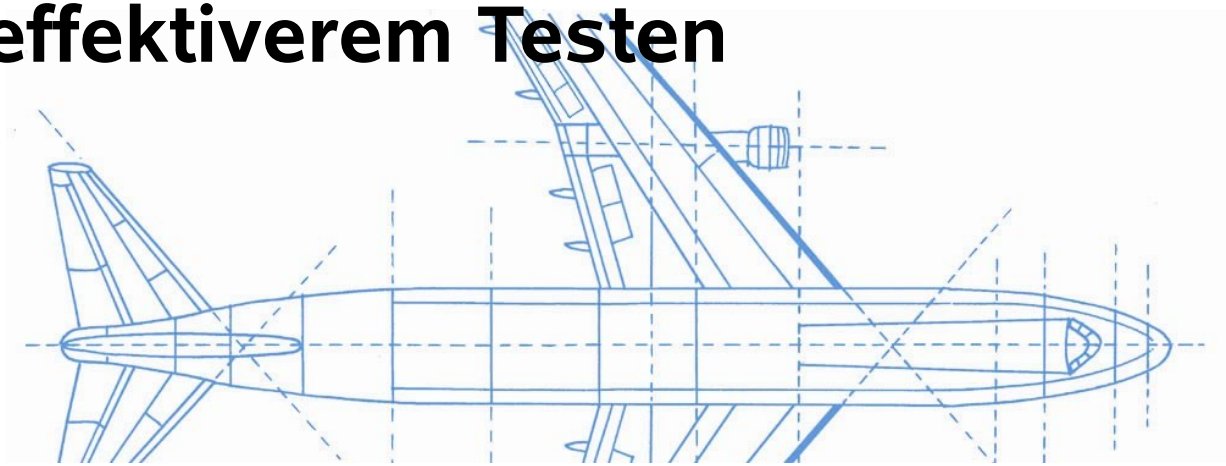
berner & mattner
optimizing your development

Modellierung und Simulation als Schlüssel zu effektiverem Testen



Teststrategien und -techniken für Onboardsysteme in der Luft- und Raumfahrt
7.10.2008 - Garching

Das Unternehmen in Zahlen und Fakten

- Gründungsjahr: 1979
- Geschäftsführer: Hans Berner
- Mitarbeiter: 250

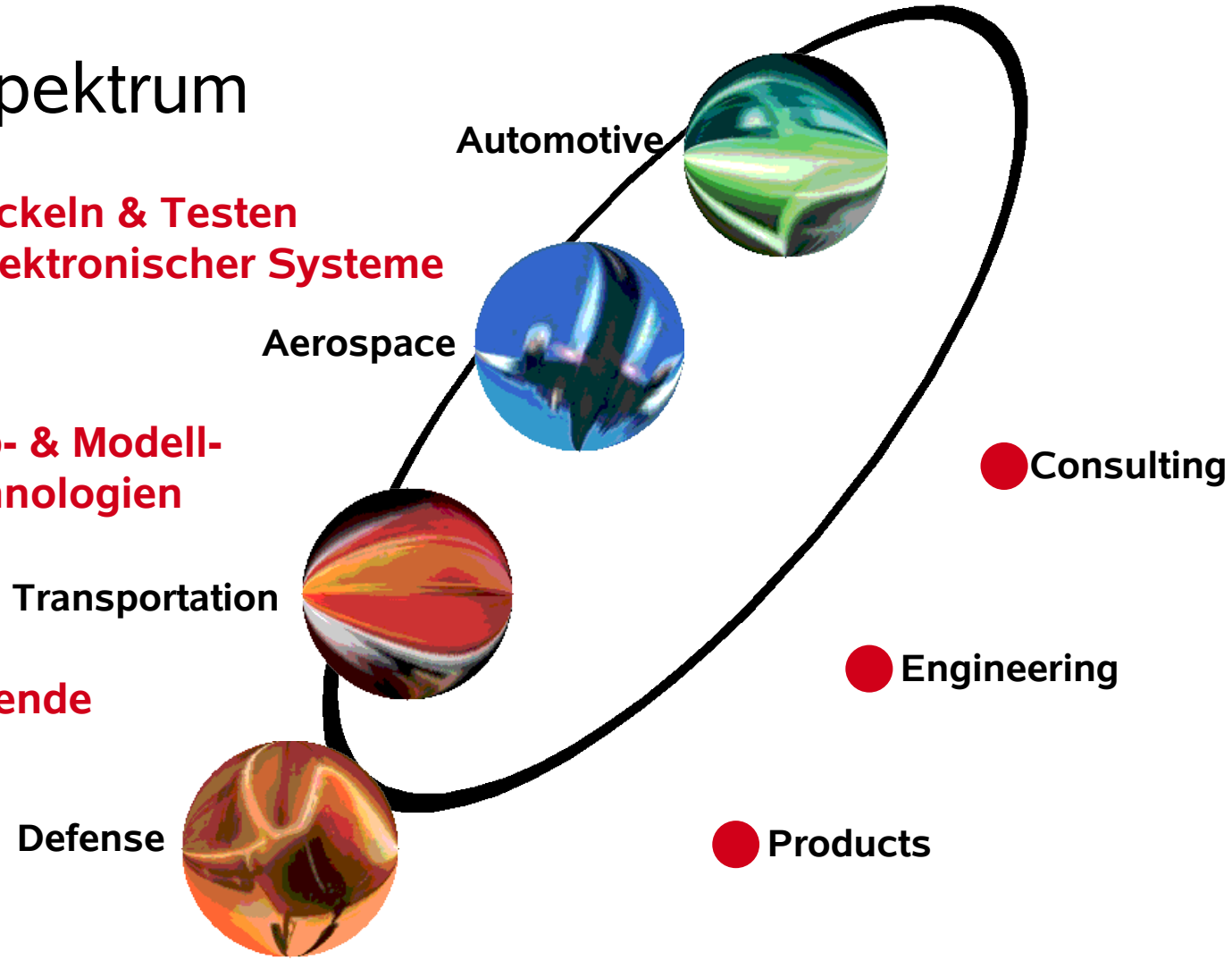
- Standorte:
  - München Hauptsitz
  - Stuttgart Niederlassung
  - Ingolstadt Niederlassung
  - Berlin Niederlassung
  - Wolfsburg Niederlassung

Leistungsspektrum

Entwickeln & Testen komplexer elektronischer Systeme

Innovative Web- & Modell-basierte Technologien

Branchenübergreifende Kompetenz



Worum es heute geht

Die Beantwortung dieser Fragen:

- Wo beginnt der Test?
- Wie erstelle ich bessere Spezifikationen?
  - Wie können Modelle dabei helfen?
  - Kann man Spezifikationen „testen“?
- Wie testet man Komponenten effektiver?
  - Reicht ein isolierter Modultest aus?
  - Welche Fragen treten bei der Integration auf?
- Welche Vorgehen wählen andere Industrien beim Test?
  - Beispiel aus der Energie-Automatisierungstechnik

Was ist ein Modell?

„Eine **Nachbildung** (Darstellung, Wiedergabe oder Reproduktion) eines Gegenstands, **bei dem die** für **wesentlich** erachteten **Eigenschaften hervorgehoben** werden. Die als nebensächlich angesehenen Aspekte lässt man außer Acht.

Ein Modell ist in diesem Sinn also ein **vereinfachtes Abbild der Wirklichkeit**. Was wichtig ist, hängt vom Betrachter (Benutzer) des Modells ab. Von Bedeutung ist dabei, was man mit einem Modell erreichen will bzw. wozu es dient.“

(von http://www.schule-bw.de/unterricht/faecher/chemie/medik/modell/mod1.html)

Was ist ein Modell?

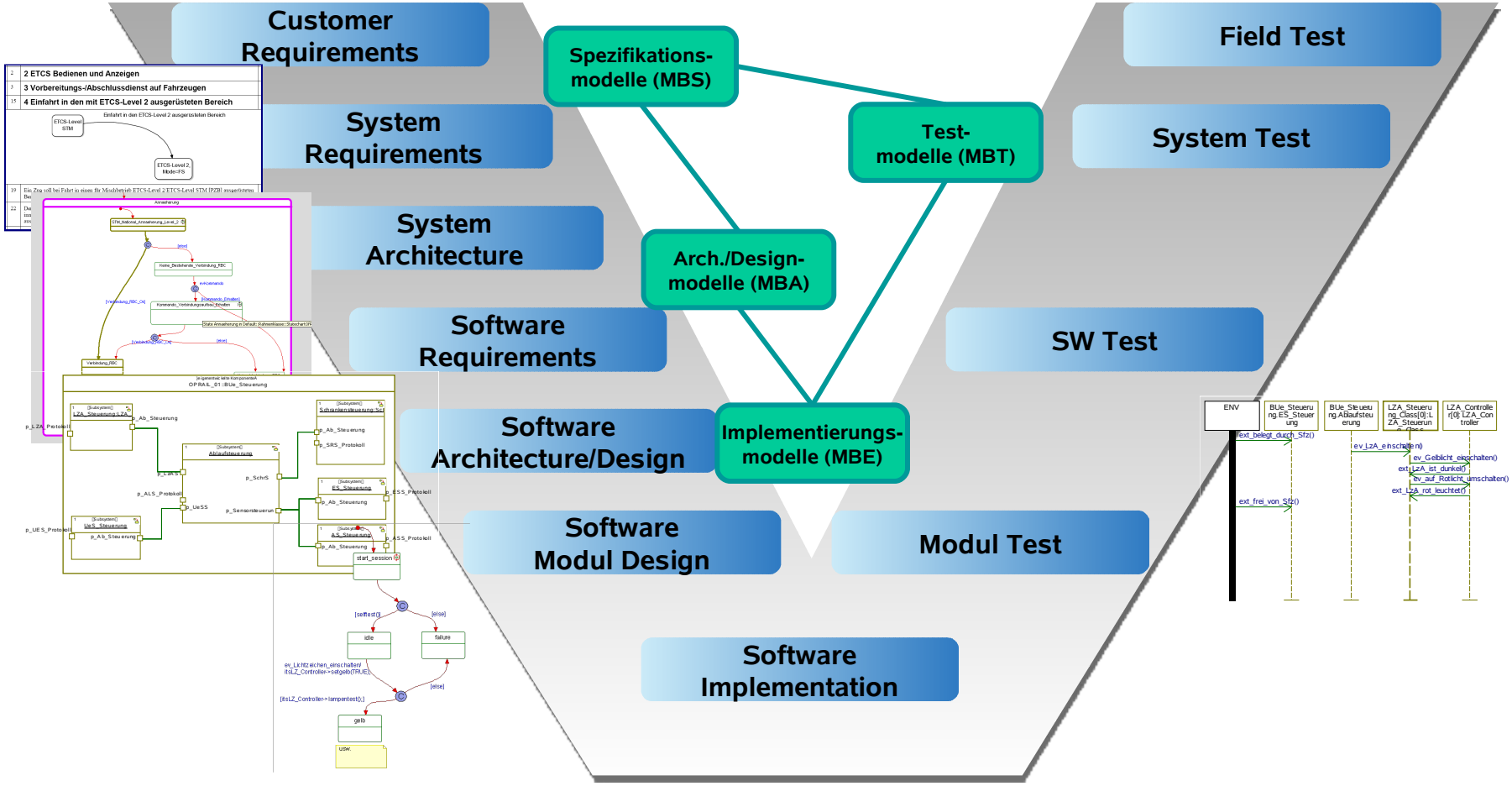
„Ein **Modell** hält eine **bestimmte Sicht** auf das modellierte **System** fest.

Es umfasst **alle Modellelemente**, die dazu **nötig** sind, das heißt alle Modellelemente **für die Beschreibung** der Struktur und **des Verhaltens** des Systems, soweit der Zweck des Modells dies erfordert.

Oft müssen für die Modellierung eines Systems nicht nur Modellelemente innerhalb sondern auch **Elemente außerhalb der Systemgrenze**, zum Beispiel Akteure, berücksichtigt werden. Auch diese Elemente aus der Umgebung des modellierten Systems können zu einem Modell gehören.“

(von Wikipedia.de, http://de.wikipedia.org/wiki/Modell_(UML))

Modellbasierte Methoden im V-Modell



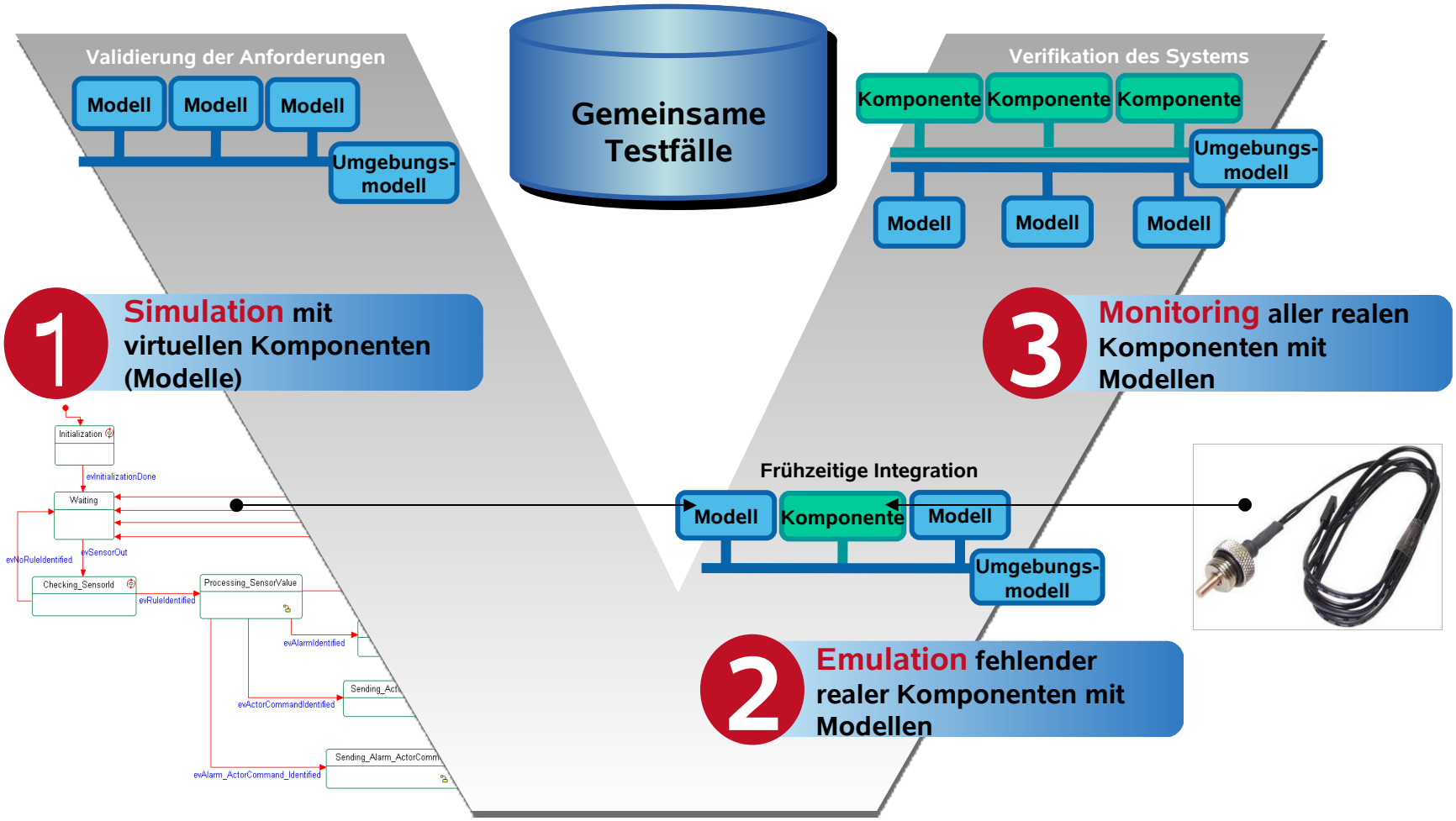
Spezifikationsmodelle

- System wird in Aufbau und Verhalten grafisch abgebildet
  - UML für reine Softwaresysteme
  - SysML für reine Hardware oder gemischte Systeme
- Umgebung wird ebenfalls modelliert
  - Verhalten der Umgebung in Bezug aufs System
  - Schnittstellen mit dem System
- Das Modell wird im Laufe der Entwicklung immer weiter verfeinert
  - Sub-Komponenten entstehen
  - Schnittstellen werden genauer spezifiziert
- Spezifikationsdokumente
  - werden aus dem Modell generiert
  - oder sie sind mit dem Modell verbunden/verlinkt

Erste Erkenntnisse

- Wo beginnt der Test?
  - Der Test beginnt mit dem ersten Requirement!
  - Der erste Integrationstest beginnt mit der zweiten spezifizierten Komponente
- Wie erstelle ich bessere Spezifikationen?
  - Wie können Modelle dabei helfen?
    - Modelle können Struktur in die Requirements bringen
    - Widersprüche können leichter gefunden werden
  - Kann man Spezifikationen „testen“?
    - Bei der Umsetzung der Anforderungen an Verhalten und Design in ausführbare Modelle ist dies möglich – Test-Beispiele:
      - Gültigkeit von Auslegungsparametern
      - Verhalten von Controllerkomponenten
    - Optimierungen des simulierten System ist möglich

Modellbasierter Ansatz zur schrittweisen Integration

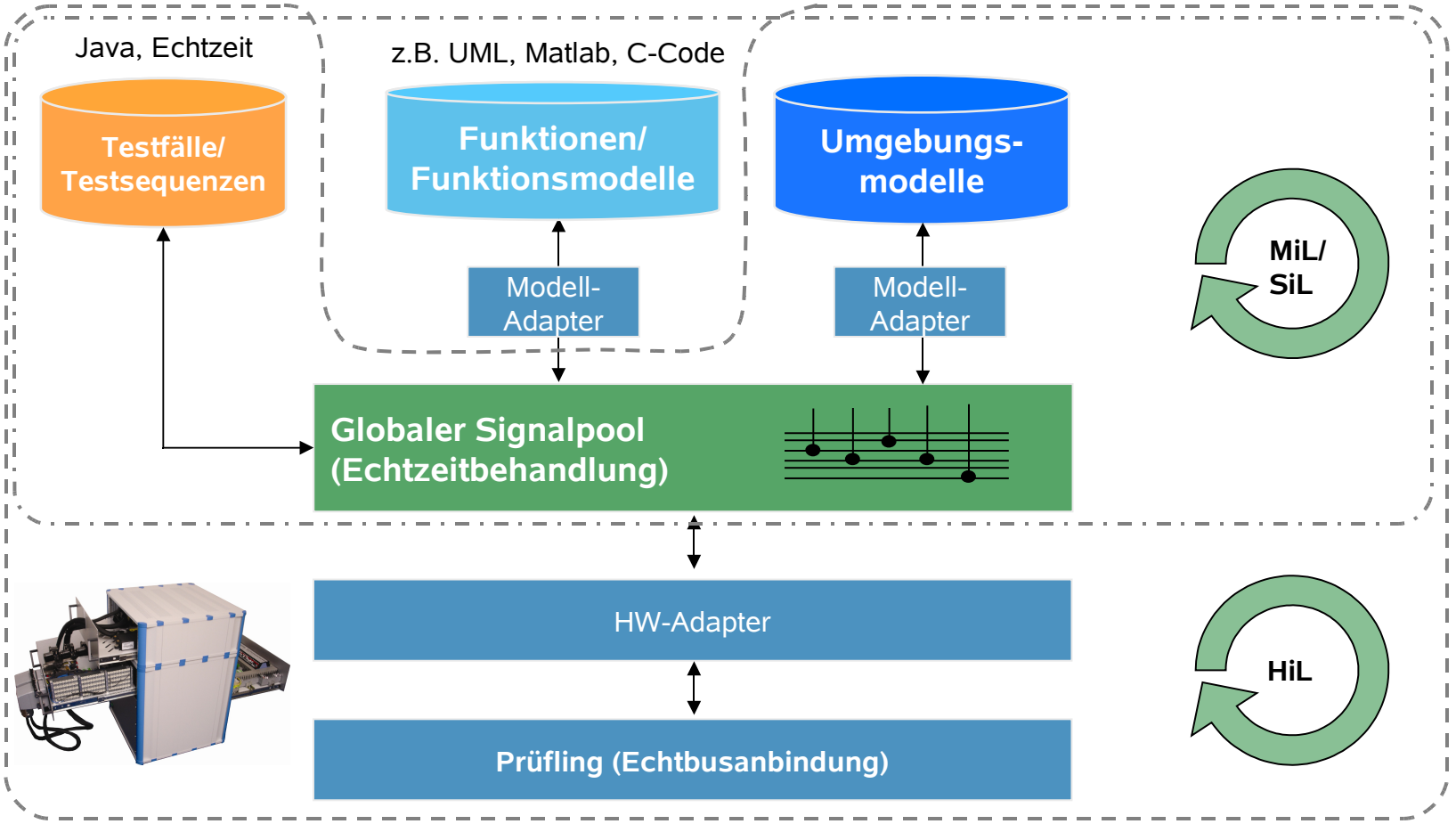


Nutzung dieser Modelle für den Test

- Test der Spezifikationsmodelle durch Simulation
  - Verhalten und Datenflüsse werden simuliert
  - Das korrekte Zusammenspiel der spezifizierten Komponenten kann überprüft werden
- Ableitung von Testfällen
  - Ableitung von Testfällen unter Nutzung der Spezifikationsmodelle als Testgegenstand -> Verfeinerung
  - Festlegung abstrakter Anforderungen treffen (z.B. Timeouts)
- Schrittweise Integration von Lieferkomponenten
  - Ersetzen von Modellkomponenten durch gelieferte Komponenten
  - Nutzung von HiL-Systemen zu HW-Ankopplung
  - Vergleich von Modell- mit Komponentenverhalten

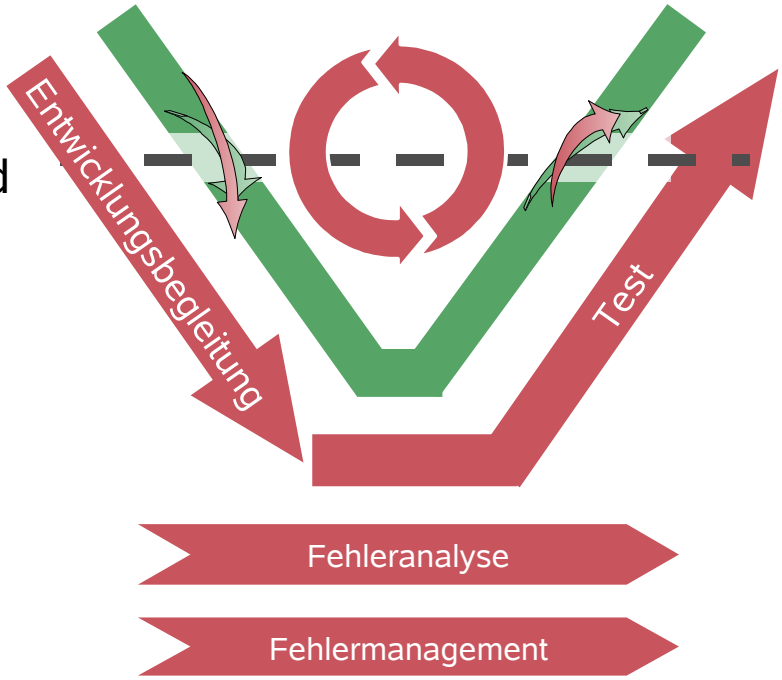
HiL: Hardware-in-the-Loop

Beispiel: modularHiL/MESSINA



Durchgängiger modellbasierter Prozess: Nutzen

- Zuverlässige Absicherung über die OEM / Zulieferer Schnittstelle
- Kostensenkung durch frühe Aufdeckung und Behebung von Fehlern
- Modellbasierte Methoden sind ein Schlüssel für Modularisierung von Systemen (Schnittstellentests an den Systemgrenzen)
- Durchgängige Verwendung der Testfälle; bessere Wiederverwendbarkeit und Anpassbarkeit gegenüber skriptbasierten Testverfahren



Beispiele aus der Praxis (1)

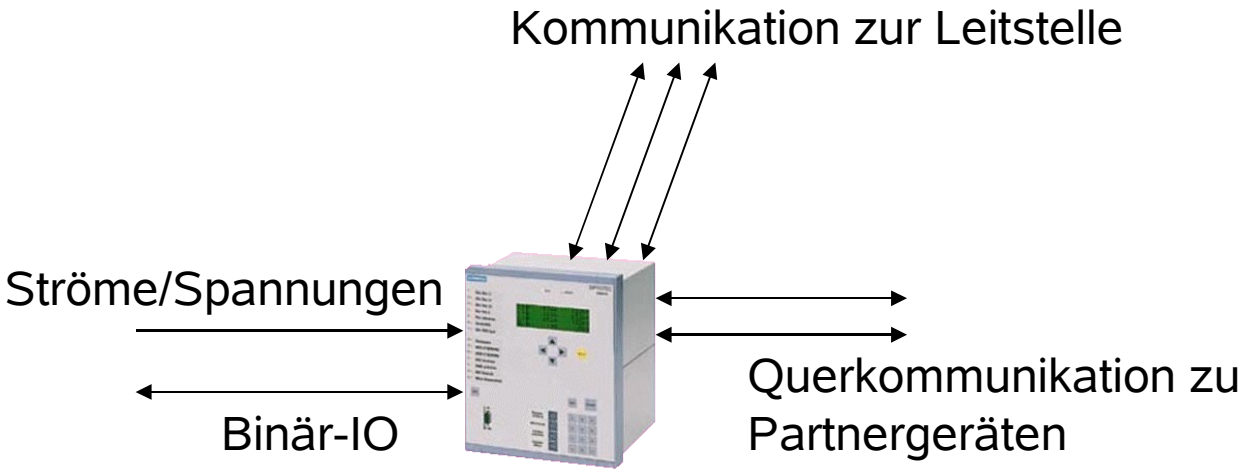
- Anwendung im Bereich Automotive bei MOST- und CAN-Bus:
  - Modellierung des Systemverhaltens und der Testfälle
  - Simulation der Modelle ohne Hardware-Anbindung
  - Bad-Case-Testing und Restbussimulation als aktiver Teilnehmer am Bus
  - Integrationstest aller Komponenten als passiver Teilnehmer am Bus
- Anwendung im Bereich Aerospace:
  - Modellierung eines Brennstoffzellensystems in Matlab und SysML
  - Erstellung von Testfällen
- Anwendung im Bereich Train Control
  - Bisher skriptbasierte Lösung: Testaufwand für Anpassung von Testfällen ca. 6 Wochen für jedes neue Release
  - Verringerung des Einarbeitungsaufwands (häufig wechselndes Testpersonal), Erhöhung der Verständlichkeit
  - Aufwand für Anpassung der Testfälle (modellbasiert) 2 Wochen / Release

Beispiele aus der Praxis (2)

Bei der Entwicklung von Automatisierungstechnik aus dem Bereich der Energieversorgung werden „host-based“ Simulationstechniken eingesetzt, um

- Entwicklungsphasen überlappen zu können
  - Gleichzeitige Entwicklung von FW und HW
- Besondere Testszenarien zu ermöglichen
  - Komplexe Fehlerzustände
  - Multiple Fehlerzustände
- Entwicklern bessere Möglichkeiten zu geben
  - „echtes“ Debugging in verteilten Systemen
- Automatisierte Testabläufe
  - Ohne HW zu entwickeln
  - Ohne HW als „Software-in-the-Loop“ Test auszuführen
- Systemintegrationstests schrittweise durchführen zu können

Schutzprüfungen - Randbedingungen



Das Umfeld der Geräte stellt sich recht komplex dar:

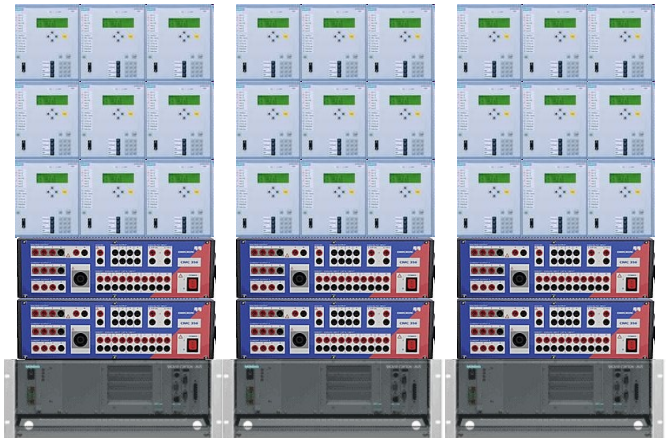
- Analoge Größen
- Binäre Größen
- Umfangreiche Kommunikation mit vielen verschiedensten Protokollen

Testaufbau für komplexe Schutzprüfungen

27 Schutzgeräte
(kleines Umspannwerk)

Signalgeneratoren

Stationsleittechnik



Tests auf einem „echten“ Teststand erfordern hohen Aufwand:

- ▶ FW-Update
- ▶ Engineering
- ▶ Verdrahtung
- ▶ SW-Konfiguration

Nutzung von „host-based“ HW-Simulation für komplexe Schutzprüfungen

3 Simulationsrechner
simulieren 27 Schutzgeräte
und das elektrische Netzwerk

Signalgeneratoren entfallen

„echte“ Stationsleittechnik

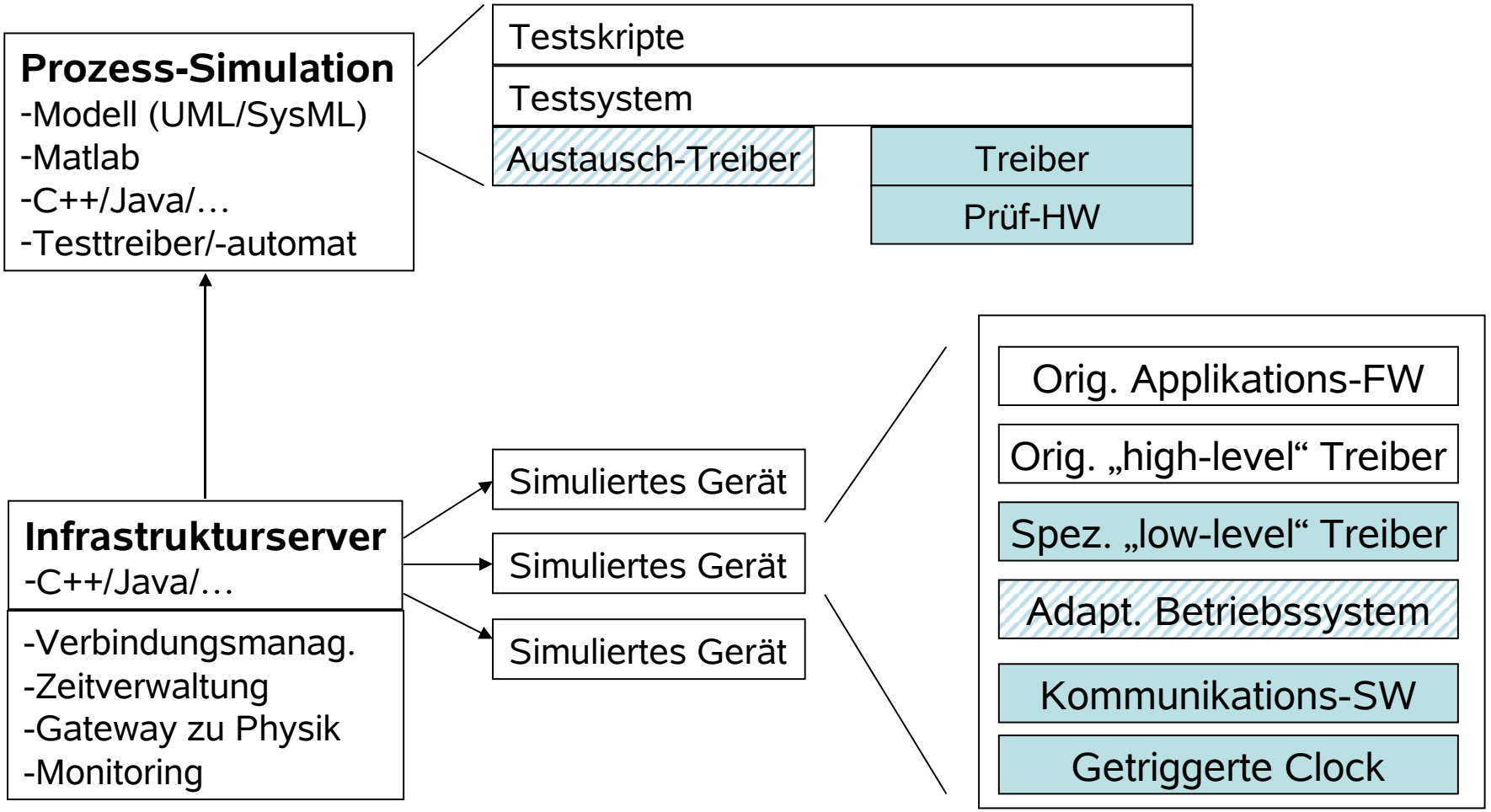
Die Nutzung der Simulation:

- Spart Investitionskosten + Raumbedarf
- Kann schneller in Betrieb genommen werden
- Änderungen im Aufbau sind sehr schnell möglich
  - FW-Update
  - HW-Änderungen

Schritte zum Einsatz von HW-Simulation

- Zu testendes System mußte komplett auf dem PC lauffähig gemacht werden
  - Adaption des Betriebsystems
    - Kann z.B. ein RTOS-Simulator sein (z.B. VxSim (VxWorks), ISIM (Integrity), …)
    - Generische Anpassung ist ebenfalls schon durchgeführt worden
      - Bietet bessere Möglichkeiten für Fehlerinjektion
  - Infrastrukturserver mußte erstellt werden (Steuerung, Zeitbasis, Datendistribution)
  - Compiler-“Spezialitäten“ mussten eliminiert werden
    - Sourcecode muss für PC und Embedded Compiler verarbeitbar sein
  - Empfehlung: Produktivcode nicht mit „simulationsspezifischen“ Dingen anreichern
    - Spezifische Simulationkomponenten auf Dateiebene trennen
    - Verhindert Probleme bei der Zertifizierung
- Umgebung wurde integriert
  - Testumgebung wurde auf der Treiberebene adaptiert
  - Gateway zu den extern angeschlossenen Systemen wurde erstellt

Basisarchitektur für „Host-based“ HW-Simulation komplexer Systeme



Erzielte Vorteile

- Frühe Entwicklung von automatisierten Tests
  - Testsystem wurde auf der Treiberebene auf die Simulation adaptiert
  - Alle Tests sind unverändert auf Simulation lauffähig
  - Algorithmisches Verhalten ist identisch
- Intensive Tests von Fehlerszenarien sind möglich
  - Simulierte Hardware kann beliebige Fehler generieren
  - Timing-Probleme können analysiert werden (z.B. Quarzfehler)
- Test von verteilten Applikationen deutlich verbessert
  - Vollständig transparenter Datenverkehr
  - Gesamte verteilte Applikation „pausierbar“
  - Beliebige Kommunikations-Fehler sind einfach zu erzeugen
  - Integration mit „echter“ Hardware möglich

Zusammenfassung

- Komplexität der Systeme mit herkömmlichen Methoden kaum beherrschbar
- Modulare Systeme erfordern erhöhten Aufwand für Schnittstellentests und Integrationstests
  - Qualitativ hochwertige Spezifikationen
  - Hoher Automationsgrad bei Schnittstellentests ist wünschenswert
  - Umfangreiche Tests von Fehlerszenarien sind notwendig
- Modellbasierte Spezifikations- und testmethoden mit UML/SysML liefern hierzu einen entscheidenden Beitrag.
- Aufwand für Anpassung modellbasierter Tests gegenüber skriptbasierten Tests reduziert sich (z.B. 2 Wochen statt 6 Wochen)

Einige Links zum Thema Simulation

DLR-Papier zum Thema Simulation:

http://www.silest.de/data/multimedia/Mai04d_Paper.pdf

ESA-Seite zum Thema „Modellierung und Simulation“

http://www.esa.int/TEC/Modelling_and_simulation/

OpenSource Simulation Framework der ESA (in Entwicklung):

http://www.esa.int/TEC/Modelling_and_simulation/TECQ6CNWTPE_0.html

http://www.pnp-software.com/eodisp/home.html

Hintergrund zu Simulationsarchitektur HLA (High-Level Architecture)

http://de.wikipedia.org/wiki/High_Level_Architecture