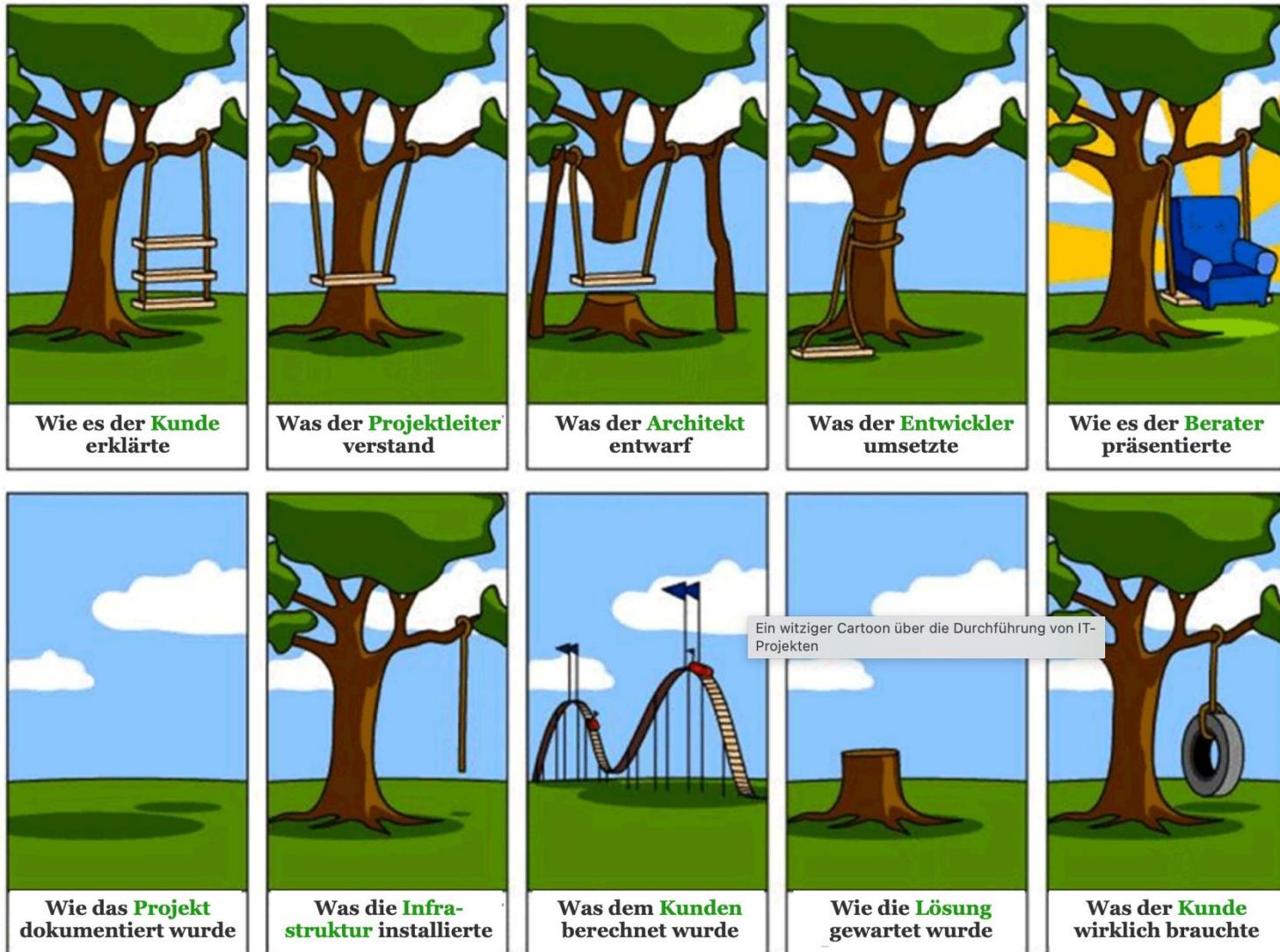


Missverständnisse im Entwicklungsprozess



Warum Modellbildung und Simulation?



- Ziele:
- Systemverständnis verbessern
 - Qualität erhöhen
 - Schwachstellen vermeiden
 - Risiken erkennen und bewerten
 - Kundennutzen optimieren
 - Produktentwicklung beschleunigen
 - Kosten reduzieren

Was ist ein Modell?

Modell

Ein Modell ist ein (vereinfachendes) Abbild einer (partiellen) Realität

- ▶ z.B. Modellbau, physikalische Experimente etc.
- ▶ in der Technik meist eine abstrakte und formale Beschreibung, typischerweise mit dem Methodenapparat der Mathematik

Modell (Quelle: dtv Brockhaus, 1998)

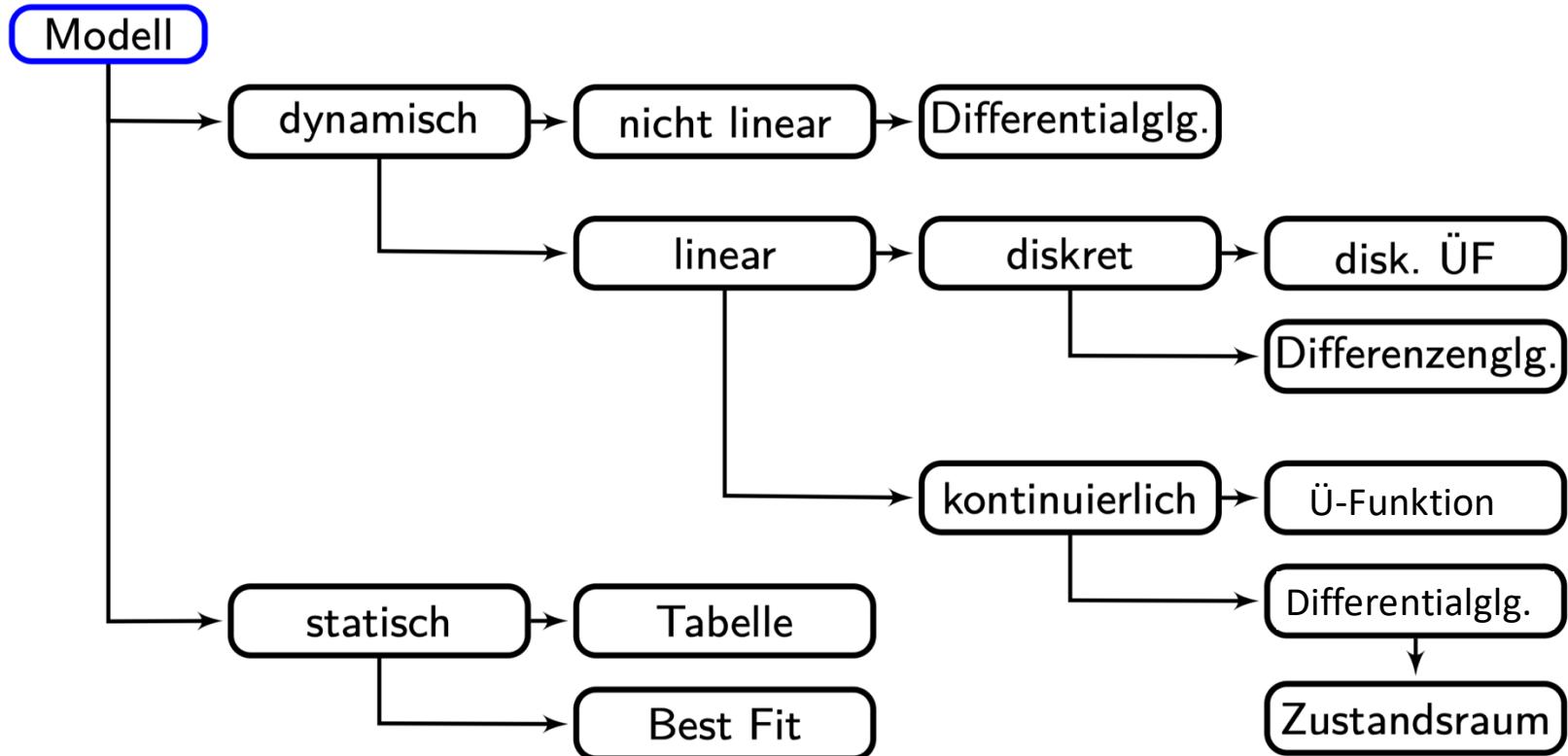
Modell (in Wissenschaft und Technik): „Darstellung, die nur die als wichtig angesehenen Eigenschaften des Vorbilds ausdrückt, um durch diese Vereinfachung zu einem überschaubaren, mathematisch berechenbaren oder für experimentelle Untersuchungen geeigneten Modell zu kommen.“

Modellbildung und Mathematik

Die Naturwissenschaften bedienen sich seit Jahrhunderten **mathematischer Modelle**. Kant formuliert es im Jahr 1796 wie folgt:

...dass in jeder besonderen Naturlehre nur so viel eigentliche Wissenschaft angetroffen werden könne, als darin Mathematik anzutreffen ist.

Mathematische Methoden der Modellbildung



Vorgehen bei der Modellbildung

- ▶ **Abgrenzung:** Nichtberücksichtigung irrelevanter Objekte, Funktionalitäten und Parameter
- ▶ **Reduktion:** Weglassen von Objektdetails (Modelltiefe)
- ▶ **Dekomposition:** Zerlegung, Auflösung in einzelne Segmente
- ▶ **Aggregation:** Vereinigung, Zusammenfassen von Segmenten zu einem Ganzen
- ▶ **Abstraktion:** Begriffs- bzw. Klassenbildung, Funktionsbildung

Verwendung von Modellen

- ▶ **Analyse des Systemverhaltens:** Auf Basis eines Modells werden Einsichten in das Systemverhalten gewonnen. Sowohl das funktionale Verhalten als auch Systemparameter werden simuliert und analysiert.
- ▶ **Vorhersage des Systemverhaltens:** Die Dynamik des Systems wird numerisch simuliert. Eine Simulation kann bereits in der Entwurfsphase durchgeführt werden und ist oft kostengünstiger als das reale Experimente bzw. Prototyp.
- ▶ **Systementwurf:** Das Modell ist Grundlage des Entwurfs eines Eingebetteten Systems. Hier werden die Methoden der Systemtheorie angewendet.
- ▶ **Diagnose und Monitoring:** Ein Modell kann genutzt werden, um im Betrieb des Systems Fehler zu erkennen und ggf. eine Fehlermeldung auszulösen. So könnte beispielsweise der Strom eines elektrischen Antriebs überwacht werden.
- ▶ **Optimierung:** Auf Basis des Modells können z.B. Parameter des Systems optimiert werden, um etwa einen ressourceneffizienten Betrieb zu ermöglichen

Modell-basierter Entwurf

Model-Based Design

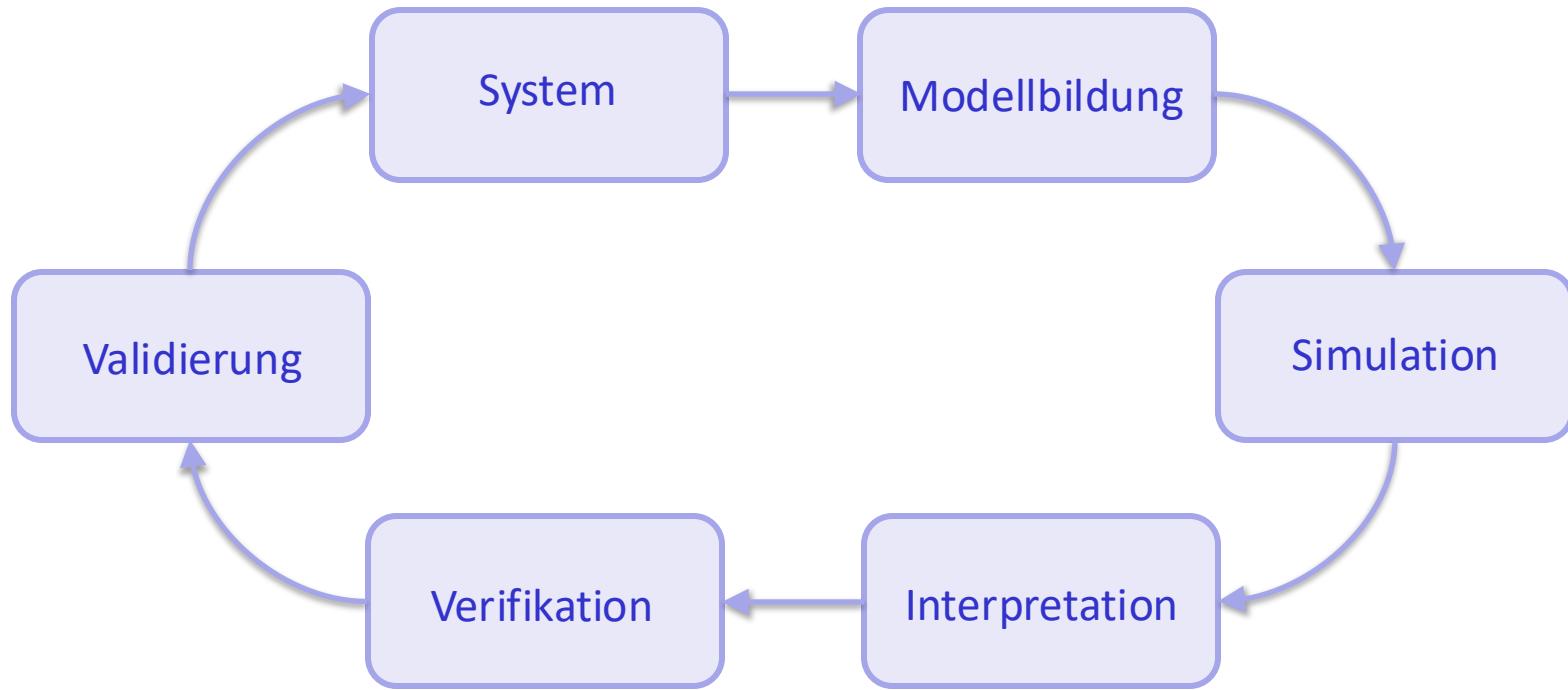
„Model-Based Design (MBD) is a **mathematical** and **visual method** of addressing problems associated with designing complex control, signal processing and communication systems. It is used in many motion control, industrial equipment, aerospace, and automotive applications. Model-based design is a methodology applied in designing embedded software/hardware systems.“

Quelle: https://en.wikipedia.org/wiki/Model-based_design

MBD ist ein Ansatz zur Entwicklung und Simulation technischer HW- und SW-Systeme unter Verwendung von Modellen als zentrale Entwurfswerkzeuge.

Es ermöglicht eine effiziente Entwicklung, indem es den gesamten Entwurfsprozess von der Anforderung bis zur Implementierung modellbasiert gestaltet und somit für schnellere, kostengünstigere und risikoreduzierte Entwicklungsprozesse sorgt.

Modellierungskreislauf bei MBD



Simulation

Simulation

Die Simulation ist ein virtuelles und i.A. rechnergestütztes Experiment am Modell und das eigentliche Ziel der Modellierung bzw. Modellbildung.

Anmerkung: Simulation oder Simulator, abgeleitet von dem lateinischen *simulatio* („Schein, Verstellung, Täuschung“)

Eine Simulation ist die Nachbildung eines Prozesses oder eines Systems, um diese virtuell zu analysieren und zu optimieren. Sie dient dazu, ein reales Szenario nachzubilden, ohne die tatsächlichen Risiken, Kosten oder Komplexitäten berücksichtigen zu müssen. Simulationen ermöglichen es, verschiedene Szenarien zu testen, Verhaltensweisen zu analysieren und potenzielle Probleme frühzeitig zu erkennen und zu beheben.

Simulation vs. Emulation

- ▶ „A **simulation** is a system that behaves similar to something else, but is implemented in an entirely different way. It provides the basic behaviour of a system, but may not necessarily adhere to all of the rules of the system being simulated. It is there to give you an idea about how something works.“
- ▶ „An **emulation** is a system that behaves exactly like something else, and adheres to all of the rules of the system being emulated. It is effectively a complete replication of another system, right down to being binary compatible with the emulated system's inputs and outputs, but operating in a different environment to the environment of the original emulated system. The rules are fixed, and cannot be changed, or the system fails.“

Eine **Emulation** ahmt ein bekanntes System in einer neuen Umgebung nach, während eine **Simulation** das Verhalten eines Systems basierend auf einem Modell nachbildet, um das System zu analysieren und detailliert zu verstehen

Einsatzgebiete von Simulationen

- **Physik:** Astrophysik, Geophysik
- **Elektrotechnik:** Analoge und digitale Signalverarbeitungssysteme
- **Biologie:** Bioinformatik, Bioverfahrenstechnik
- **Materialwissenschaften:** Smart Materials, Nanostrukturen
- **Computergraphik:** Lokale und globale Beleuchtungsmodelle
- **Automobilindustrie:** Crashtests, Windkanal, Airbags, Fahrdynamik, Schallabstrahlung
- **Halbleiterindustrie:** Bauelementsimulation, Prozesssimulation, Schaltkreissimulation, Chip-Layout
- **Finanzwirtschaft:** Kursprognosen, Option Pricing

Verifikation

Verifikation

„Bestätigung durch einen objektiven Nachweis, dass Anforderungen erfüllt werden“

Quelle: ISO9000

Verifizierung oder Verifikation aus dem lat. veritas „Wahrheit“ und facere „machen“

Verifikation von Software

Varianten

- ▶ **Review** - Die Entwurfsdokumente und der Quellcode des Programms werden kritisch bewertet, ohne das Programm selbst laufen zu lassen (vier-Augen-Prinzip).
- ▶ **Test** - Das Programm wird mit Eingabedaten betrieben und das Verhalten des Programms wird mit dem erwarteten Verhalten verglichen (*Golden Reference*).
- ▶ **Formale Verifikation** – Ohne Programmausführung wird mit Hilfe formaler Methoden bewiesen, dass der Programmcode korrekt ist, d.h. dass er die Spezifikation erfüllt.

Testebenen

- ▶ **Modultest** - unterste Ebene: einzelne Funktionen bei prozeduralen Programmiersprachen bzw. einzelne Methoden/Klassen bei objektorientierten Programmiersprachen
- ▶ **Integrationstest** - Zwischenebene, testet Zusammenspiel von Komponenten
- ▶ **Systemtest und Abnahmetest** - oberste Ebene: Gesamtprogramm/System

Formale und automatisierte Verifikation

- ▶ **Theorem Proving:** Das Theorem Proving lässt sich auf Basis verschiedener Logiken durchführen. Um Aussagen auf Basis einer Spezifikation beweisen zu können, bedarf es eines Beweissystems, das passend zur verwendeten Logik ist. Das Beweissystem enthält Axiome und Regeln, mittels derer es z. B. möglich ist zu beweisen, dass eine Formel eine Tautologie ist oder dass eine Formel unter einer bestimmten, gegebenen Menge von Annahmen Gültigkeit besitzt. Ein automatischer Theorem Prover versucht durch geschickte Anwendung der Regeln die zu beweisende Aussage so umzuformen und zu vereinfachen, dass ihre Richtigkeit festgestellt werden kann.
- ▶ **Model-Checking:** Das Model-Checking ist eine Technik zur automatisierten Verifikation von Software im finiten Zustandsraum. Wie die theoretische Informatik lehrt, ist die vollautomatische Verifizierung für eine breite Klasse von Programmen ein unlösbares Problem. (Stichwort: Testabdeckung)

(Tautologie: Aussage, die immer wahr ist oder keine zusätzliche Information liefert)

Validierung

Validierung

„Bestätigung durch objektiven Nachweis, dass die Anforderungen für eine bestimmte Anwendung oder einen bestimmten Gebrauch erfüllt sind“

Quelle: ISO9000:2015

Validierung oder Validation: zu validieren von lateinisch validus = „kräftig“, „wirksam“, „fest“

Oder: Prozess des Bestätigens, dass die Spezifikation einer Phase oder des Gesamtsystems passend zu und konsistent mit den Anforderungen des Kunden ist.

Electronic Design Automation

EDA

„Electronic Design Automation (EDA), zu Deutsch Entwurfsautomatisierung elektro-nischer Systeme, bezeichnet rechnergestützte Hilfsmittel für den Entwurf von elektro-nischen Systemen, insbesondere der Mikroelektronik. EDA wird zumeist als Teilgebiet des computer-aided design (CAD) bzw. des computer-aided engineering (CAE) ver-standen. Alternativ wird anstelle von EDA auch von ECAD (electronic CAD) gespro-chen.“

- ▶ Die Konzepte zur Modellbildung, der Simulation, des Tests sowie der Verifikation und Validierung werden von den modernen Tools (Entwurfswerkzeuge) unterstützt.

Ausführbare Spezifikationssprachen

Executable Specification

Ausführbare Spezifikationssprachen (executable specification) zur funktionalen Beschreibung von Systemen und Systemkomponenten.

Ziele:

- ▶ Anforderungsanalyse (requirements analysis)
- ▶ Systemanalyse und Systementwurf (systems design)
- ▶ Validierung und Verifikation (golden reference)

Zum Einsatz kommen dabei Hochsprachen wie:

- ▶ TLA+ (Temporal Logic of Actions)
- ▶ PROMELA (Process/Protocol Meta Language)

Models of Computation

Models of computation (MoCs) define

- ▶ Components and the organization of computations in components: Procedures, processes, functions, and finite state machines are possible components.
- ▶ Communication protocols: These protocols describe methods for communication between components.

Ein MoC (Berechnungsmodell) ist eine Abstraktion in der Informatik, die definiert, wie Computer grundlegende Operationen ausführen, Daten verarbeiten und Probleme lösen. Diese Modelle dienen der theoretischen Analyse von Algorithmen und der Untersuchung der Grenzen dessen, was berechenbar ist. Bekannte Beispiele sind die *Turing Maschine* und das *Lambda-Kalkül*, die verschiedene Aspekte der Berechenbarkeit und Programmierparadigmen wie imperative und funktionale Programmierung beeinflusst haben

Abstraktionsebenen von Programmiersprachen

System-level Models	SysML, Verilog-AMS, SystemC, C++, Python, Matlab/Simulink
Algorithmic level	Python, C++, MATLAB/Simulink, VHDL, Verilog HDL
Instruction set level	Assembler, C/C++
Register-transfer level	VHDL, Verilog HDL, (HLS in C++)
Gate-level models	SPICE, Verilog HDL, VHDL

HDL : Hardware Description Language

HLS : Hardware Level Synthesis

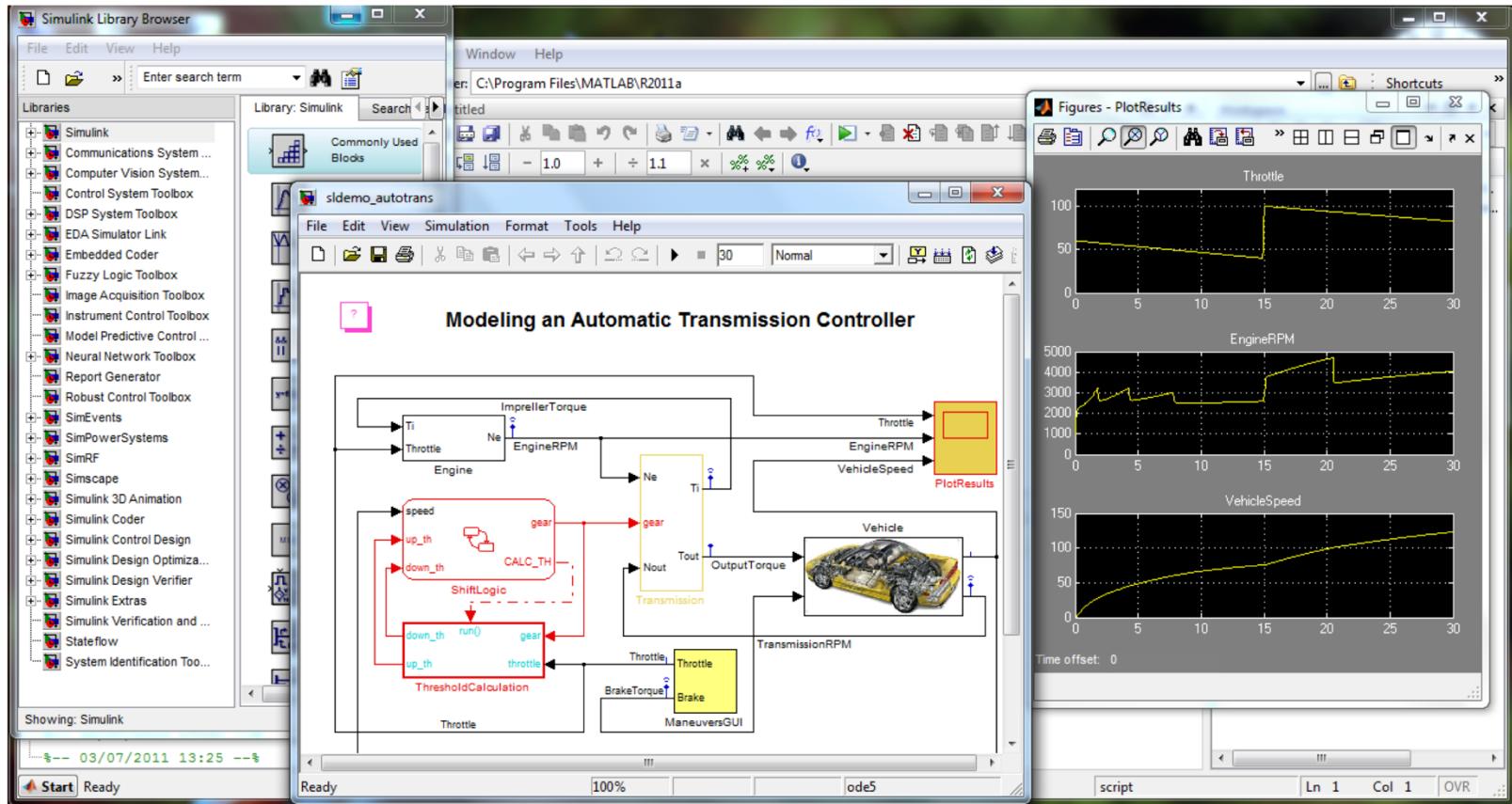
SysML : Systems Modeling Language (UML-basiert)

SystemC : Klassenbibliothek, die für C++ hardware-nahe Komponenten ergänzt

Verilog-AMS : Verilog (Verification and Logic) für Analoge und Mixed Signals

VHDL : Very High Speed Integrated HDL

Symbolische Modellierung - Simulink



Embedded System (Eingebettetes System)

Embedded System (Eingebettetes System)

„Ein (Mikro-) Computersystem, das in ein technisches System eingebettet ist, welches selbst nicht als Computer erscheint.“

Merkmale:

- ▶ feste Funktionalität
- ▶ kontrolliert und verarbeitet physikalische Größen
- ▶ in der Regel sicherheits- und zeitkritisch (z.B. Airbag)
- ▶ eingeschränkte Ressourcen

Cyber-physical system (CPS)

„Cyber-Physical Systems (CPS) are integrations of computation and physical processes.“ Quelle: Lee, E.A.: Computing foundations and practice for cyber-physical systems

Cyber-physical system (CPS)

Embedded System (ES)

Dynamic physical environment

Als CPS bezeichnet man vernetzte, komplexe Systeme, die in der Lage sind, autonom Entscheidungen zu treffen und auf variable Bedingungen zu reagieren

Aspekte von CPS

