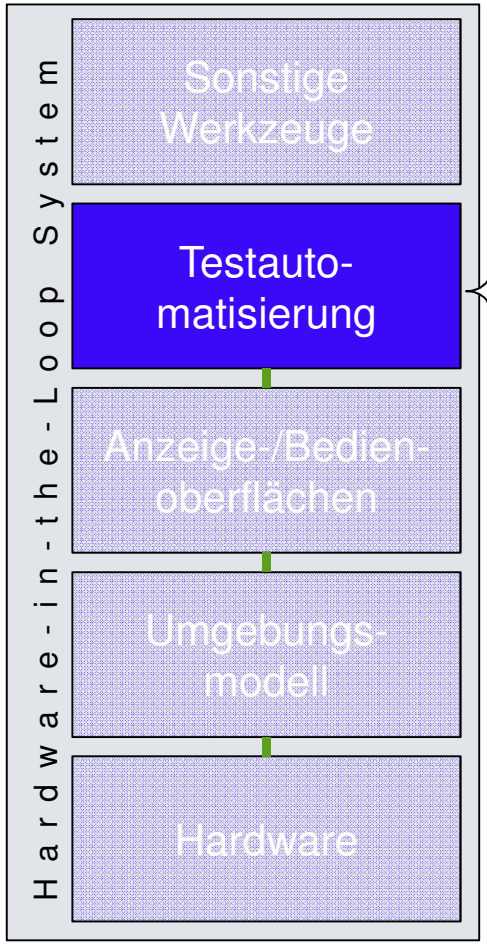
Teilkomponenten eines HiL-Systems: Anzeige- und Bedienoberflächen



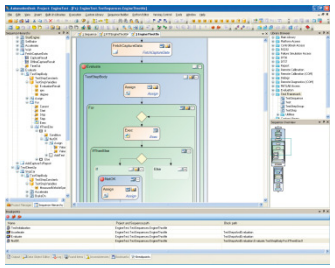
Beispiel: dSPACE-ControlDesk

Automotive Softwareentwicklung

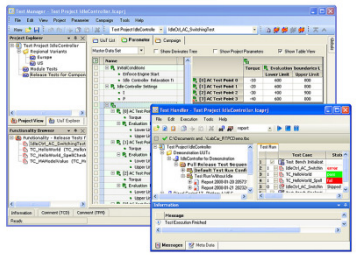
Teilkomponenten eines HiL-Systems: Testautomatisierung



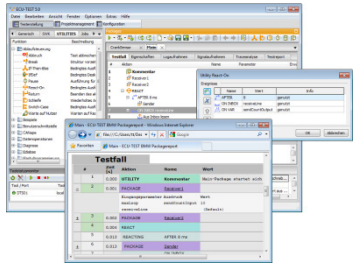
- **Komplette Testfall- und Testergebnisbeschreibung**
- **Vollautomatische Testdurchführung und -auswertung**
- **Erstellung einer vollständigen Testdokumentation**
- **Schnittstellen zur Messtechnik und zu Diagnose-, Flash- und Kalibrierwerkzeugen**



dSPACE AutomationDesk



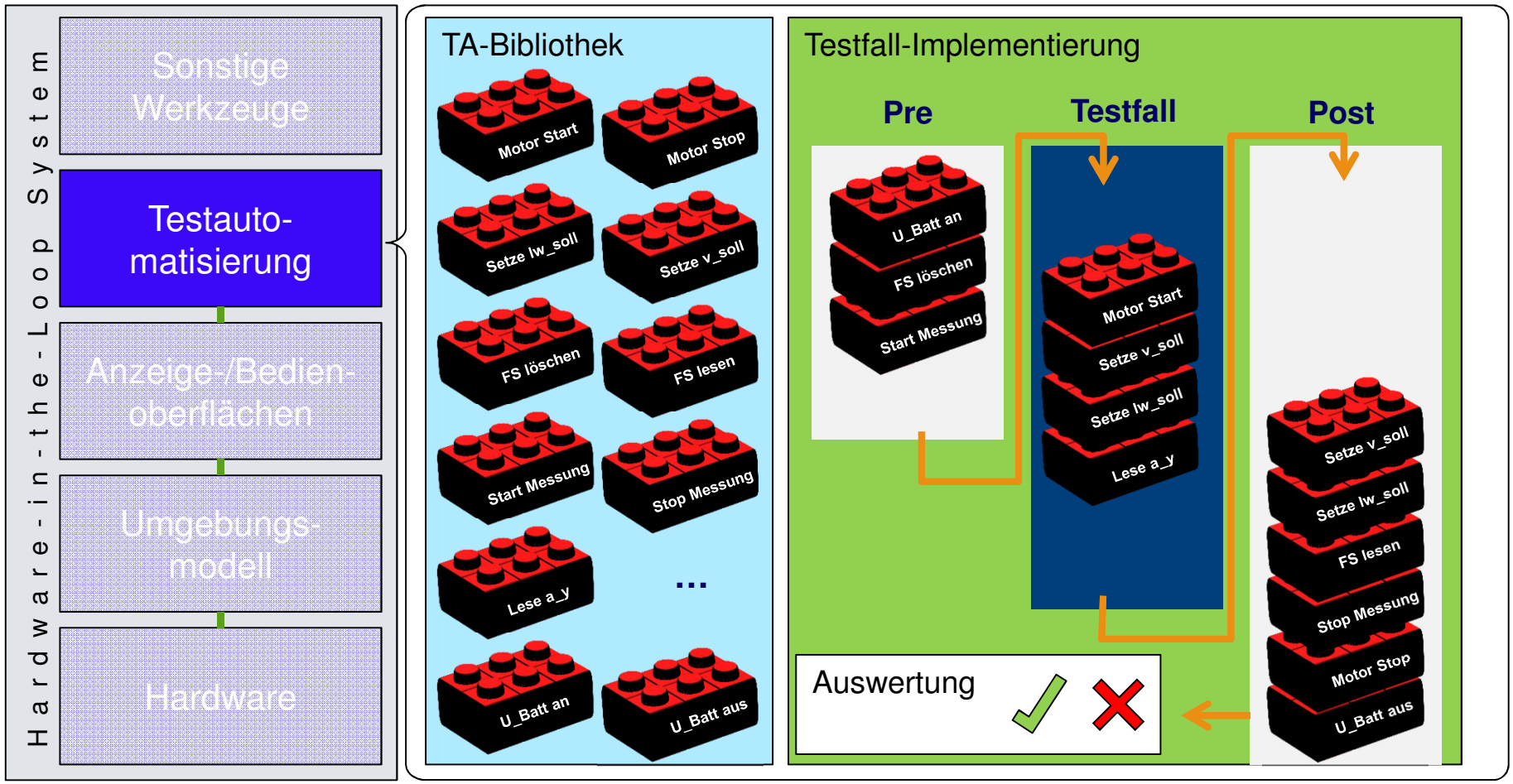
ETAS LABCAR AUTOMATION



TraceTronic ECU-TEST

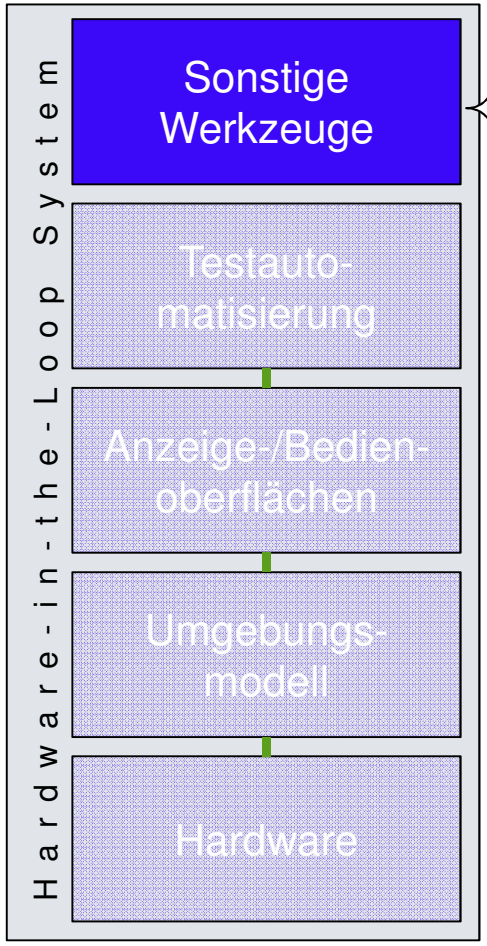
Automotive Softwareentwicklung

Teilkomponenten eines HiL-Systems: Testautomatisierung

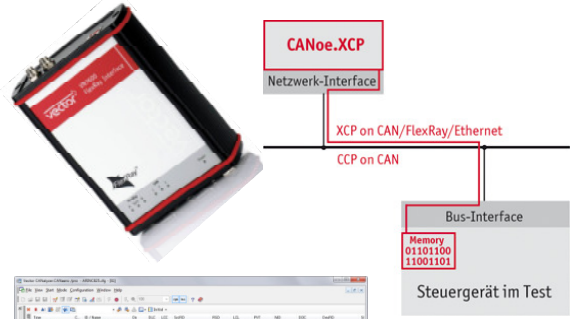


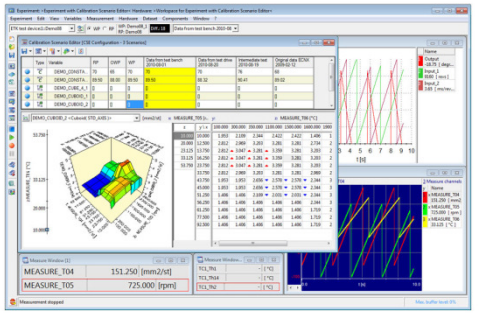
Automotive Softwareentwicklung

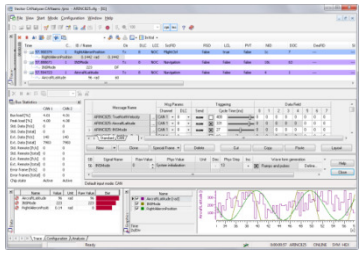
Teilkomponenten eines HiL-Systems: sonstige Werkzeuge



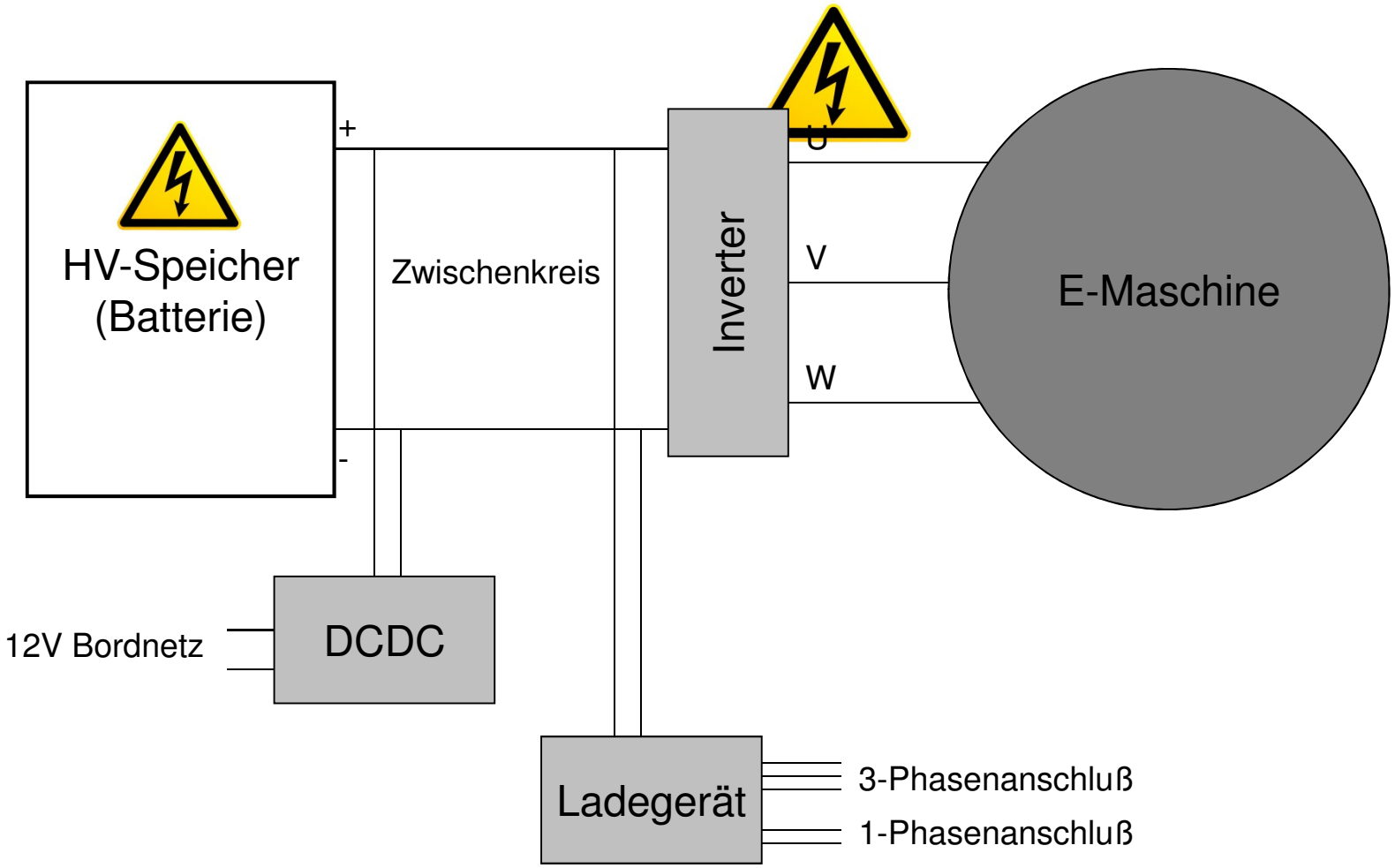
- **Flash- und Diagnose-Werkzeuge**
- **XCP-Werkzeuge (z.B. INCA, CANape)**
- **Busmesstechnik (für CAN, FlexRay, LIN,…)**
- **Testobjekt-spezifische Werkzeuge**
- **Sonstige Messtechnik**







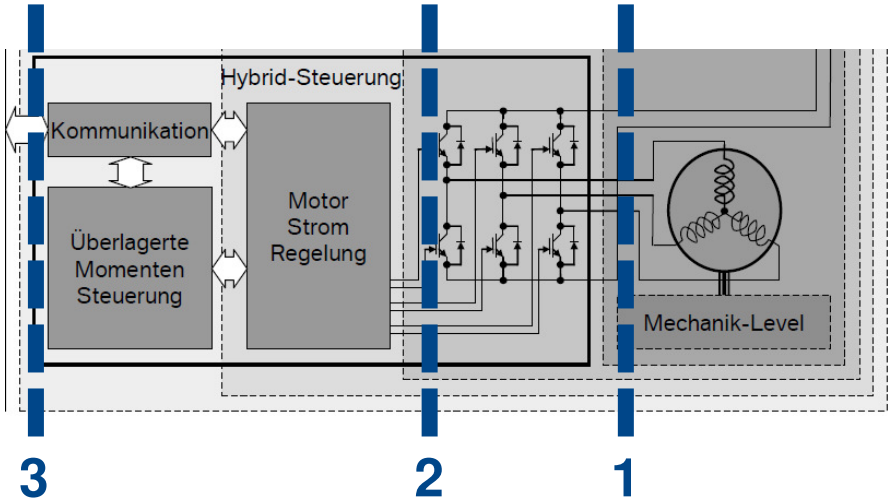
Beispiel: Der Elektroantrieb am HiL



Alternative Test-Ebenen

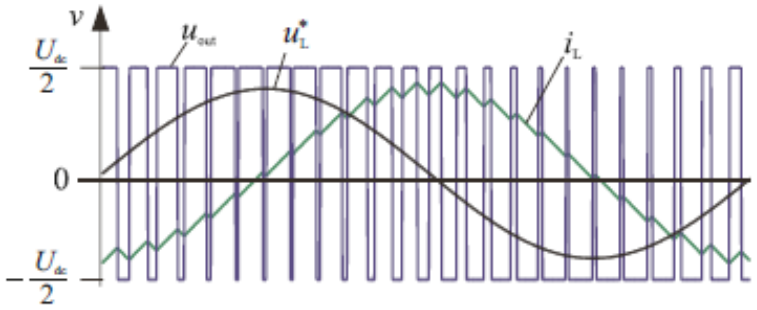
- Verschiedene Testebenen bei Steuergeräten führen zu unterschiedlichen Ausgestaltungen der Prüfstände:

1. Ganzes SG
   - HV-Testplatz (erhöte Sicherheitsanforderung, Kühlung)
     - Echter Motor (auch Prüfling)
     - Motoremulation
2. Nur "halbes" SG
   - HiL: Signalebene
     - Inverter-Modellierung
3. Prüfling ist anderes SG
   - Restbus

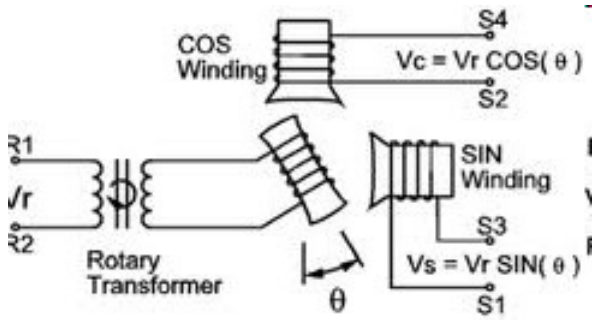


E-Antrieb am HiL: Herausforderungen

- Schnelleres Rechnen nötig
  - hohe Abtastraten
  - Sonst typische Rechenschrittweite: 1ms (1 kHz)
- Alternative Implementierungen:
  - FPGA (Field Programmable Gate Array)
  - Getriggertes Teilmodell, zeitdiskrete Implementierung



PWM-Ansteuerung (100 kHz)



Rotorlagesensor (10kHz)

Fazit

Normgerechte Entwicklung durch modellbasierten Ansatz

- Durchgängige und konsistente Dokumentation von Architektur und Design
- Durchgängige Traceability (bidirektional)
  - Anforderungen / Modell
  - Modell / Code (automatisch)
  - Testfälle / Anforderungen
- Automatische Überprüfung der Modellierungsrichtlinien
- Nachweis Testabdeckung
- Ausschluss von Laufzeitfehlern

Fazit

HiL-Testing: Echtzeitsimulation als Alternative zum Fahrzeug

- Einsparung von kostenintensiven Prototypen
- Keine Gefahr für Mensch und Technik
- Automatisierbarkeit von Testfällen
- Hohe Auslastung möglich (24/7/365)
- Kurze Umrüstzeiten
- Viel größere Anzahl an Testfällen durchführbar (Durchsatz größer)
- Reproduzierbarkeit
- Bessere Möglichkeit der Fehlerstimulation
- Einfacher Zugriff auf Schnittstellen

Ralph Bittner
Ralph.Bittner@itk-engineering.de

www.itk-engineering.de
www.itk-karriere.de

Möglichkeit für Studenten

Studentische Arbeiten, Praktika, Werkstudententätigkeit

- auf Webseite: http://www.itk-karriere.de/bewerben/
- gerne auch Initiativbewerbungen, Kontaktadresse: student@itk-engineering.de

- **Beispiele für studentische Arbeiten:**
  - Signalflussorientiertes Lösen und Implementieren elektrischer Gleichungen im Mehrspannungsbordnetz
  - Inbetriebnahme und Basissoftware-Entwicklung für eine zwei-achsige Nachführeinheit für Flugobjekte und Satelliten
  - Implementierung eines intelligenten Batteriemanagements in ein Modellfahrzeug