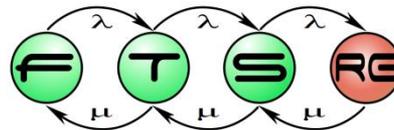


Systemmodellierung

GÖNCZY László
HUSZERL Gábor

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



Die Lehrveranstaltung

- Lehrveranstaltung (VIMIAA00)
 - 3 in 1 (auf ungarisch/deutsch/englisch)
- GÖNCZY László
 - Operative Leitung, Organisatorische Fragen
- HUSZERL Gábor (huszerl@mit.bme.hu)
 - deutschsprachige Vorlesung, ...
- BAJCZI Levente
 - deutschsprachige Übung



Die Deutschsprachige Lehrveranstaltung

- Vorlesungen und Leitung
 - HUSZERL Gábor
- Übungen
 - BAJCZI Levente
- Webseite
 - <https://inf.mit.bme.hu/edu/courses/remo-de>
 - Infos, Nachrichten, Folien, ...
 - <https://q2a.inf.mit.bme.hu/>
 - Über Fragen der Hausaufgabe

Die Lehrveranstaltung

- 14 Vorlesungen
 - Mittwochs 10-12 Uhr, I.L405
 - 6 Übungen (2 fallen aus: 10. April, 1. Mai)
 - Freitags 12-14 Uhr (in ungeraden Wochen), I.L405
 - Zu erst am 28. Februar
 - Tests? (keine Eingangstests!)
 - Anwesenheitspflicht (an mind. 4 aus den 6 Übungen)
- 1 Hausaufgabe (mit mündlicher Verteidigung)
- 1 Klausur (mit Eingangstest)

Bewertung

- Klausur (mit Eingangstest): 70% der Endnote
 - am Freitag, dem 22. Mai, 8-10 Uhr
(Wiederholung: 29. Mai, 8-10 Uhr)
- Hausaufgabe (mit Verteidigung): 30% der Endnote
- Alle zwei Teile mit mindestens 40% der Punkte
 - die Klausur und die Hausaufgabe können einmal nachgeholt werden
- Optionale Zusatzpunkte (nach positiver Bewertung):
 - Übungsvorbereitungen
 - optionale Zusatzaufgaben für zusätzliche Punkte

Hausaufgabe

- Abgabepflicht, Verteidigungspflicht
- Persönliche Aufgabe: Zustandsmodellierung (Yakindu)
- Wichtige Termine
 - Vorbereitungsaufgabe: 3-4. Semesterwoche
 - Ausgabe der Aufgaben: 4. Semesterwoche
 - Abgabe: 10. Semesterwoche
- Elektronische Abgabe
- Mündliche Verteidigung

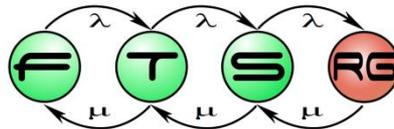
Thematik

- Strukturmodelle
- Verhaltensmodelle
 - Zustandsmodelle, Prozessmodelle
- Entwicklung von Modellen
- Überprüfung von Modellen
- Leistungsmodellierung
- Visuelle Datenanalyse
- Simulation
- Benchmarking, Codegenerierung

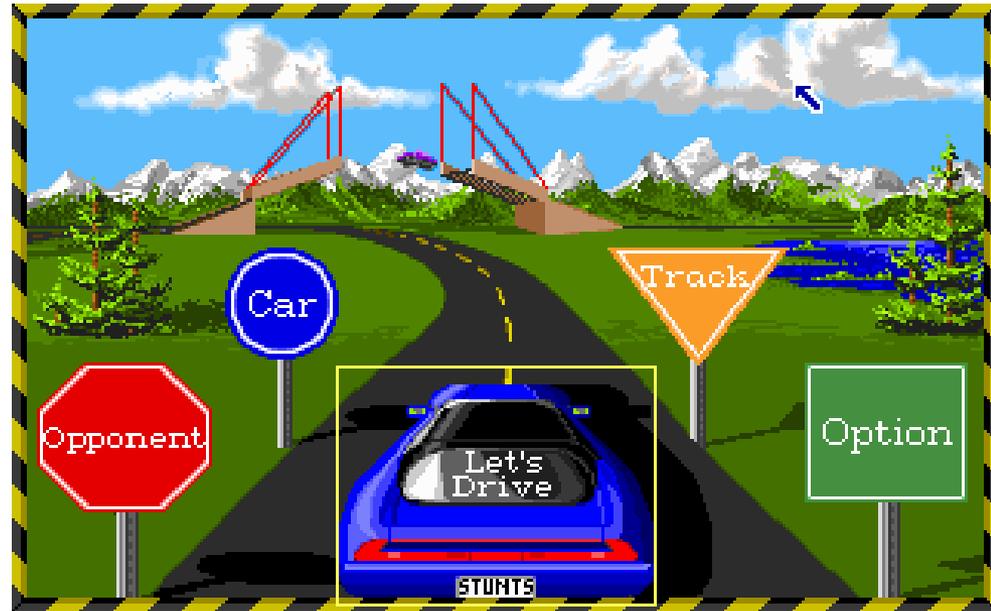
Grundlagen der Modellierung

Dr. PATARICZA András, GÖNCZY László
HUSZERL Gábor

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



“Motivation”



■ Stunts Autorennspiel

- Distinctive Games/Brøderbund Software, 1990
- [https://en.wikipedia.org/wiki/Stunts_\(video_game\)](https://en.wikipedia.org/wiki/Stunts_(video_game))
- <http://www.pcgameshardware.de/Stunts-Spiel-35676/>

■ Rennspiel + Gestaltung eigener Strecken(!)

- Eigentlich ein “Fachgebietspezifisches Modell”

“Motivation”: Wofür ist ein Modell gut?

Kann die Strecke durchgefahren werden?
Wenn ja, mit welchem Auto?
Ist die Strecke eindeutig gestaltet?

Kombinierbare Elemente?
Was für Strecken sind zu gestalten?

Ist die Strecke auch in der anderen
Richtung zu befahren?

Ist eine Strecke „realistisch“?
Ist für Rennspiele geeignet?

Wie können neue Elemente
definiert werden?

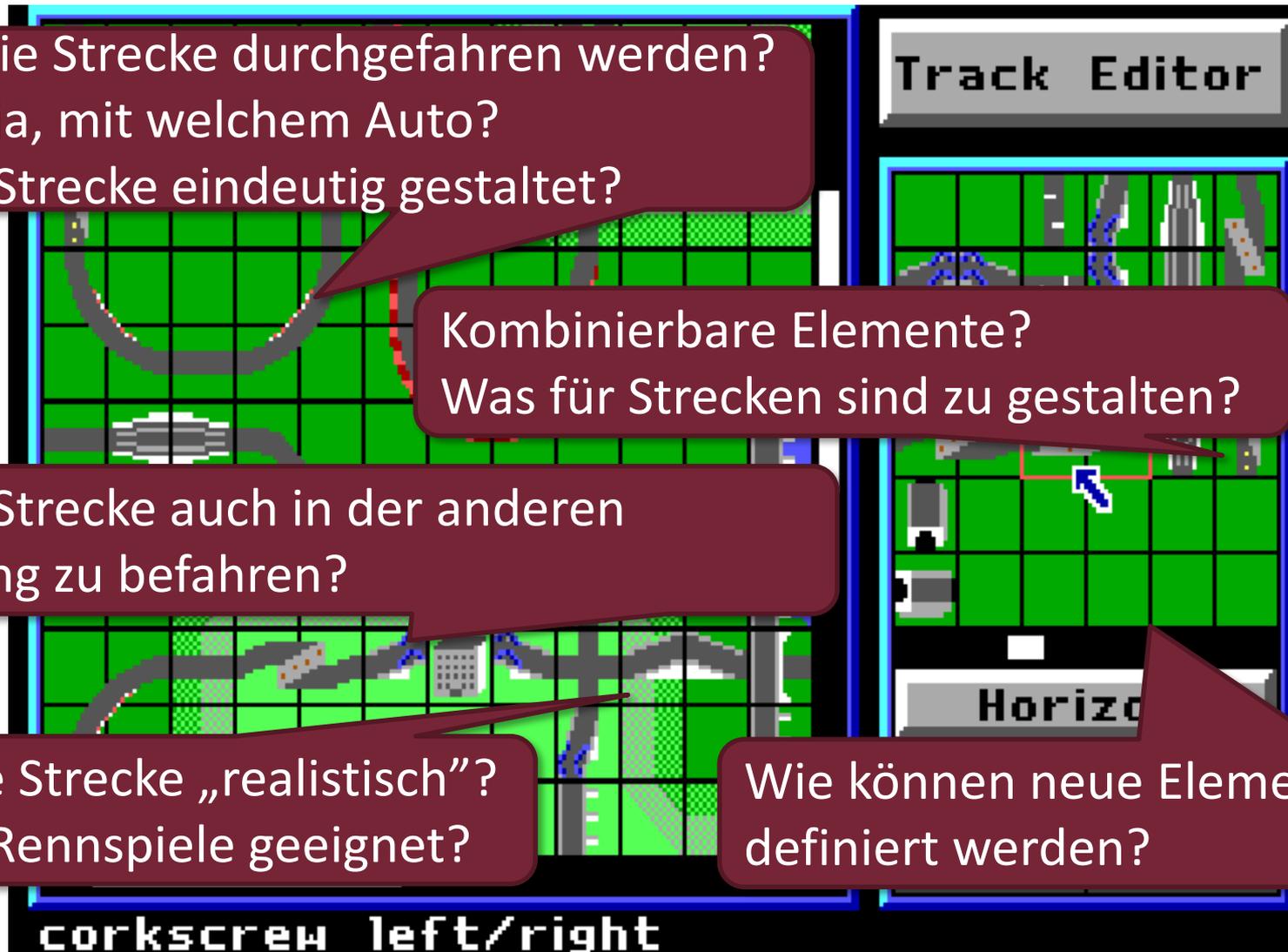


Bild: <http://www.abandonia.com/games/73/Stunts.htm>

Inhalt

Modelle und
Modellierung

Wofür werden Modelle
benutzt?

Grundbegriffe

Illustrative
Beispiele

Inhalt

Modelle und
Modellierung

Wofür werden Modelle
benutzt?

Grundbegriffe

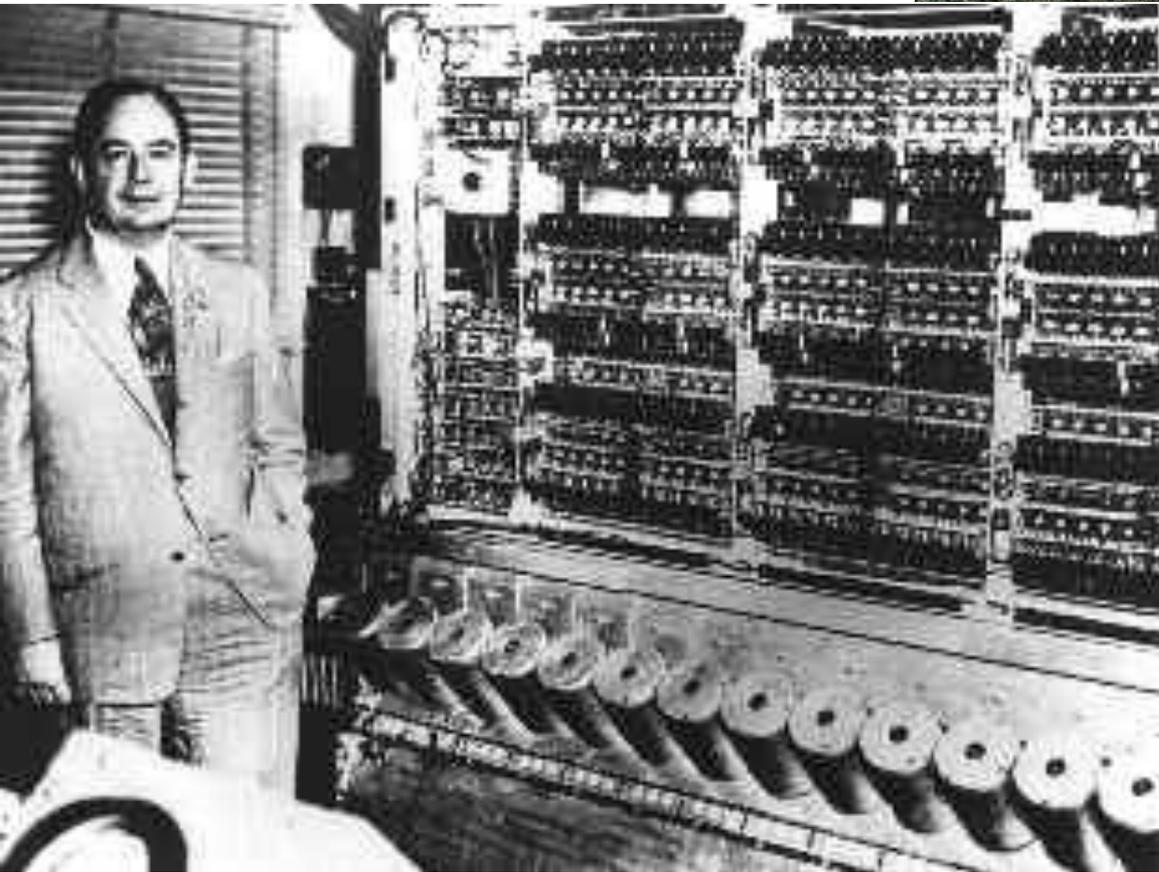
Illustrative
Beispiele

Modelle und Modellierung

Was ist ein Modell?

- "The sciences
 - do not try to explain,
 - they hardly even try to interpret,
 - they mainly make models.
- By a model is meant
 - a mathematical construct which,
 - with the addition of certain verbal interpretations,
 - describes observed phenomena.
- The justification of such a mathematical construct is solely and precisely that it is expected to work.,,

János von Neumann



E HÁZBAN SZÜLETETT
ÉS ÉLT 18 ÉVES KORÁIG
NEUMANN JÁNOS
1903 — 1957

A XX. SZÁZAD EGYIK LEGKIVÁLÓBB
MATEMATIKUSA,
AKI 1951 — 1952 — BEN
AZ AMERIKAI MATEMATIKAI
TÁRSULAT ELNÖKE VOLT.
AZ EMLÉKTÁBLÁT SZÜLETÉSÉNEK
100. ÉVFORDULÓJÁRA
A BOLYAI JÁNOS MATEMATIKAI
TÁRSULAT ÉS
AZ AMERIKAI MATEMATIKAI
TÁRSULAT KÖZÖSEN ÁLLÍTOTTA.



IN THIS HOUSE WAS BORN
AND LIVED UNTIL HE WAS 18
JOHN VON NEUMANN
1903 — 1957

ONE OF THE MOST OUTSTANDING
MATEMATICIANS OF THE 20TH
CENTURY. PRESIDENT OF THE
AMERICAN MATHEMATICAL
SOCIETY IN 1951 — 1952.

THIS MEMORIAL PLAQUE WAS
ERECTED JOINTLY BY THE
JÁNOS BOLYAI MATHEMATICAL
SOCIETY AND THE AMERICAN
MATHEMATICAL SOCIETY ON THE
100TH ANNIVERSARY OF HIS BIRTH.

Wissenschaftliche vs. Technische Modelle

- Ein wissenschaftliches Modell ist falsch, wenn es der Welt nicht entspricht
 - Die Newtonschen Gesetze gelten für schwarze Löcher nicht
- Ein technisches Modell ist immer richtig.
 - Bei Unstimmigkeiten ist die Welt (die Implementation) falsch.
- Die ersten sind *beschreibende* Modelle
- Die letzteren sind *vorschreibende* Modelle

Was ist ein Modell?

- **Vereinfachtes Bild** eines **Teiles** einer realen oder hypothetischen Welt („des Systems“), das das System in bestimmten Überlegungen **ersetzen kann**
- Entscheidungen:
 - **Welches Teil der Welt?**
 - **Was wird vernachlässigt?**
 - **Wie kann es der Welt entsprechend gemacht werden?**
- Vorteile
 - kleiner (endlich)
 - übersichtlicher

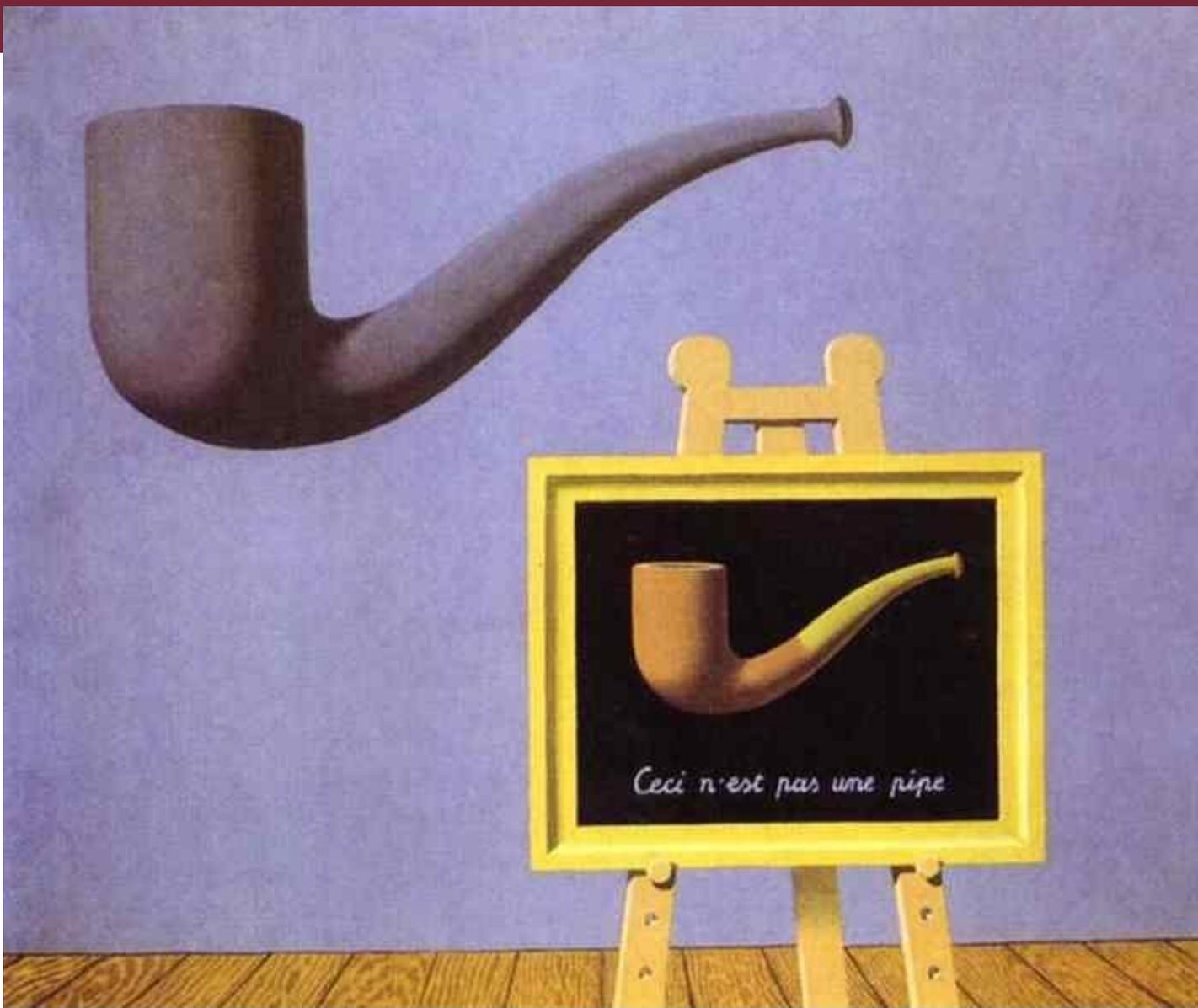
Wann kann es verwendet werden und wann lohnt es sich?

Was ist KEIN Modell?

- Das Modell ist nicht die Wirklichkeit!



- Das Modell ist nicht das Diagramm
 - Das ist nur eine Sicht des Modelles

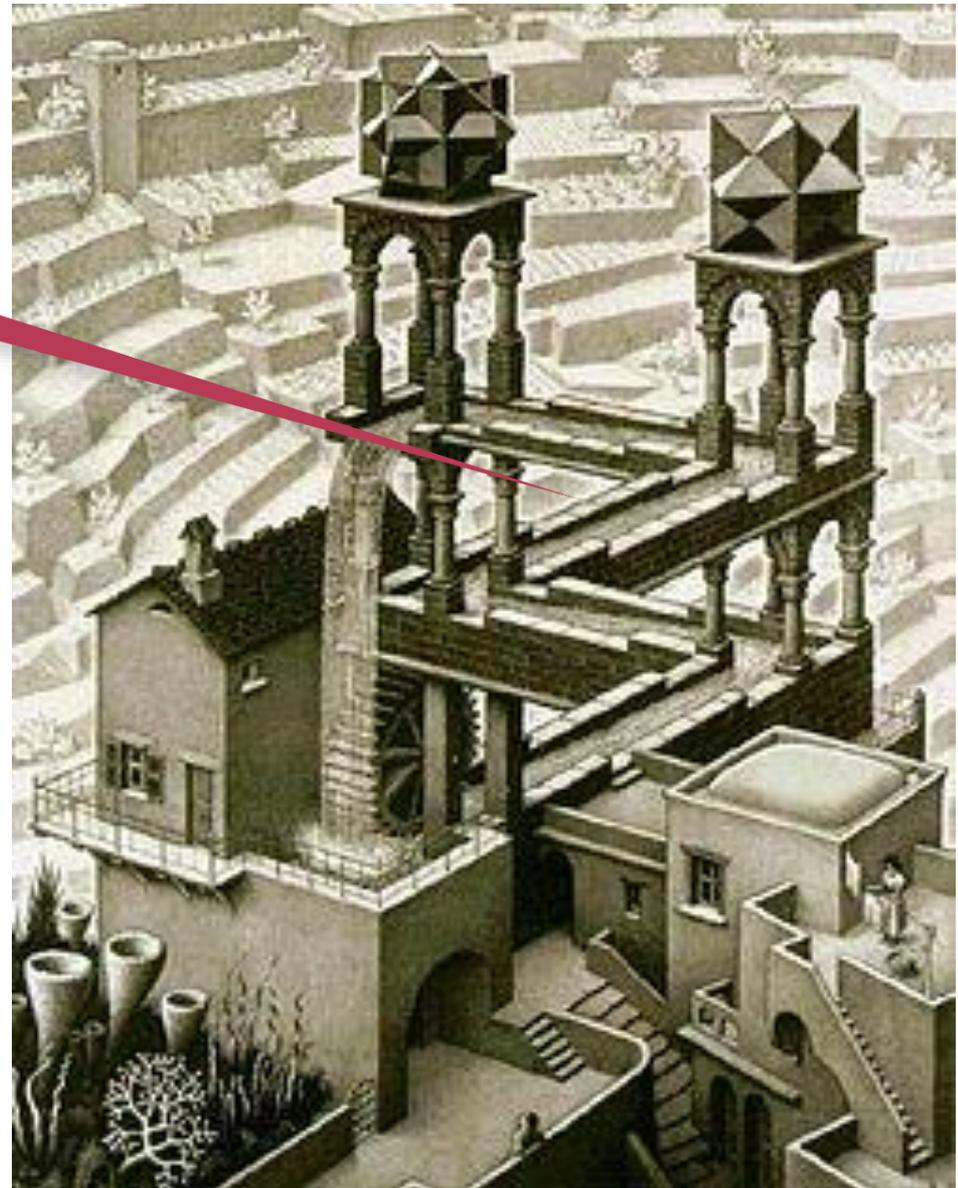


abcgallery.com - Internet's biggest art collection

Modell vs. Wirklichkeit

Modellanalyse?

- Ist die Beschreibung richtig?
(Syntax)
- Ist das Modell realisierbar?
(π berechnen,
endliche/unendliche
Zahlentypen, ...)
- Beschreibt das Modell, was es
beschreiben soll?
(Unendliche Schleifen,
Dividierung mit 0, ...)



Mathematisches Modell vs. Wirklichkeit

- Jedes Modell:
eine geschlossene Welt
 - Effekte, Faktoren
 - Parameter
 - Gültigkeit
- Das Modell funktioniert außer dieser Welt unsicher
- Nicht alles kann im Voraus ausgedrückt werden
 - *menschliche Entscheidung*
 - *generierte Modelle*
- *Validation der Lösung*
- *Für Beantwortung von Fragen gebaut*

- Normale Funktion
 - Randbedingungen
 - Genug Material steht zur Verfügung
 - **Jede** Bestellung termingerecht
 - Zielfunktion:
 - Kosteneffizienz
- Außerordentlicher Fall
 - Randbedingung
 - Materialmangel
 - Zielfunktion:
 1. **Möglichst viele** Bestellung termingerecht
 2. Kosteneffizienz

Mathematisches Modell vs. Wirklichkeit

- Jedes Modell:
eine geschlossene Welt

- Effekte, Faktoren
- Parameter
- Gültigkeit

- Das Modell funktioniert
außer diesen Grenzen

- Nicht
auf

-
-

- *Validierung*
- *Für Beantwortung von
Fragen geeignet*

- Normale Funktion

- Basis

zur

gerecht

**Das Modell ist ein Mittel für die
Beantwortung ein(ig)er Fragen!**

(Fragen über das Modell? Fragen über die Wirklichkeit?)

- Zielfunktion: **Bestimmter Fall**

- Materialmangel

- Zielfunktion:

1. **Möglichst viele** Bestellung
termingerecht
2. Kosteneffizienz

Beispiel: Sicherheitskritische SW

- Bremsen von Flugzeugen: Radbremse + Schub



1993 Warschau: Lufthansa 2904

- (SW) Sicherung:

(load on both wheels) ✘

OR

(one wheel is rotating fast) ✘

→ *(plane on the ground)* ✘

→ **(PILOT CAN BRAKE)** ✘

Wheel sliding

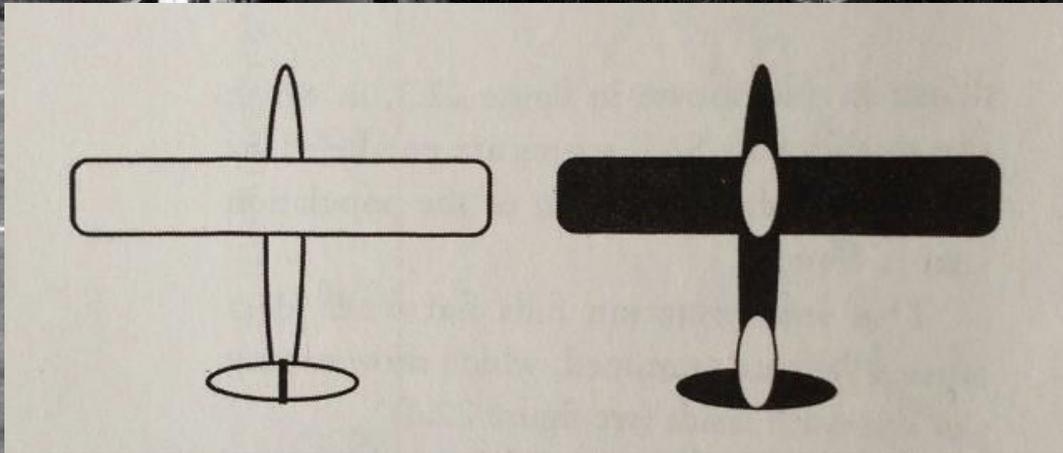
Wheel in the air



Qualität der Modelle

- Realität: eine **offene Welt** \leftrightarrow
Modell: eine **geschlossene Welt**
- **“Treue” eines Modells:**
 - für wahrscheinliche Fälle
 - für kritische Fälle
- Die Umsetzung eines schlechten Modells kann tödlich sein ...

Wald Ábrahám



Inhalt

Modelle und
Modellierung

Wofür werden Modelle
benutzt?

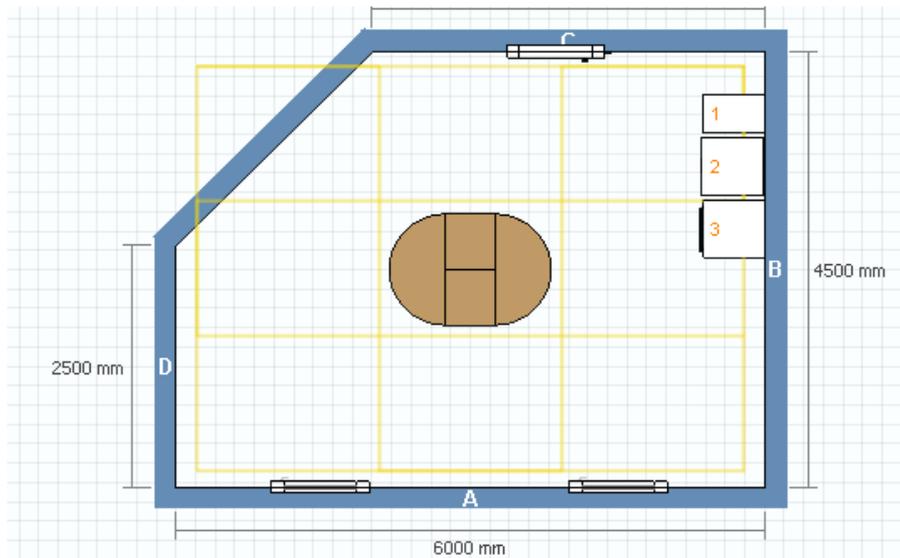
Grundbegriffe

Illustrative
Beispiele

**WOFÜR WERDEN MODELLE
BENUTZT?**

Modellierung in der praktischen Welt?

z.B.: webbasiertes Küchenplaner
[einer schwedischen Firma]



Modellierung in der praktischen Welt?

z.B.: webbasierter Küchenplaner
[einer schwedischen Firma]

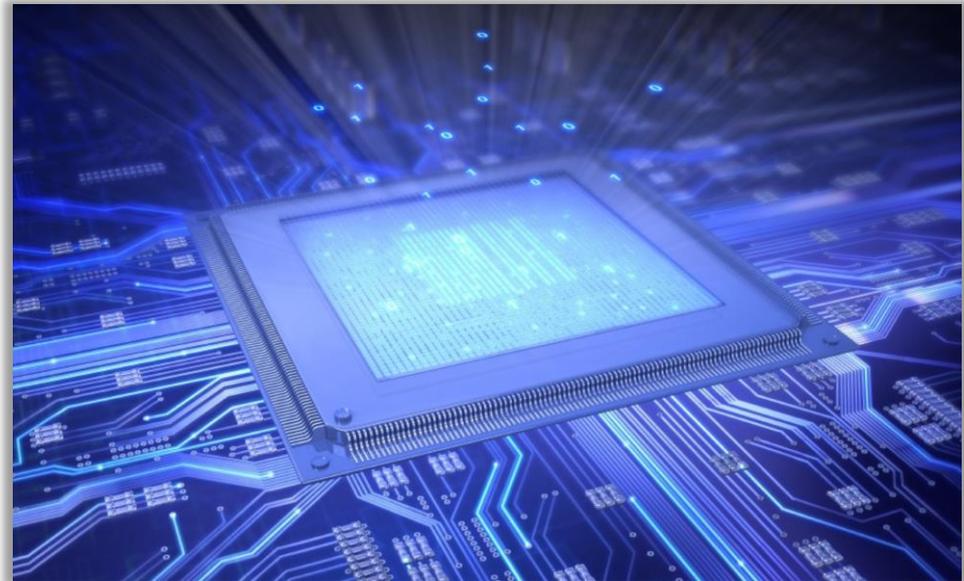
**An einem Modell können
verschiedene Variationen leichter
ausprobiert werden
(als mit echten Möbeln)**



Auch das ist eine Modellierungssprache! 😊

- VHDL, Verilog – Fachgebietsspezifische Hardwarebeschreibungssprachen

```
1
2 ...
3 ARCHITECTURE Struct OF MyLogic IS
4
5     COMPONENT And2 IS
6         PORT (x, y: IN std_logic;
7             f: OUT std_logic);
8     END COMPONENT;
9
10    COMPONENT CustomHW IS
11        PORT (x: IN std_logic;
12            f: OUT std_logic);
13    END COMPONENT;
14
15    SIGNAL n1, n2: std_logic;
16
17 BEGIN
18     And2_1: And2 PORT MAP ( , , );
19     And2_2: And2 PORT MAP ( , , );
20     CustHW: CustomHW PORT MAP ( , , );
21 END Struct;
22
```



- Modelle können HW, SW, komplette Systeme, oder was auch immer beschreiben

Welchen Sinn hat die Modellierung?

- Ich schreibe Software. Soll ich auch modellieren?
 - Du machst es schon!
 - (Der Quellencode der Software ist auch ein Modell)
 - Was wichtiger sind: **mentale Modelle**
 - Wann müssen diese Modelle ausdrücklich *dokumentiert* werden?
 - Hauptfunktion der Modelle: **Kommunikation**
 - Mensch → Mensch
 - Mensch → Maschine
 - Maschine → Maschine
 - Mensch → er selbst, wenig später
 - z. B. sollte man sich jahrelang auf die Gründe der Entwurfsentscheidungen erinnern

Modellierungssprachen

- Das Ziel ist die Kommunikation
 - Verständnis des Modells ist nötig
 - **Modellierungssprachen** (Wann brauchen wir sie?)
- Syntax (**Wie beschreibe ich das Modell? zB. in „SystemC“**)
 - „mathematische Struktur“: abstrakte Syntax
 - Darstellung: konkrete Syntax
 - graphische Symbole / Textformat
- Semantik (**Was bedeutet „i++“?**)
- Randbedingungen, Einschränkungen
 - Syntaktische Korrektheit, Wohlgeformtheit
 - Entwurfskonventionen (jede Gruppe hat ihre eigene)

Modell einer Modellierungssprache

Die konkrete Strecke:
Modell

Mögliche Streckenelemente:
Metamodell

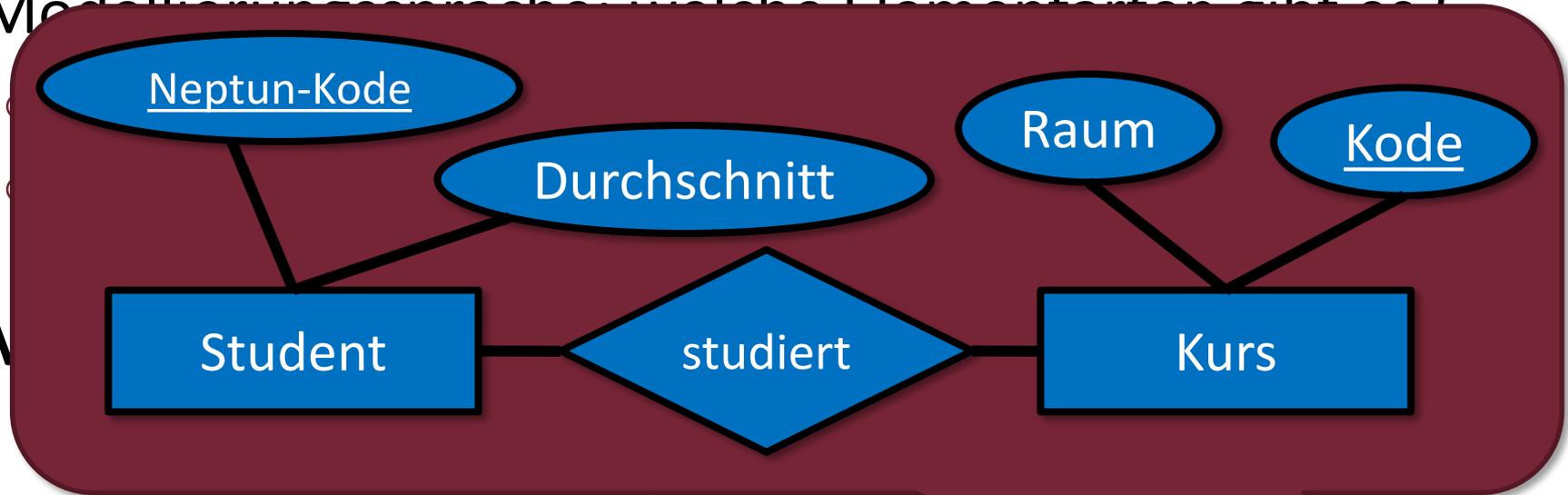


Grundbegriffe - Metamodellierung

- Modellierungssprache: welche Elementarten gibt es?
 - ... was für Relationen kann es unter diesen Elementen geben?
 - ... wie sind die Relationen zwischen diesen Elementarten?
- **Metamodell** = das Modell einer Modellierungssprache

Grundbegriffe - Metamodellierung

- Modellierungsmetamodelle, welche Elementartypen sieht es?



- M

- Illustrationen, die jeder kennt

- Datenbankplatte → Datenbankschema mit Relationen
- XML Dokument → XML Schema (oder DTD)
- Datenbankschema → Individuum-Verbindung (ER) Modell
- UML Class Diagramm → Klassendiagramm
- C++ Objekte → C++ Klassen

Modelle in der Ingenieurarbeit

■ Ingenieurarbeit

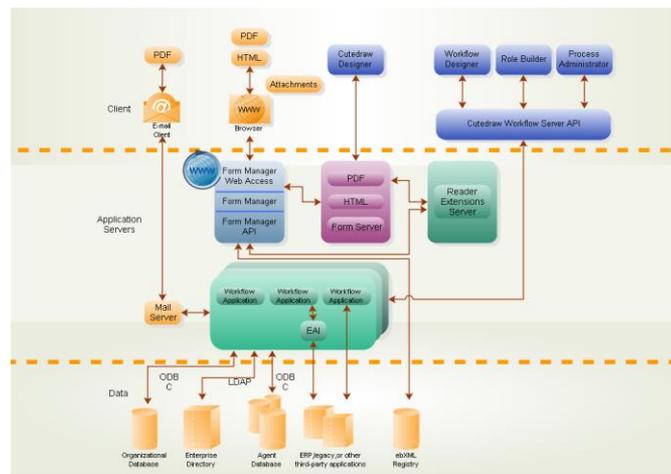
(Architekten, Maschinenbauingenieure, Elektroingenieure, ..., Landschaftsarchitekten, ...)

- Konzeptentwurf
- Beschreibung → Zeichnung von Modellen
 - Verfeinerung → Modellverfeinerung
- Dimensionierung → Modellverfeinerung
- Überprüfung → Modellanalyse u. Simulation
- Bau → Implementation, Ableitung

Es ist die altbewährte Ingenieurmethode: **Planung**

Anwendung – Dokumentation

- Das Modell ist einfacher
 - leichter kommunizierbar, als die ganze Wirklichkeit
 - ständig verfeinerbar (siehe später ...)
- Kommunikation, Veranschaulichung
 - Demonstration (siehe später ...)
 - verständliche textuelle Sprache
 - anschauliches Diagramm
- Unterstützung für Konzept- und Produktentwicklung
 - die Aspekte sind ähnlich
 - „Selbstkommunikation“



„Der Kode als Dokumentation“



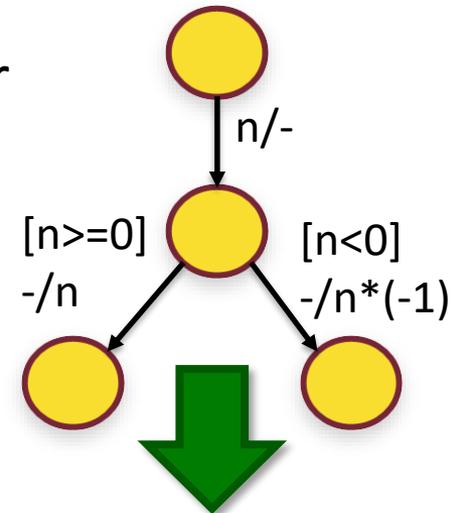
Anwendung – Simulation

- Demonstration
 - Als Mittel der Kommunikation
- Validation
 - „Ich habe es richtig erbaut ..., aber habe ich das Richtige gebaut?“
- Experimente
 - Überprüfung von bestimmten erwünschten Eigenschaften
 - Messung von quantitativen Eigenschaften
 - Ersatz für in der Wirklichkeit kostspielige Versuche
 - für Eigenschaften, die auf dem theoretischen Weg nicht zu bestimmen sind



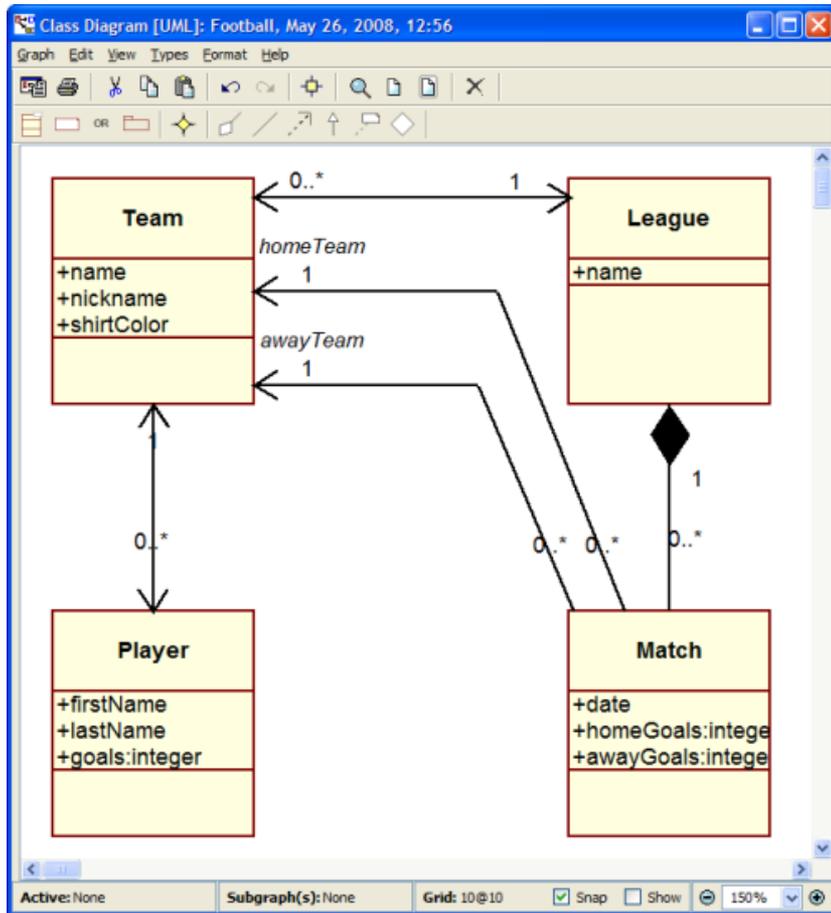
Anwendung – Ableitung

- Manuell oder (teilweise) **automatisiert**
- Ergebnis
 - Generierung von Programmcode, analysierbarer Sprache, Dokumentation, uvm.
 - Ein anderes Modell
 - Verfeinerung, nächste Entwurfsphase
 - Teilaspekte
 - Integration der Modelle
- Es kann eigenschaftserhaltend sein
- Z.B.: Berechnung des Absolutwertes



```
int n;  
scanf("%d", &n);  
if (n >= 0) {  
    printf("%d", n);  
} else {  
    n = n * (-1);  
    printf(„%n“, a);  
}
```

Entwicklung einer Webanwendung



League [Gears] - Mozilla Firefox

File Edit View History Bookmarks Tools Help

League [Gears]

Enter a League to store in the database:

name

OK

Teams name

Matches date

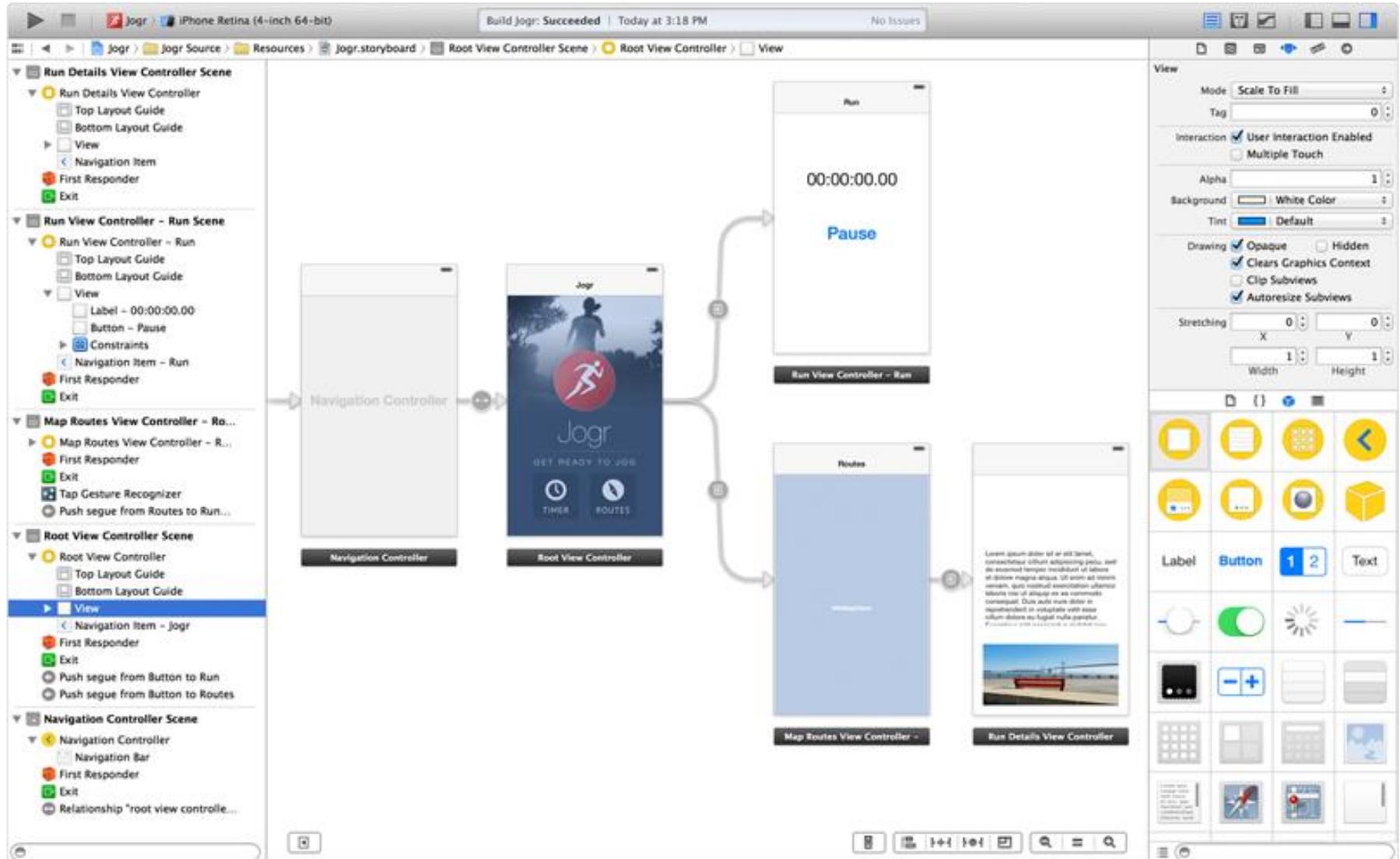
4 most recently edited League entries:

name	
<input type="button" value="Edit"/>	Premiership <input type="button" value="Delete"/>

[Back to top page of Football application.](#)

This page uses Gears to record your entries on the local disk. If you navigate away and revisit this page, all your data will still be here. Try it!

Entwicklung einer Smartphone-Anwendung



Reverse Engineering

- (Wieder)Herstellung des Modelles anhand des Systems

League [Gears] - Mozilla Firefox

File Edit View History Bookmarks Tools Help

League [Gears]

Enter a League to store in the database:

name

OK

Teams name

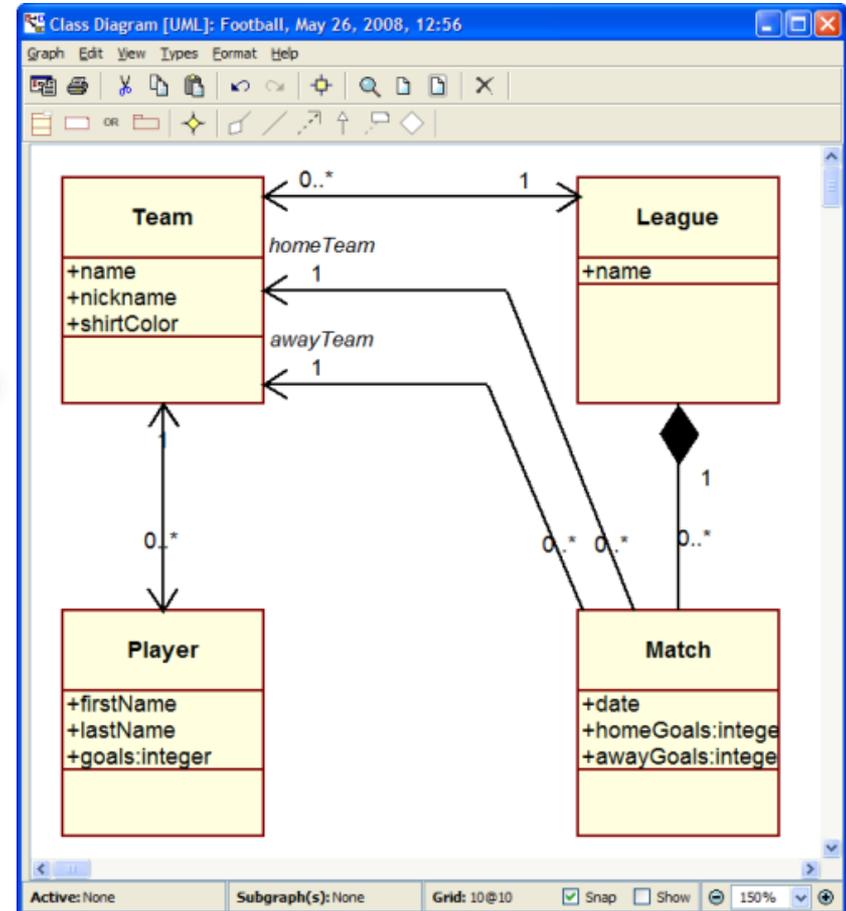
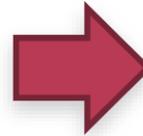
Matches date

4 most recently edited League entries:

name
<input type="button" value="Edit"/> Premiership <input type="button" value="Delete"/>
<input type="button" value="New"/> <input type="button" value="View All"/> <input type="button" value="Delete All"/>

[Back to top page of Football application.](#)

This page uses Gears to record your entries on the local disk. If you navigate away and revisit this page, all your data will still be here. Try it!

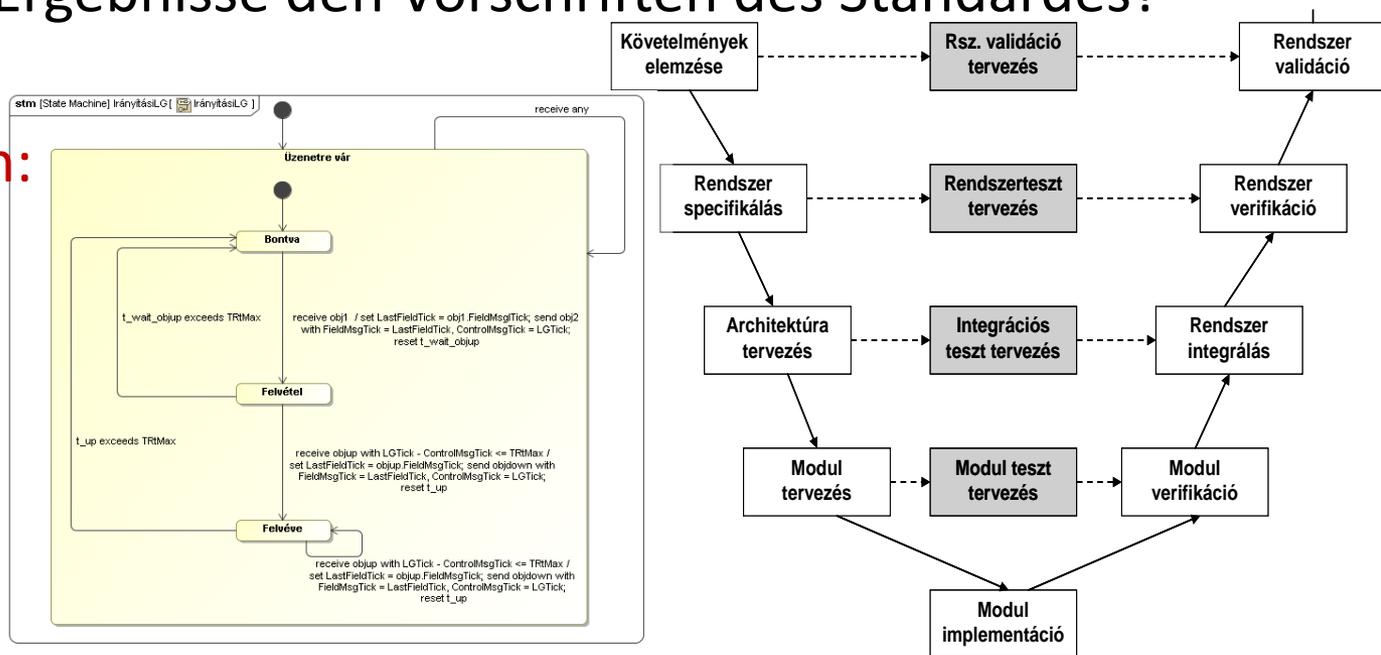


Entwicklung von Bahnsoftware

- **Zu lösendes Problem:**
Überprüfung der Entwicklungsphasen einer SIL-4 Bahnsignalvermittlungsanwendung und dessen Ergebnissen aufgrund des für die Software in Bahnwesen geltende Standardes (EN50128)
- Entsprechen die Ergebnisse den Vorschriften des Standardes?

Herausforderungen:

- systematische Überprüfung
- Modellierung und formale Verifikation



Prozess → Modell → Qualitätssicherung

Entwicklung von Bahnsoftware

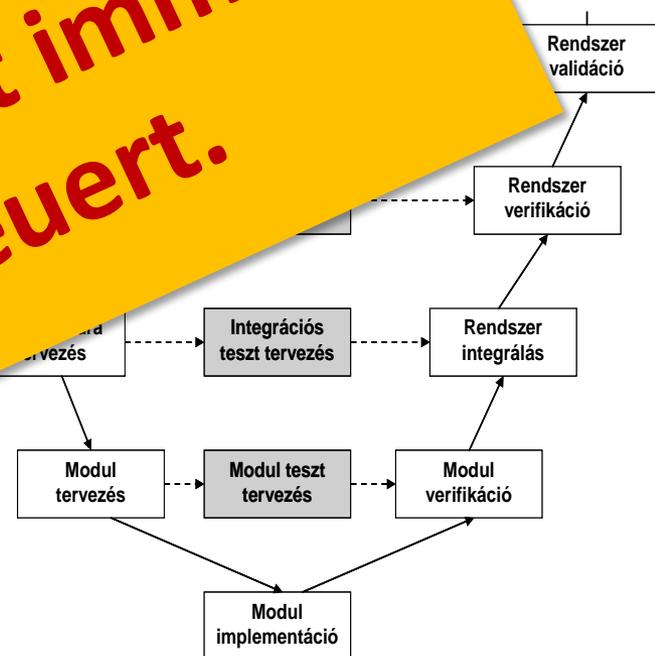
- **Zu lösendes Problem:**
Überprüfung der Entwicklungsphasen einer SIL-4 Bahnsignalvermittlungsanwendung und dessen Konformität mit dem europäischen Standard (EN50128)
- Entsprechen die Ergebnisse...

Herausforderungen

- systematische Überprüfung
- Modellierung und formale Verifikation

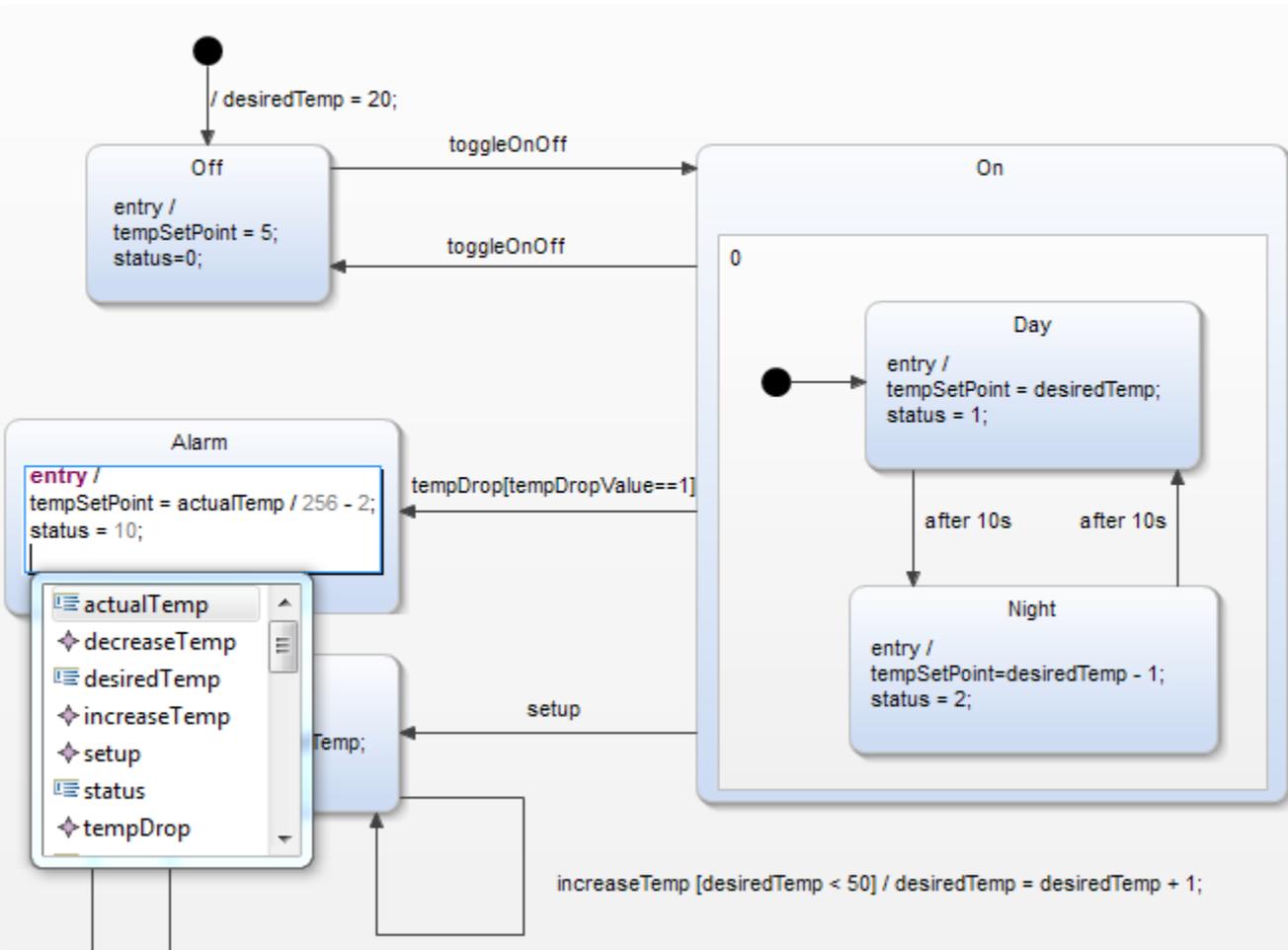
Die Entwicklung kritischer Anwendungen ist immer modellgesteuert.

```
receive objup with LGTick -> ControlMsgTick <= TRTMax /
set LastFieldTick = objup.FieldMsgTick; send objdown with
FieldMsgTick = LastFieldTick; ControlMsgTick = LGTick;
reset L_up
```



Prozess → Modell → Qualitätssicherung

Yakindu - Zustandsdiagramm



```

import java.io.*;
import java.net.*;
import java.security.*;
import protection;

public class Client {
    public void sendAuthentication(String user,
    OutputStream outStream) throws IOException {
        DataOutputStream out = new DataOutputStream(
        OutputStream outStream).getTime();
        long t1 = Math.random();
        byte[] protected1 = Protection.main(
        long t2 = (new Date()).getTime());
        double q1 = Math.random();
        double q2 = Math.random();
        byte[] protected2 = Protection.
        out.writeUTF(user);
        out.writeInt(protected1.length);
        out.write(protected2);
        out.flush();
    }
}

public static void main(String[] args) {
    String host = args[0];
    int port = 7999;
    String user = "John";
    String password = "shh";
    Socket s = new Socket(host, port);
    Client client = new Client(s);
    client.sendAuthent
  
```

Yakindu - Zustandsdiagramm



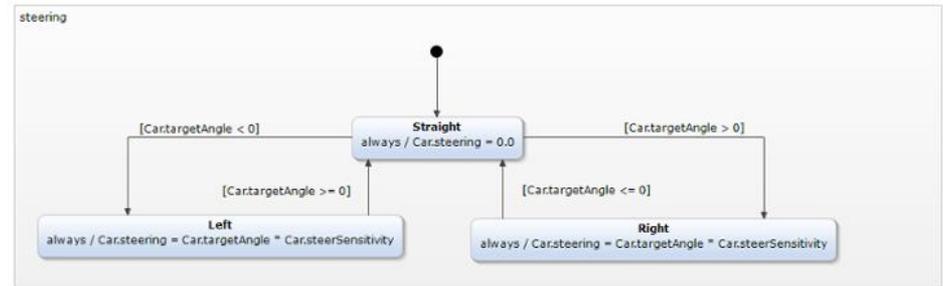
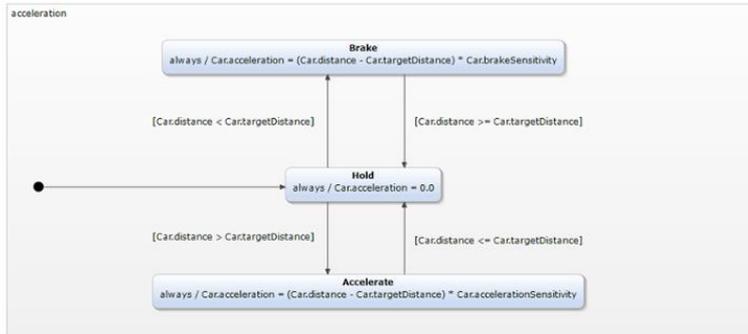
Ein gemeinsames Modell
Validierung + Simulation + Kodengenerierung
Konsistenz

```
public static void main(st
String host = args[0];
int port = 7999;
String user = "John";
String password = "sh
Socket s = new socket

Client client = new
client.sendAuthent
```

Yakindu → 3D szimuláció

Interaktive 3D-Simulation eines Verfolgung/Automatische Distanzregelung (ADR) Systems



Interactive 3D Visualization and Simulation with State Machines, Benjamin Bolte, 2017.
<https://blogs.itemis.com/en/interactive-3d-visualization-and-simulation-with-state-machines>

Inhalt

Modelle und
Modellierung

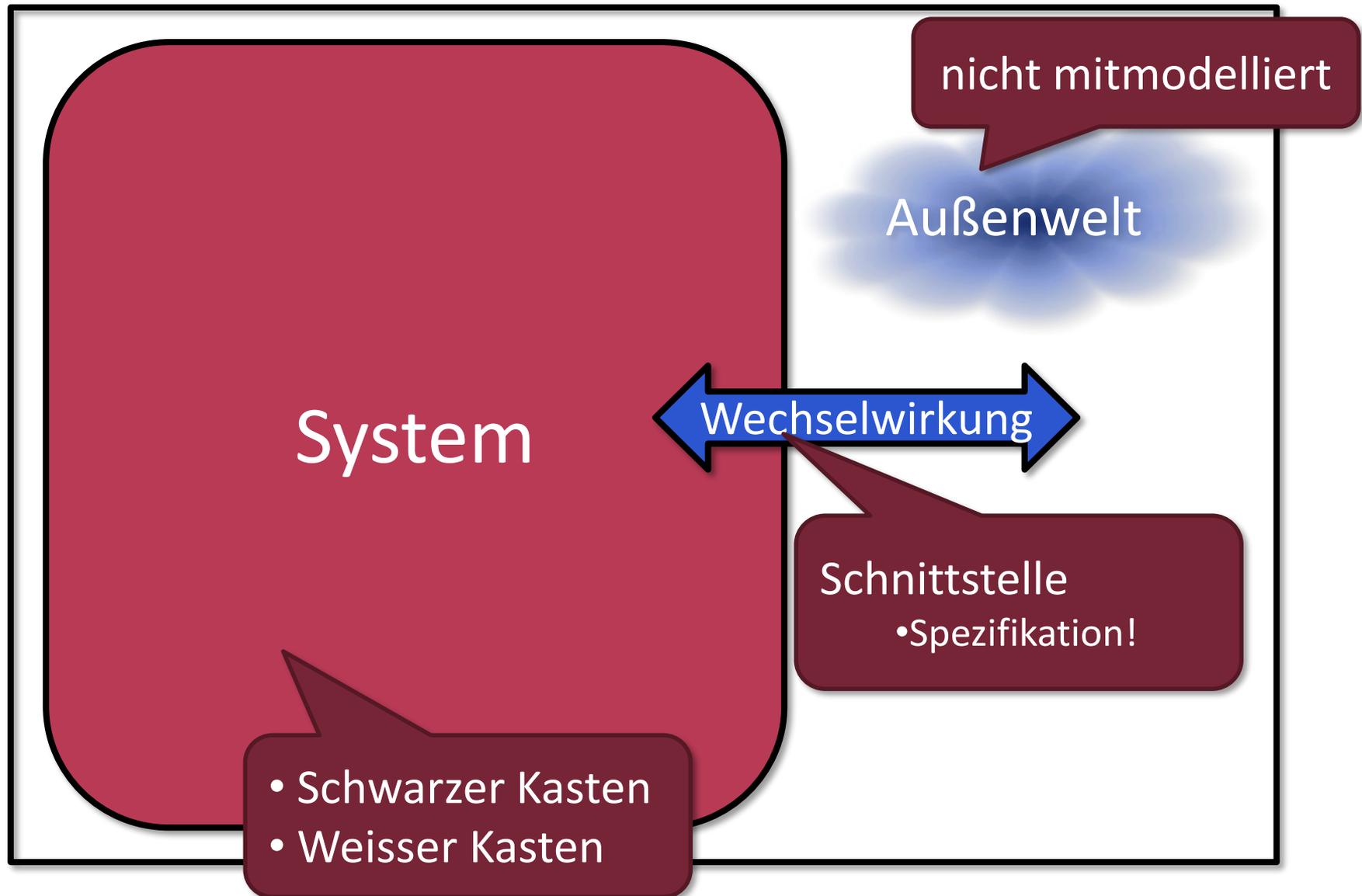
Wofür werden Modelle
benutzt?

Grundbegriffe

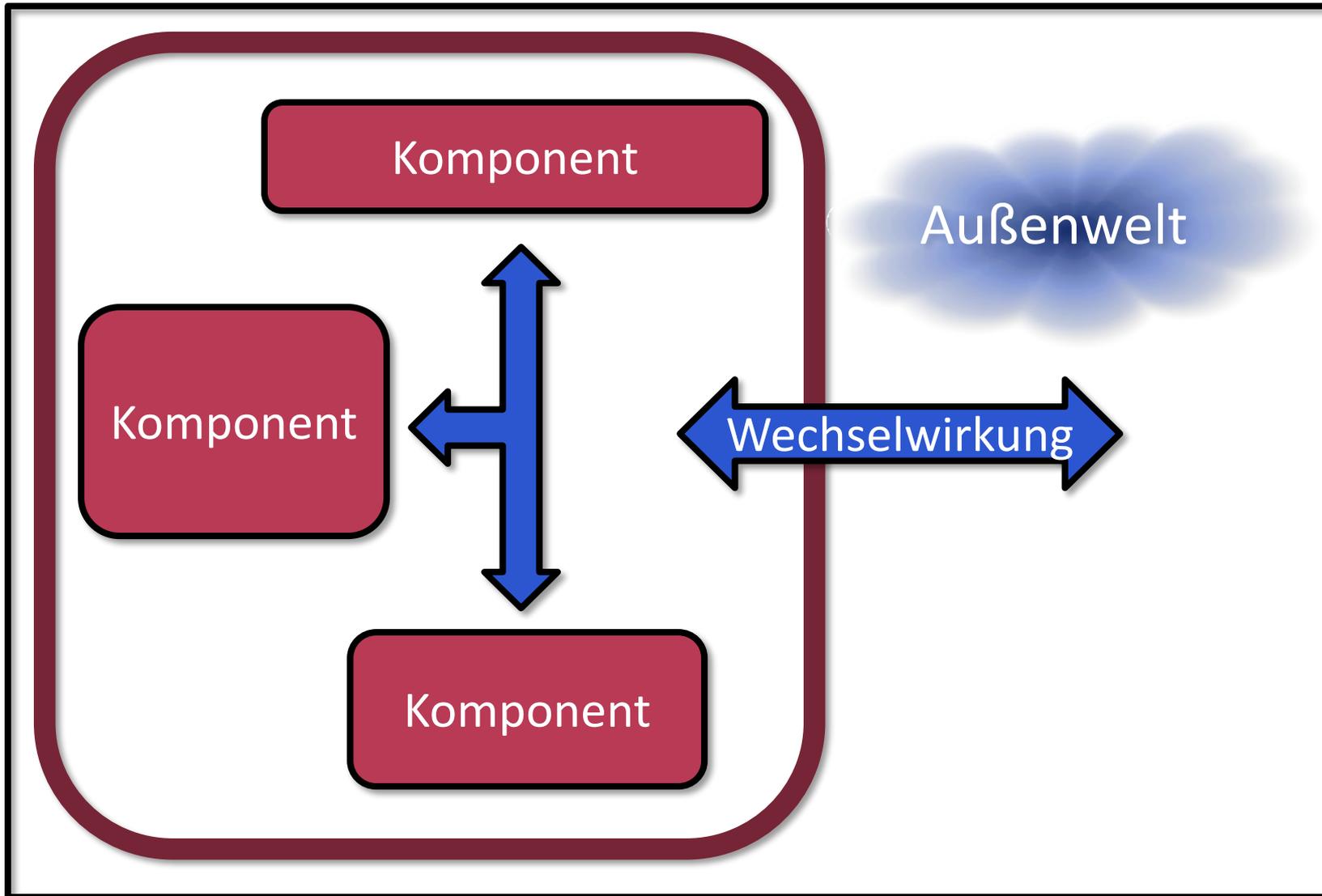
Illustrative
Beispiele

Grundbegriffe der Modellierung

Grundbegriffe – System und Außenwelt



Grundbegriffe – System und Außenwelt



Grundbegriffe – Verfeinerung/Abstraktion

■ *Verfeinerung:*

Bereicherung des Modells mit Einzelheiten

- ... so dass das originale Modell dessen Abstraktion bleibt
- Zusätzliche, *aber nicht widersprechende(!)* Einzelheiten
- Auf die vorigen Folie wurde eine *hierarchische Verfeinerung*
 - „Kasten auspacken“

■ *(vertikale) Abstraktion:* Inverse der Verfeinerung

■ Nicht nur die Struktur kann verfeinert werden ...

- z.B. Mengenverfeinerung: Wertemengen von Variablen
 - anstatt **gut** / **schlecht**
 - **schnell** / **durchschnittlich** / **langsam** / **mangelhaft** / **gefährlich**

Grundbegriffe – Verfeinerung

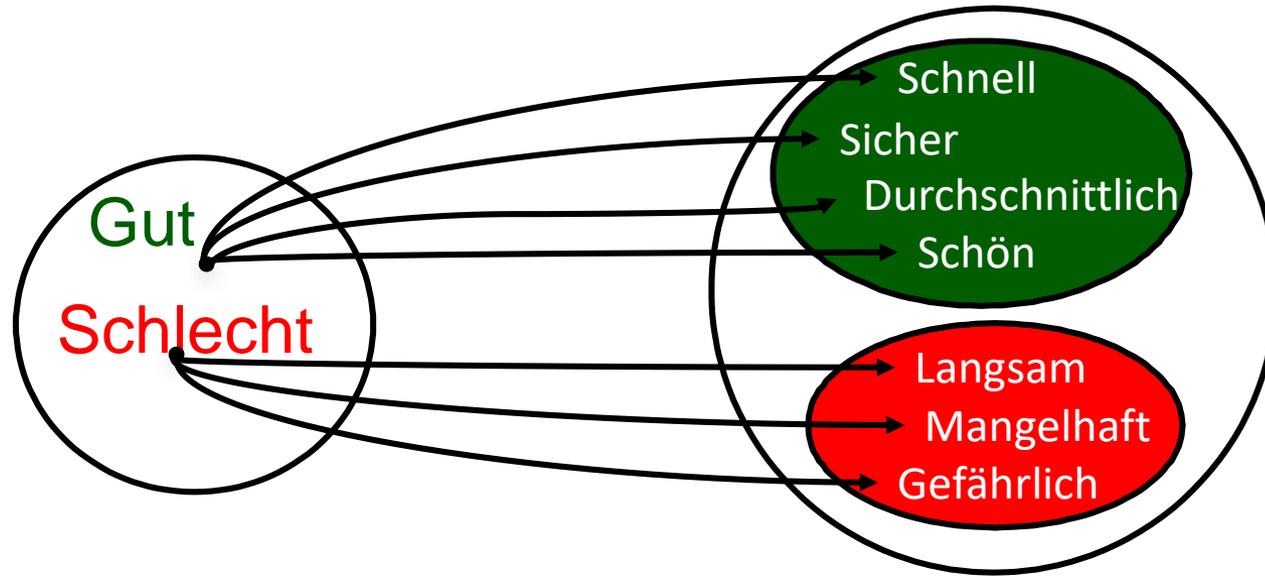


Grundbegriffe – Verfeinerung



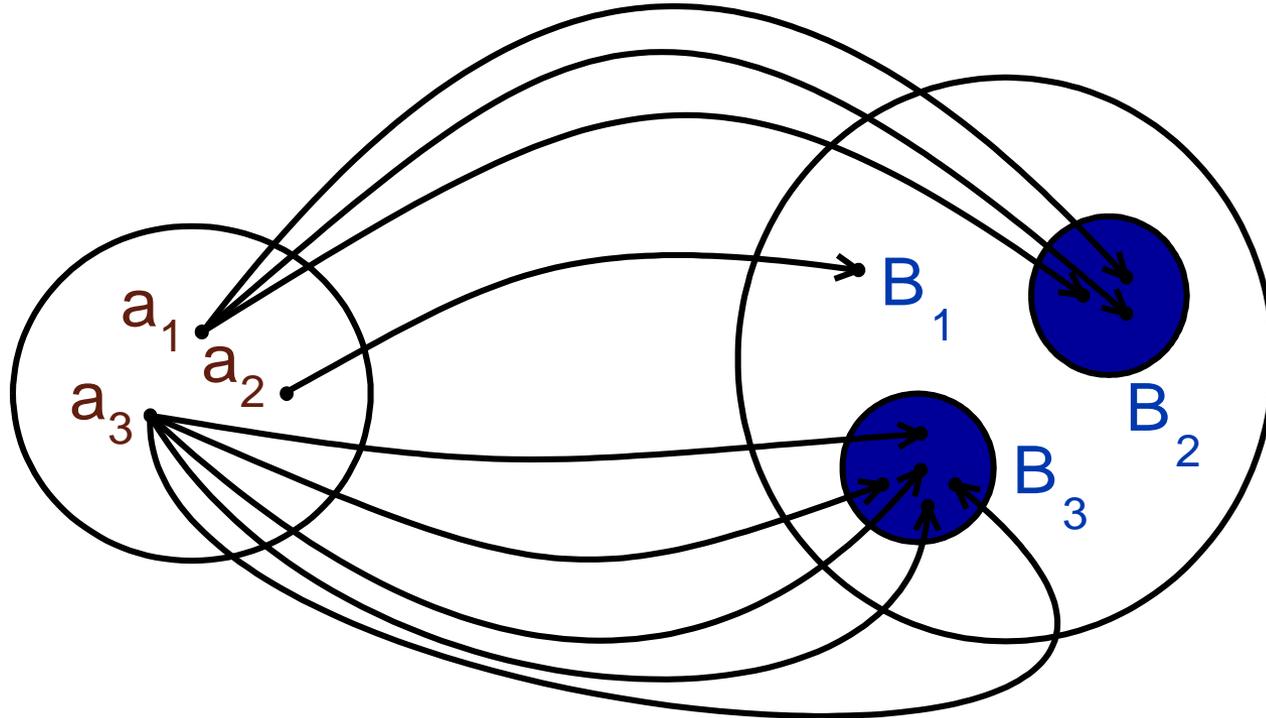
Mengenverfeinerung

Zuordnung disjunkter Teilmengen zu den Elementen



Mengenverfeinerung

Zuordnung disjunkter Teilmengen zu den Elementen



$\forall a_i \in A, R(a_i) \subset B$ so, dass $R(a_i) \cap R(a_j) = \emptyset \quad \forall i \neq j$

Abstraktion

- Die Abstraktion ist **korrekt**, wenn zu jedem Element des Originalsystems ein Element des abstrakten Systems zugeordnet werden kann.
- Die Abstraktion ist **vollständig** bezüglich der betrachteten Eigenschaft, wenn jedes Merkmal des Systems auch nach der Abstraktion erscheint, welches die betrachtete Eigenschaft beeinflusst.

(Verschiedene Definitionen sind hier möglich.)

Beispiel: Refactoring

- Ordnung von Zahlen
 - Aufsteigend: 18,23,131,247,1925
 - Absteigend: 1925,247,131,23,18
 - Alphabetisch: 131,18,1925,23,247
- Kode (Minimumsuche):

```
void Ordnung_aufsteigend(int* Liste, int Anzahl){
    int i, j;
    for(i = 0; i<Anzahl; i++){
        int MinimumIndex = i;
        for(j= i + 1; j<Anzahl; j++){
            if(Liste[j] < Liste[MinimumIndex])
                MinimumIndex = j;
        }
        int tmp = *i;
        *i = *MinimumIndex;
        *MinimumIndex = tmp;
    }
}
```

Was ist der Unterschied im Kode bei den anderen zwei Ordnungen?

Beispiel: Refactoring

- Ordnung von Zahlen
 - Aufsteigend: 18,23,131,247,1925
 - Absteigend: 1925,247,131,23,18
 - Alphabetisch: 131,18,1925,23,247
- Kode (Minimumsuche):

```
void Ordnung_aufsteigend(int* Liste, int Anzahl){
    int i, j;
    for(i = 0; i<Anzahl; i++){
        int MinimumIndex = i;
        for(j= i + 1; j<Anzahl; j++){
            if(Liste[j] < Liste[MinimumIndex])
                MinimumIndex = j;
        }
        int tmp = *i;
        *i = *MinimumIndex;
        *MinimumIndex = tmp;
    }
}
```

Was ist der Unterschied im Kode bei den anderen zwei Ordnungen?

Beispiel: Refactoring

- Gedanke: Der Vergleich könnte in einer getrennten Funktion implementiert werden

```
void Ordnung(int* Liste, int Anzahl, bool (*vergleich)(int,int)) {
    int i, j;
    for(i = 0; i<Anzahl; i++){
        int MinimumIndex = i;
        for(j= i + 1; j<Anzahl; j++){
            if( (*vergleich)(Liste[j],Liste[MinimumIndex]) )
                MinimumIndex = j;
        }
        int tmp = *i;
        *i = *MinimumIndex;
        *MinimumIndex = tmp;
    }
}
```

Was kann aus diesem Code noch wiederverwendet werden?

→ Vermeidung der Kodeduplikation

Beispiel: Refactoring

- Gedanke: Der Vergleich könnte in einer getrennten Funktion implementiert werden

```
void Ordnung(int* Liste, int Anzahl, bool (*vergleich)(int,int)) {
    int i, j;
    for(i = 0; i<Anzahl; i++){
        int MinimumIndex = i;
        for(j= i + 1; j<Anzahl; j++){
            if( (*vergleich)(Liste[j],Liste[MinimumIndex]) )
                MinimumIndex = j;
        }
        int tmp = *i;
        *i = *MinimumIndex;
        *MinimumIndex = tmp;
    }
}
```

Was kann aus diesem Code noch wiederverwendet werden?

→ Vermeidung der Kodeduplikation

Rolle der Abstraktion in Programmierung

- Anmerkung:
 - Ordnungsalgorithmus mit Auswählen: eine Abstraktion
 - Verschiedene Vergleichsfunktionen (→ Ordnungen): konkrete Instanzen
- Guter Kode:
 - Algorithmen mit Parametern
 - Verschiedene Anwendungen → versch. Instanzen
 - z.B. dieser Algorithmus kann auch für Ordnen von Wörtern benutzt werden

Modellierung in anderen LVAs

- Wir bauen auf ...
 - *Digitale Systeme*: Zustandsmaschinen, Bool-Algebra
 - *Programmieren1*: Grundbegriffe, Datenstrukturen
 - *Dokumentation und Presentation*: Xmind
- Auf uns wird gebaut in ...
 - *Datenbanksysteme*: Datenstrukturen
 - *Software-Technologien*: Modellbasierter Entwurf
 - *Operationssysteme*: Leistungsmodellierung
 - *Rechnernetzwerke*: Simulation
 - *Künstliche Intelligenz*: Ontologien
 - *Systemtheorie*: Simulink Kontrollelemente

Inhalt

Modelle und
Modellierung

Wofür werden Modelle
benutzt?

Grundbegriffe

Illustrative Beispiele

Illustration

Einige repräsentative Projekte



R3COP (EU ARTEMIS)

- Automated testing of robots
- Robustness and security analysis
- ARTEMIS Innovation Award 2012
- Altogether: 15 EU projects

TRANS-IMA (Embraer)

- Eclipse based development tooling
- HW-SW allocation: avionics architecture
- Integration to the distributed Embraer simulator
- (1st time in Europe)

Data Storage Systems (IBM)

- Supply-chain simulation and optimization
- Prediction of order data
- IBM Vác: Data Storage Systems
- Supply Chain Technology Award 2012

VCL: Virtual Computing Lab

- Open source cloud infrastructure
- Apache project
- Education: lab courses" (BYOD)
- First time in Hungary
- Tempus Award