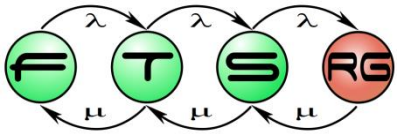
Systemmodellierung

Dr. PATARICZA András

HUSZERL Gábor

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



Die Lehrveranstaltung

- Lehrveranstaltung (VIMIAA00)
  - 3 in 1 (auf ungarisch/deutsch/englisch)

- Dr. PATARICZA András
  - Verantwortlicher Professor
- GÖNCZY László
  - Operative Leitung, Organisatorische Fragen

- HUSZERL Gábor (huszerl@mit.bme.hu)
  - deutschsprachige Vorlesung, …

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

- 13 Vorlesungen
  - Mittwochs 10-12 Uhr, I.L405 (keine VO am 1. Mai)
- 6 Übungen
  - Freitags 10-12 Uhr (in ungeraden Wochen), I.L405
  - Zu erst am 22. Februar
  - Tests (keine Eingangstests!)
  - Anwesenheitspflicht (an mind. 4 aus den 6 Übungen)
- 1 Hausaufgabe (mit mündlicher Verteidigung)
- 2 Klausuren (mit Eingangstest)

Bewertung

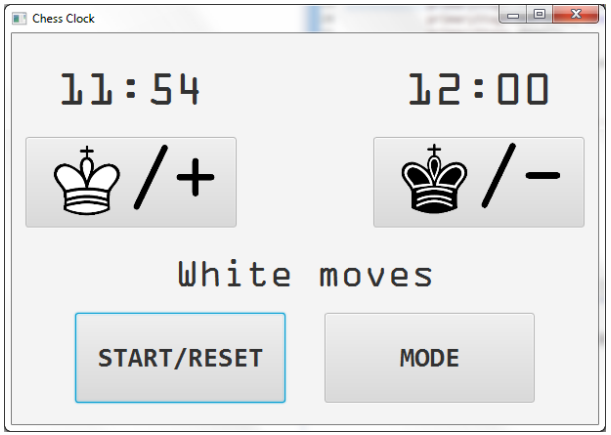
- Klausuren (mit Eingangstest): 35%+35% der Endnote
  - am Freitag, dem 29. März, 8-10 Uhr (24. Mai 8-10 Uhr)
  - am Freitag, dem 17. Mai, 8-10 Uhr (24. Mai 10-12 Uhr)
- Hausaufgabe (mit Verteidigung): 30% der Endnote

- Alle drei Teile mit mindestens 40% der Punkte
  - die Klausuren 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: 3-5. Semesterwoche
  - Ausgabe der Aufgaben: 6. Semesterwoche
  - Abgabe: 12. Semesterwoche
- Elektronische Abgabe
- Mündliche Verteidigung



Thematik

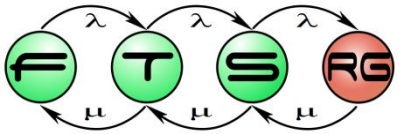
- Strukturmodelle
- Verhaltensmodelle
  - Zustandsmodelle, Prozessmodelle
- Entwicklung von Modellen
- Überprüfung von Modellen
- Leistungsmodellierung
- Visuelle Datenanalyse
- Simulation
- Benchmarking, Kodegenerierung

Grundlagen der Modellierung

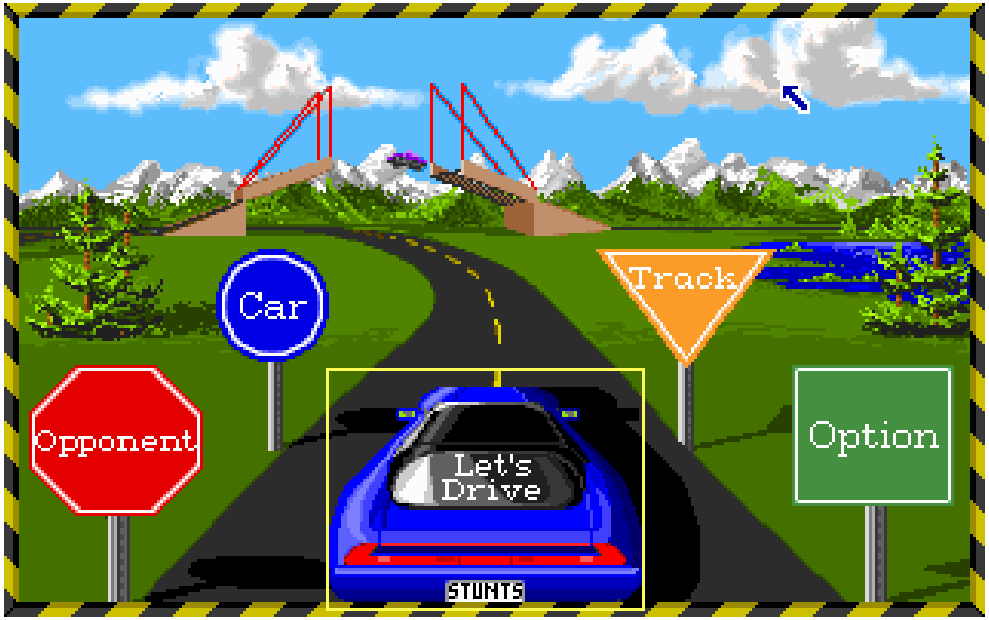
Dr. PATARICZA András, GÖNCZY László

HUSZERL Gábor

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group

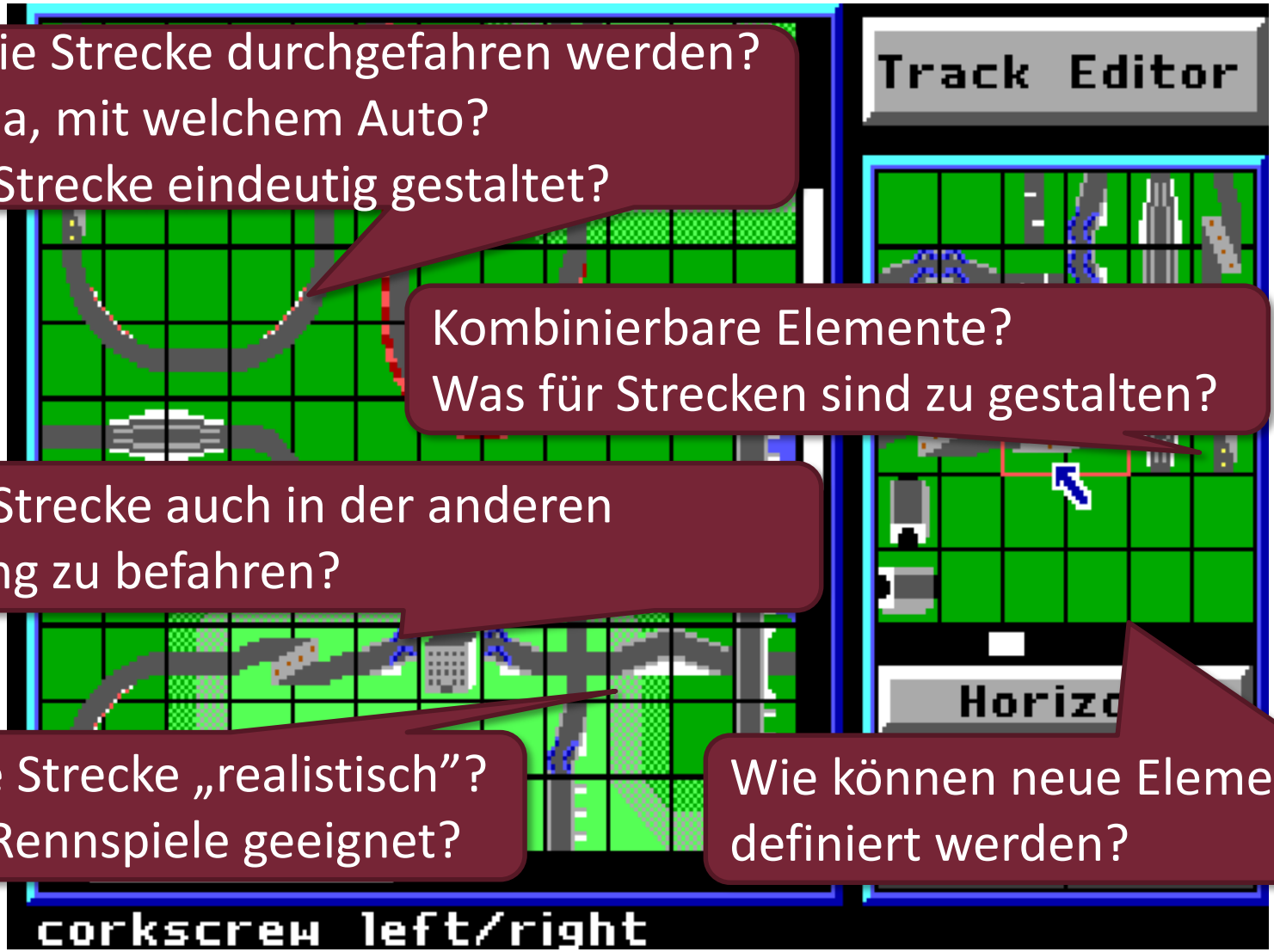


“Motivation”



- Stunts Autorennspiel
  - Distinctive Games/Brøderbund Software, 1990
  - https://en.wikipedia.org/wiki/Stunts_(video_game)
  - 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?

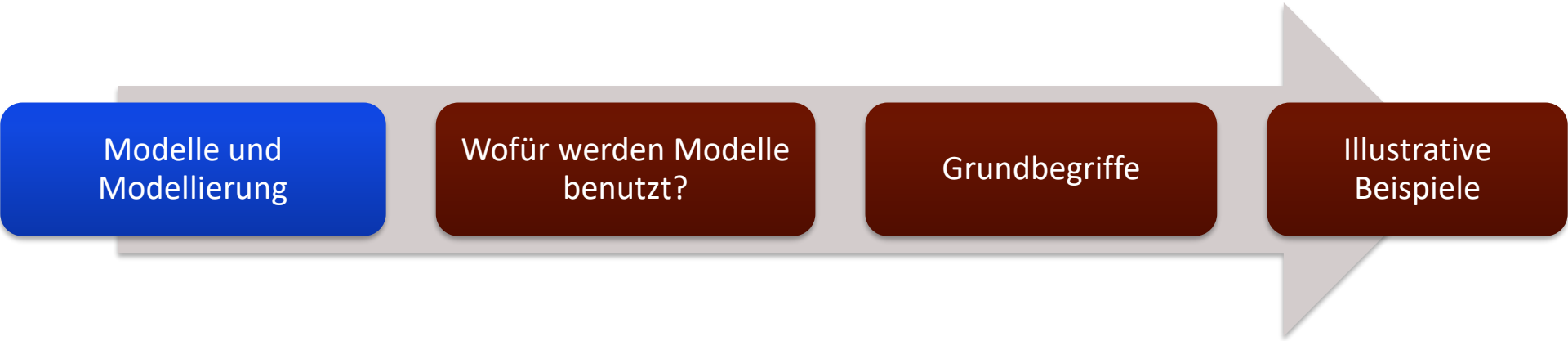
Wie können neue Elemente definiert werden?

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

Inhalt

- Modelle und Modellierung
- Wofür werden Modelle benutzt?
- Grundbegriffe
- Illustrative Beispiele

Inhalt

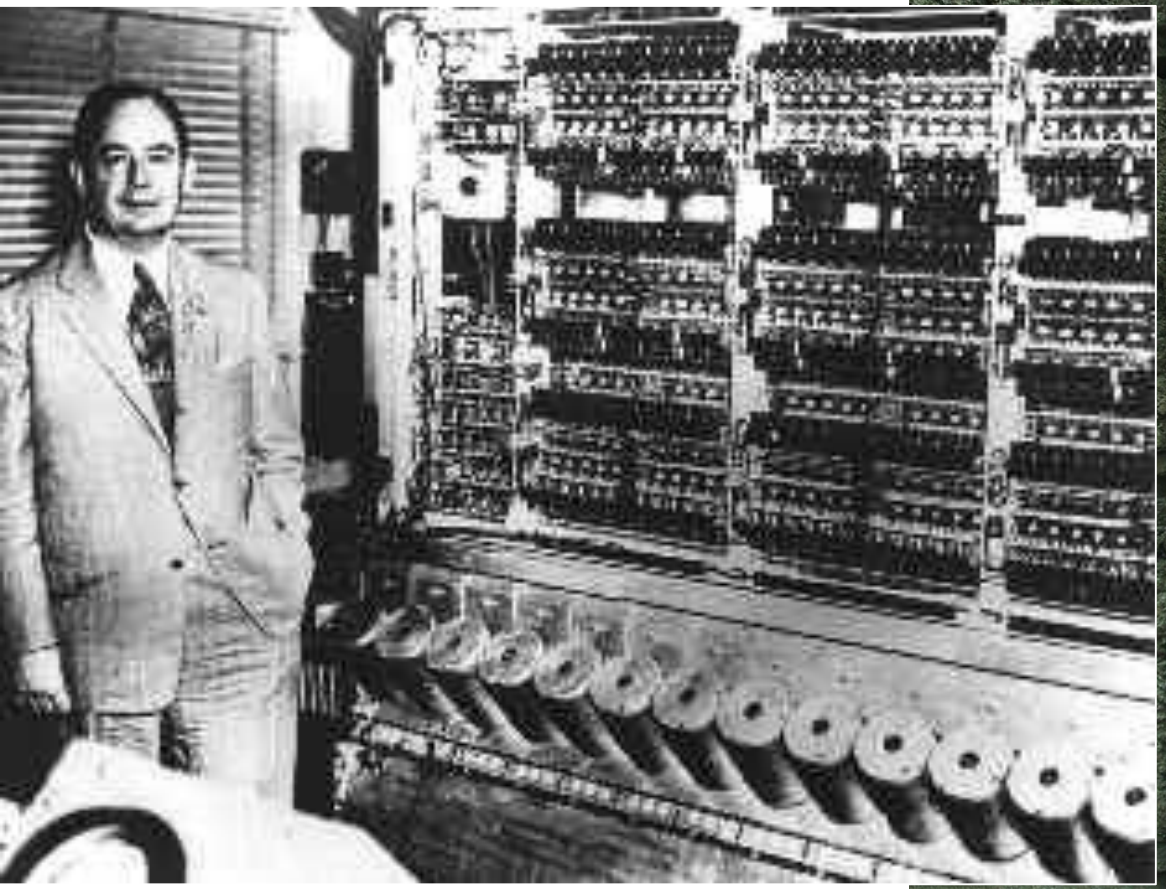


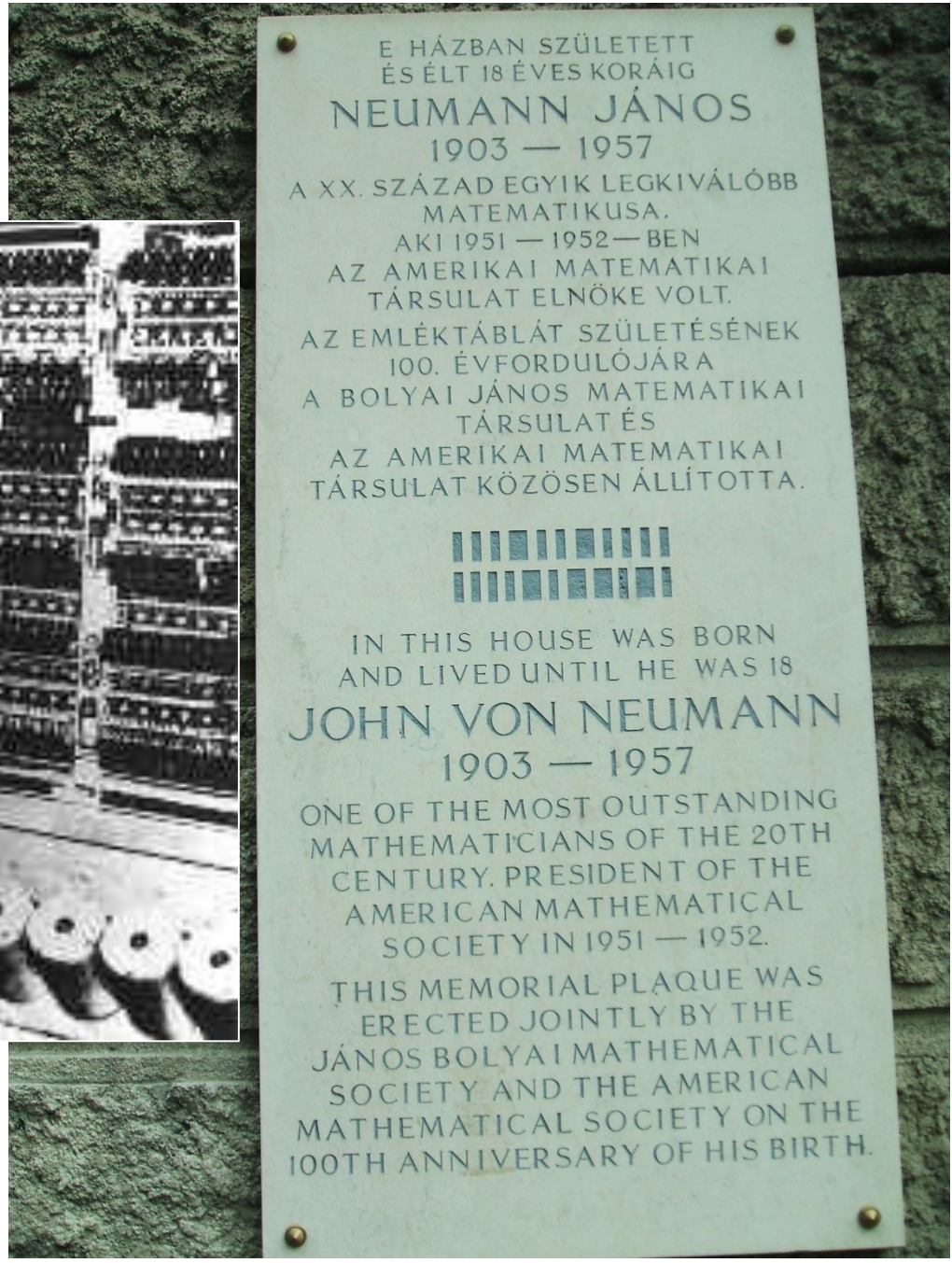
Modelle und Modellierung

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
MATHEMATICIANS 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.

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?**

  } **Wann kann es verwendet werden und wann lohnt es sich?**
- Vorteile
  - kleiner (endlich)
  - übersichtlicher

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

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 funkti
  außer diese
- Nicht
  au
  - *g*
- *Validi*
- *Für Bec von*
  Fragen g

- Normale Fu
  - Ra
    zur
    erecht
    licher Fall
    ubedingung
    - Materialmangel
  - Zielfunktion:
    1. Möglichst viele Bestellung termingerecht
    2. Kosteneffizienz

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

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

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)**

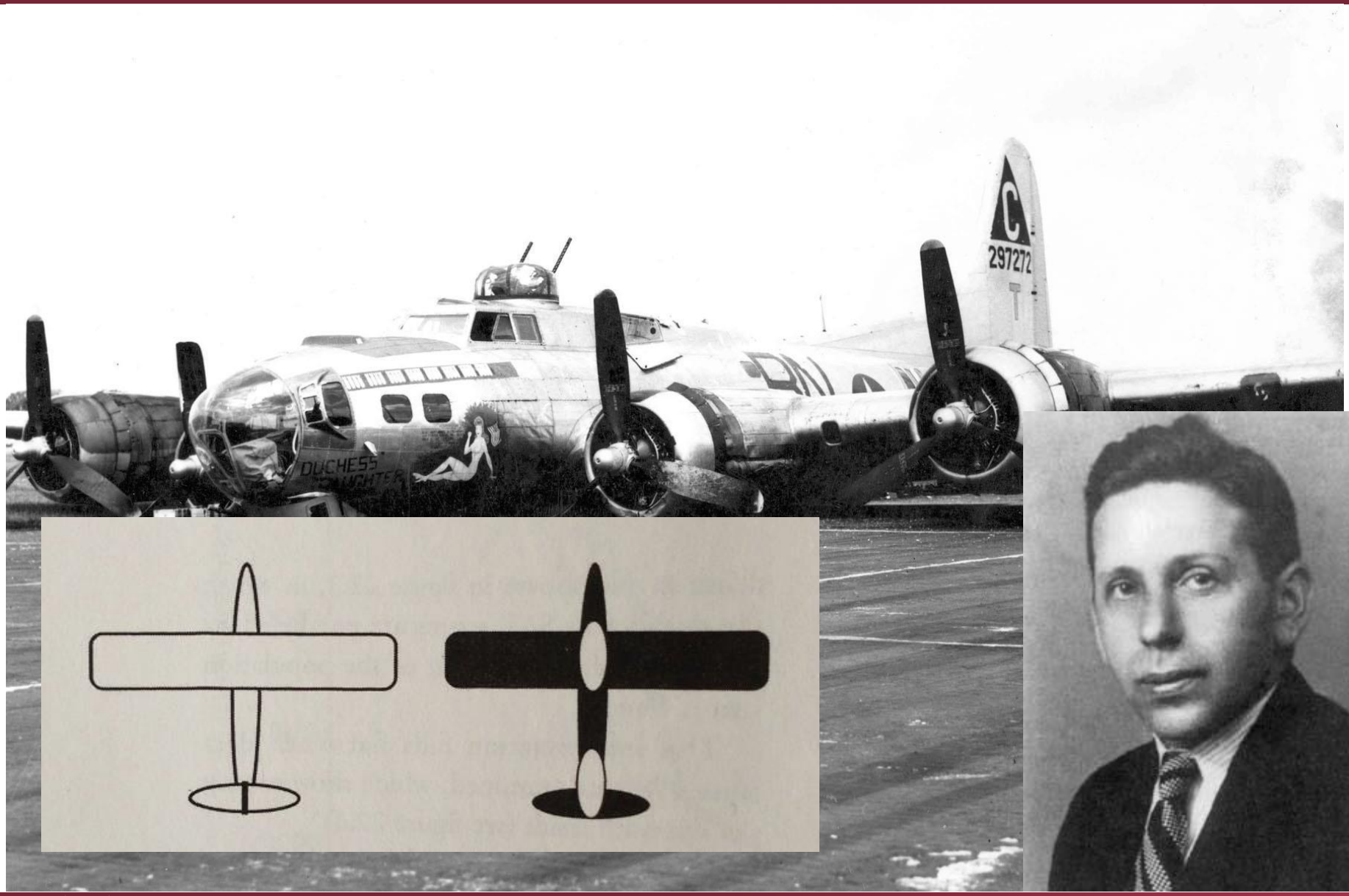
Qualität der Modelle

- Realität: eine offene Welt ↔
  Modell: eine geschlossene Welt

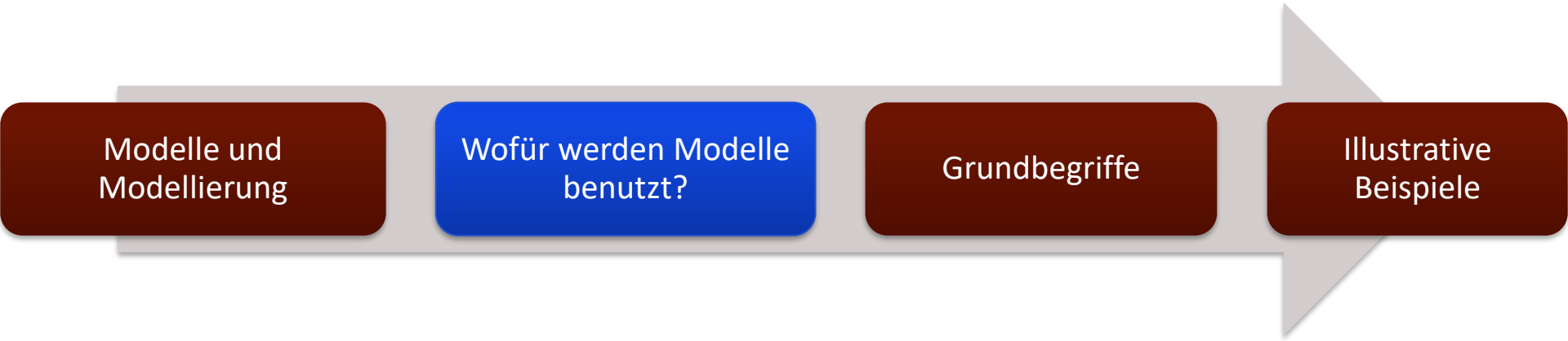
- "Treue" eines Modells:
  - für wahrscheinliche Fälle
  - für kritische Fälle

- Die Umsetzung eines schlechten Modells kann tödlich sein …

Wald Ábrahám



Inhalt



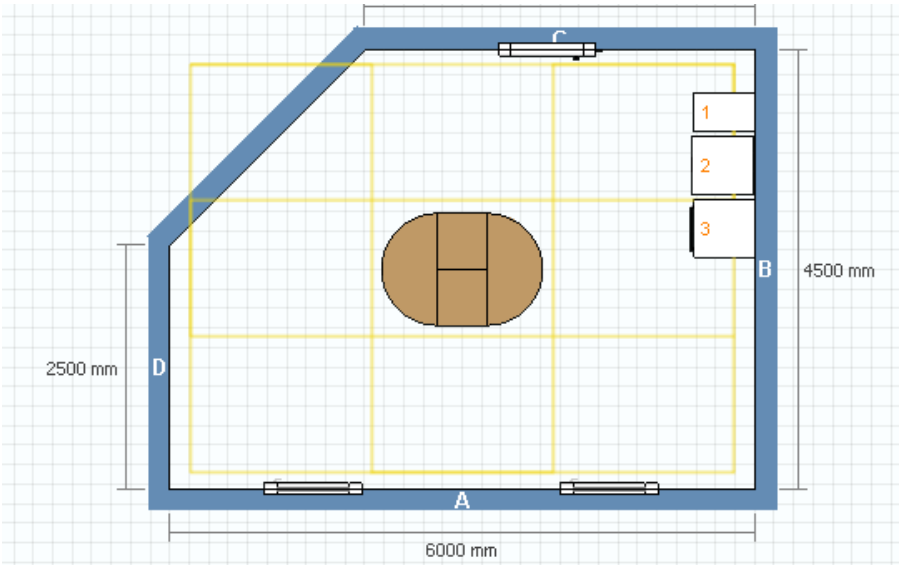
WOFÜR WERDEN MODELLE BENUTZT?

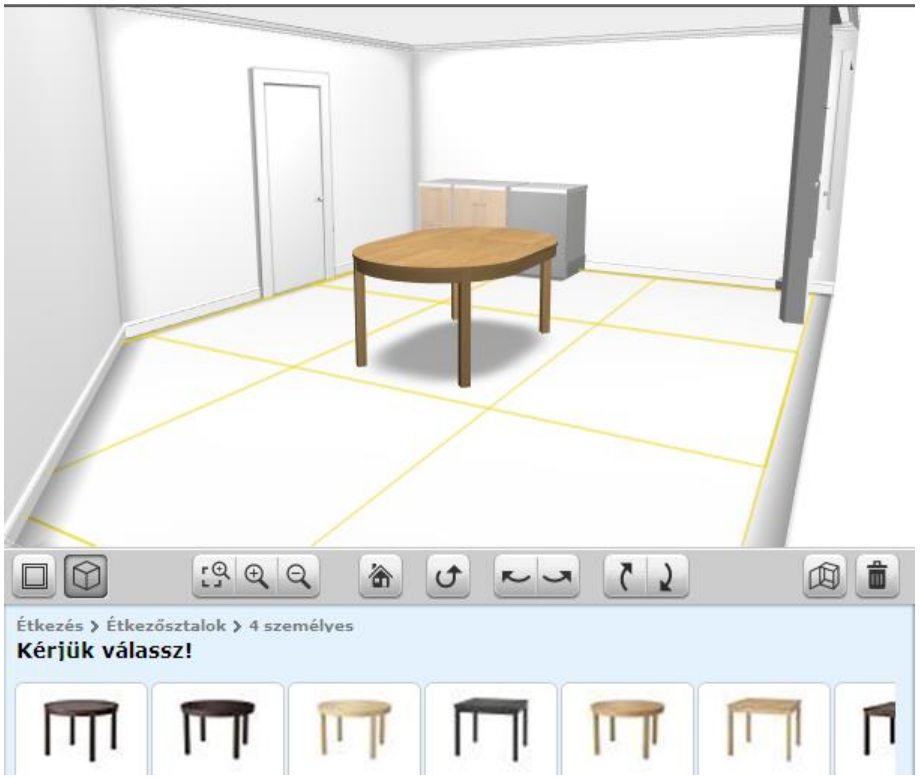
Welchen Sinn hat die Modellierung?

- Ich schreibe Software. Soll ich auch modellieren?
  - Du machst es schon!
    - (Der Quellenkode 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

Modellierung in der praktischen Welt?

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

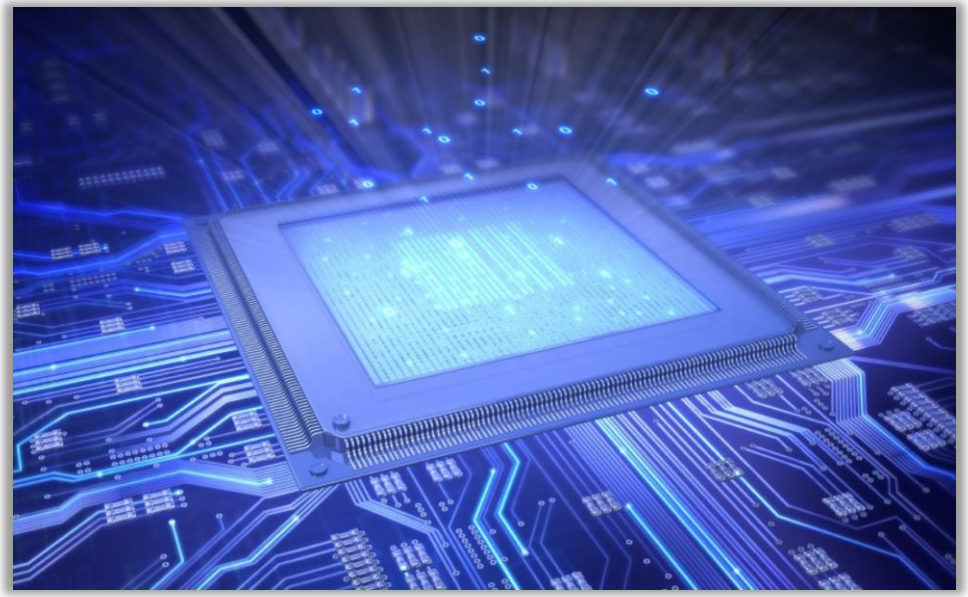




Auch das ist eine Modellierungssprache! ☺

- VHDL, Verilog – Fachgebietspezifische 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
```



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)

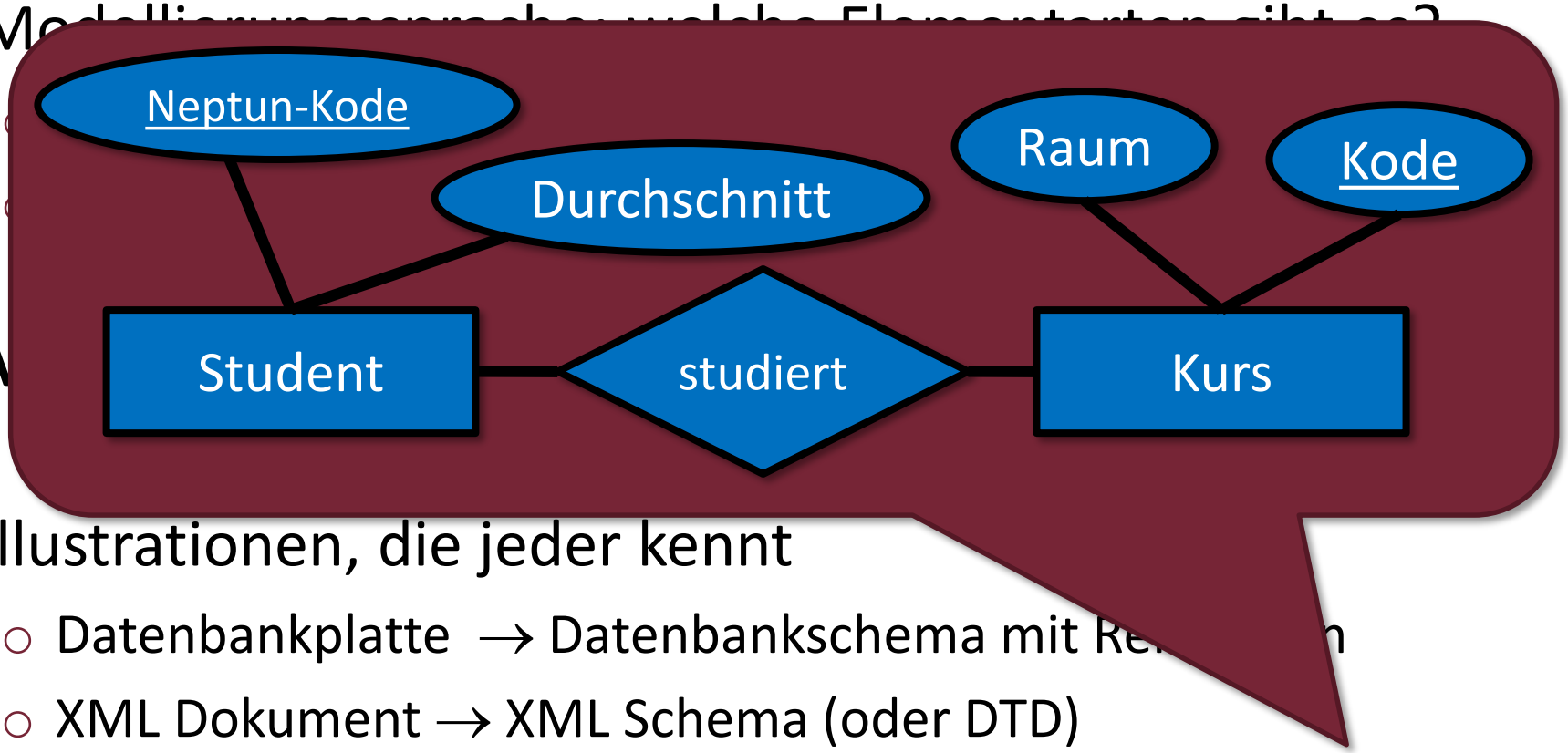
Grundbegriffe - Metamodellierung

- 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

- Mo
  -
  -
- M
- Illustrationen, die jeder kennt
  - Datenbankplatte → Datenbankschema mit Re n
  - XML Dokument → XML Schema (oder DTD)
  - Datenbankschema → Individuum-Verbindung (ER) Modell
  - UML Class Diagramm → Klassendiagramm
  - …



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**

Systemplanung als Prozess

Management

- Besorgung, Versorgung
- Planung, Führung, Bewertung

ANSI/EIA 632 Standard

Ingenieuraufgaben

System-planung

- Definition der Anforderungen
- Definition der Lösung

Produkt Herstellung

- Implementation
- Benutzbarkeit

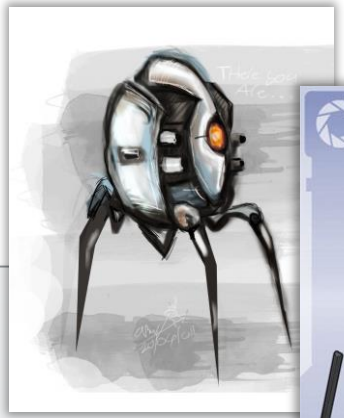
Auswertung

- Systemanalyse
- Anforderungsvalidation
- Systemsverifikation
- Endproduktvalidation

Diese werden typischerweise mit Modellen unterstützt

Systemplanung als Prozess - Analogie

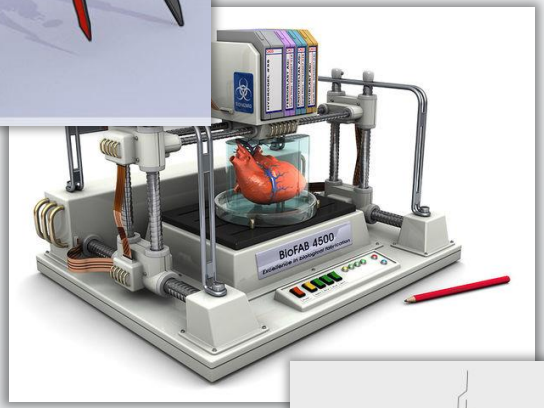
Management



PORTAL 2

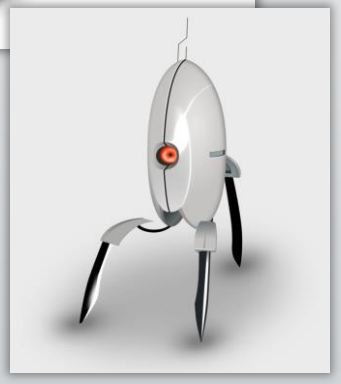
System-
planung

ngenieuraufgaben



Herstellung
der Produkt

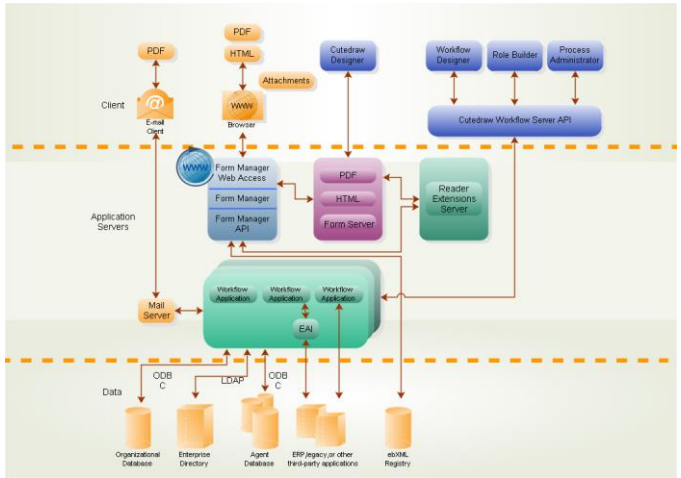
Auswertung



Validation: Bauen wir das richtige Produkt?
Verifikation: Bauen wir das Produkt richtig?

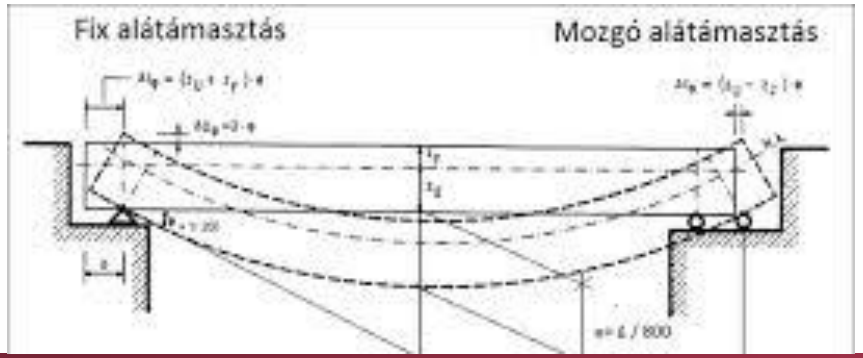
Anwendung – Dokumentation

- 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”



Anwendung – Analyse

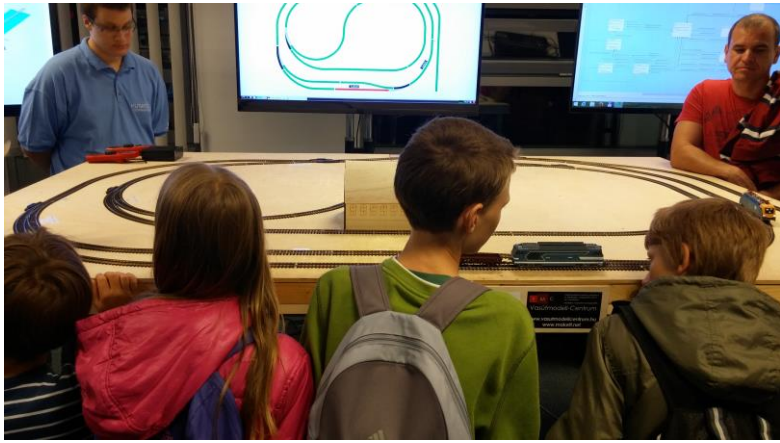
- Manuell oder (teilweise) automatisiert
- Methode
  - Oberflächliche, statische Analyse
  - Mit dynamischer Zustandsraumexploration – Modellprüfung (model checking)
  - Mit Beweis von formalen Aussagen
- Ziel
  - Untersuchung, Fehlersuche (best effort)
  - Bestätigung von Dienstleistungszuverlässlich-keitskriterien (stärker!)
  - Charakteristiken rechnen/planen (z.B. Zeitplanerstellung)



Anwendung – Ableitung

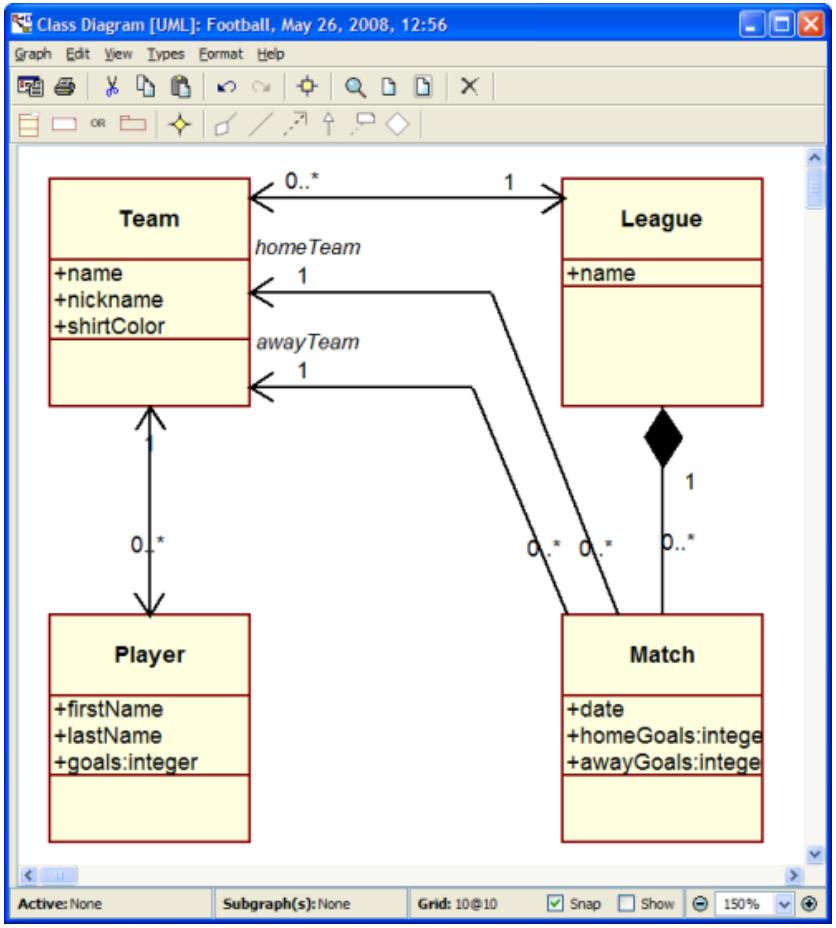
- Manuell oder (teilweise) automatisiert
- Ergebnis
  - Generierung von Programmkode, analysierbarer Sprache, usw.
  - Anderes Modell
    - Verfeinerung, nächste Entwurfsphase
    - Teilaspekte
    - Integration der Modelle
- Es kann eigenschaftserhaltend sein

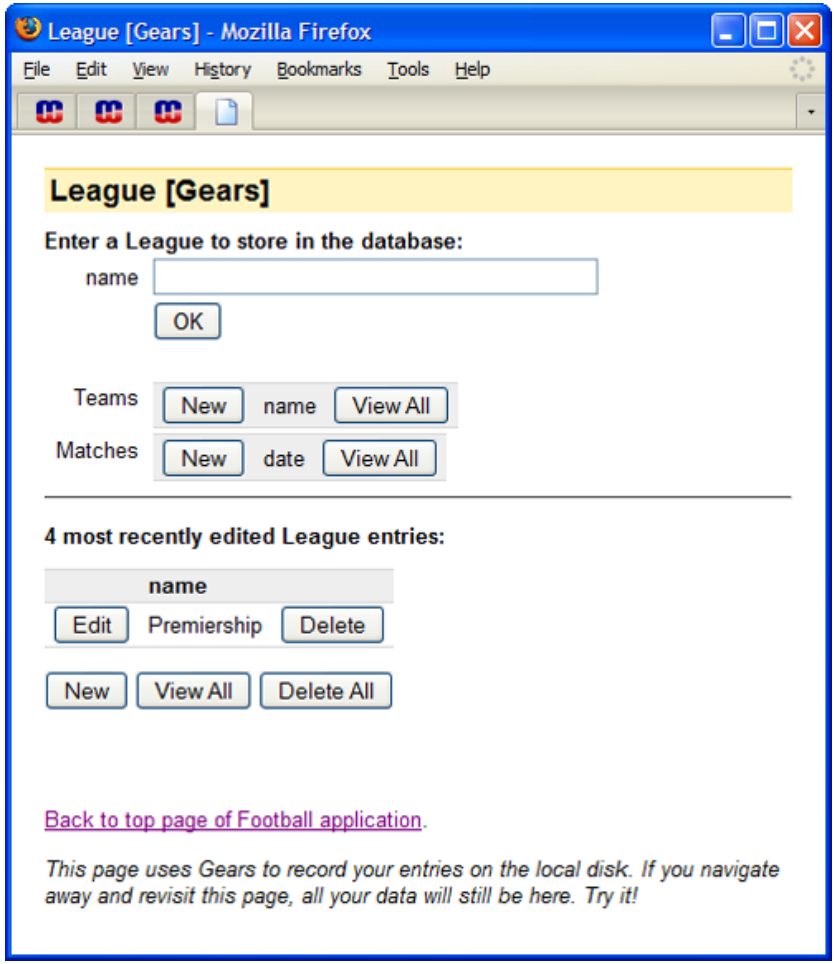
Anwendung – Simulation



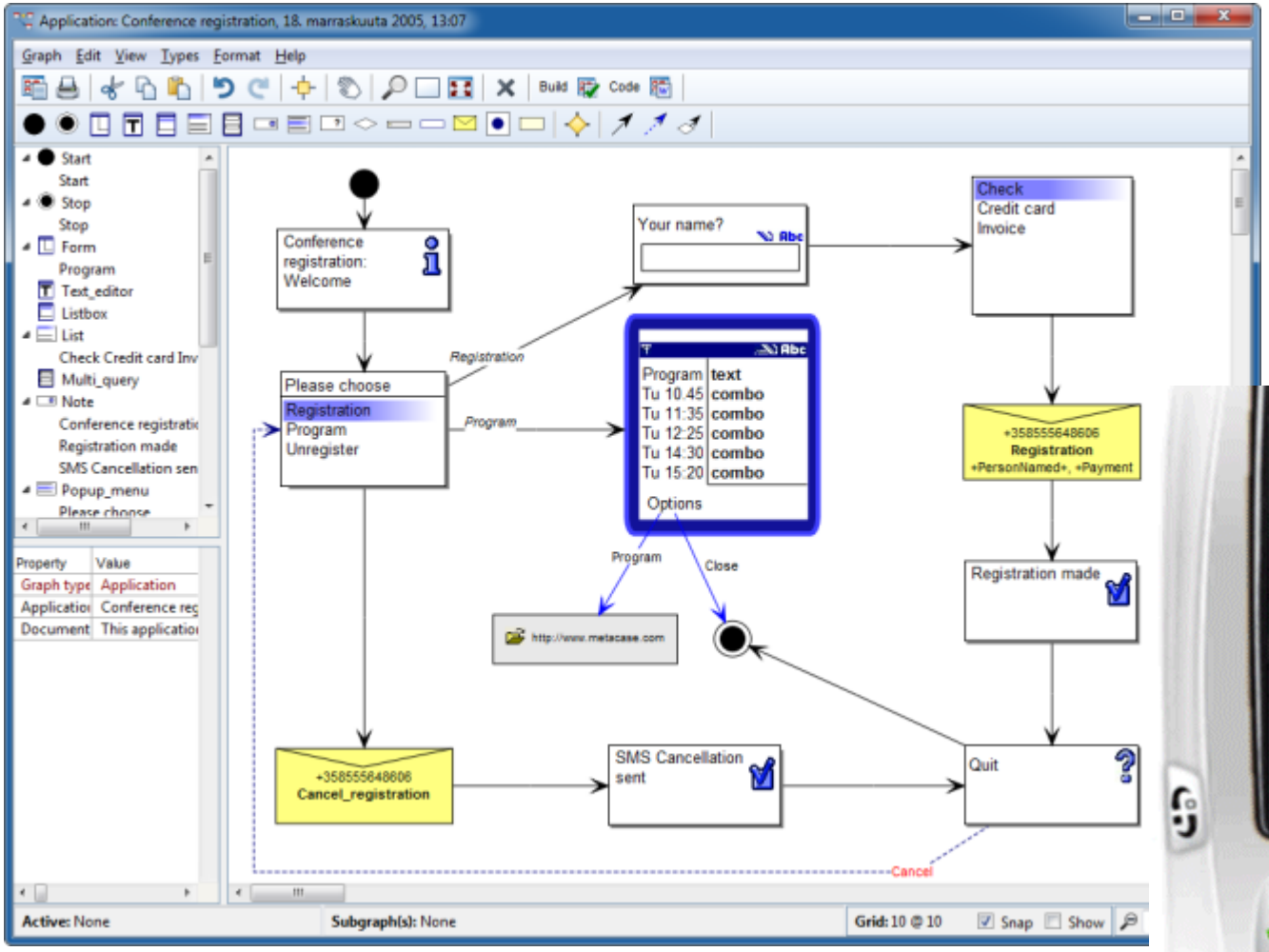
- Demonstration
  - Als Mittel der Kommunikation
- Validation
  - „Ich habe es richtig erbaut …, aber habe ich das Richtige gebaut?”
- Experimente
  - Überprüfung von bestimmten erwünschten Eigenschaften
  - Messung von quantitativen Eigenschaften
  - Ersatz für in der Wirklichkeit kostspielige Versuche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

Entwicklung einer Webanwendung

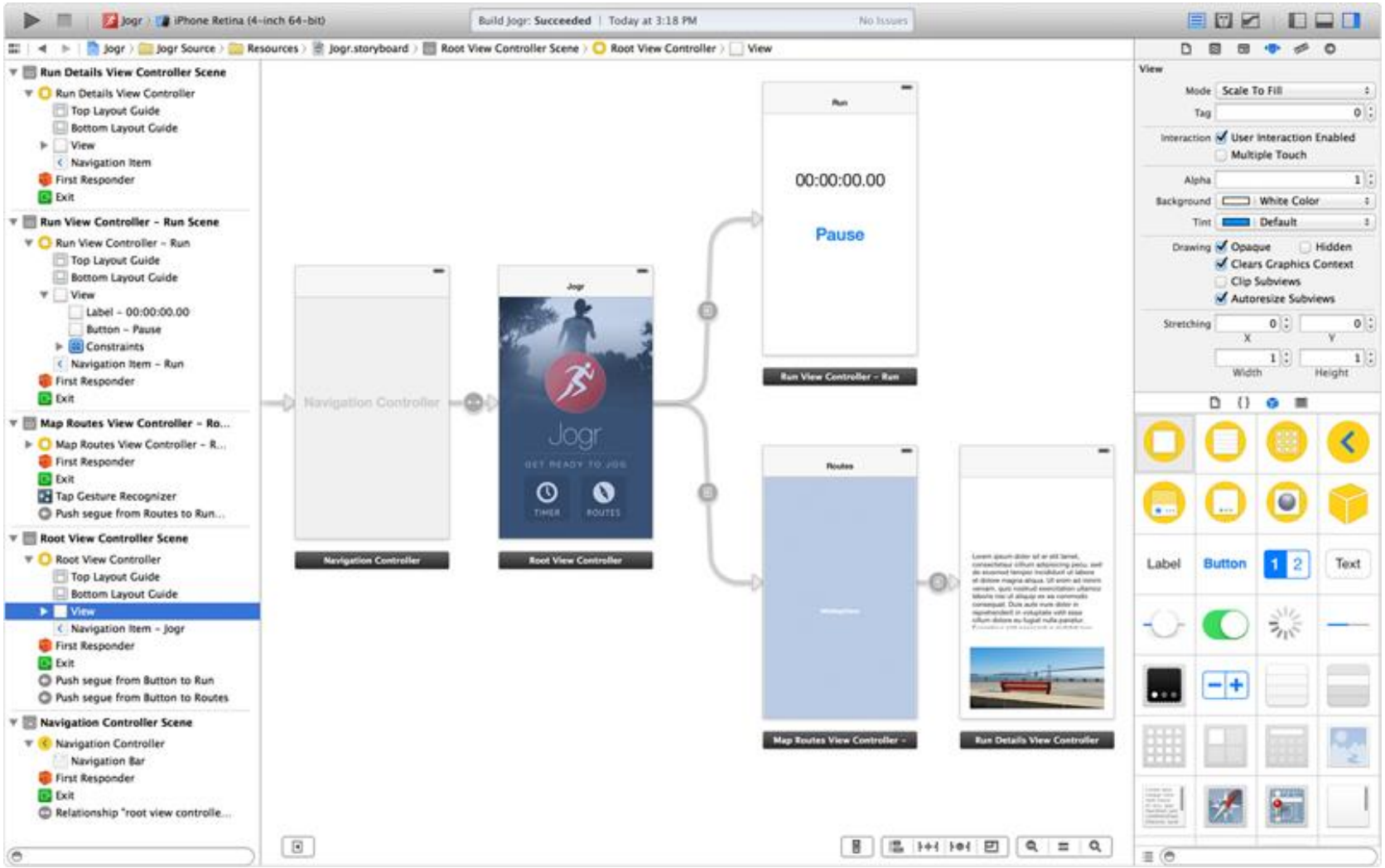




Entwicklung einer Smartphone-Anwendung

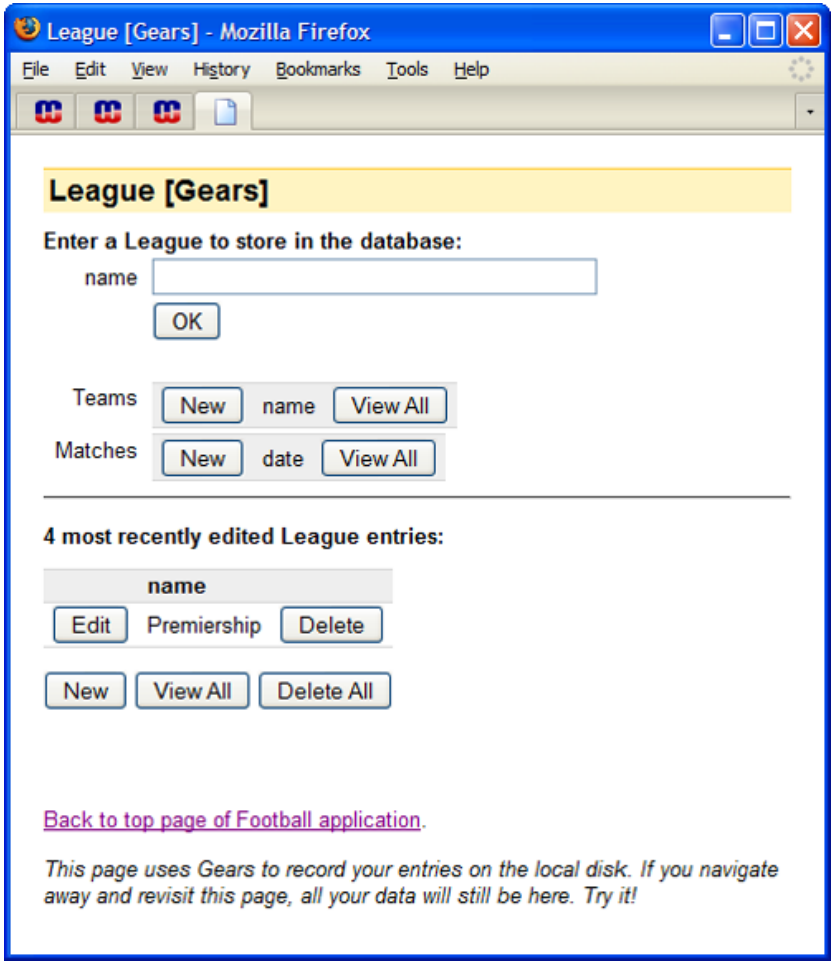


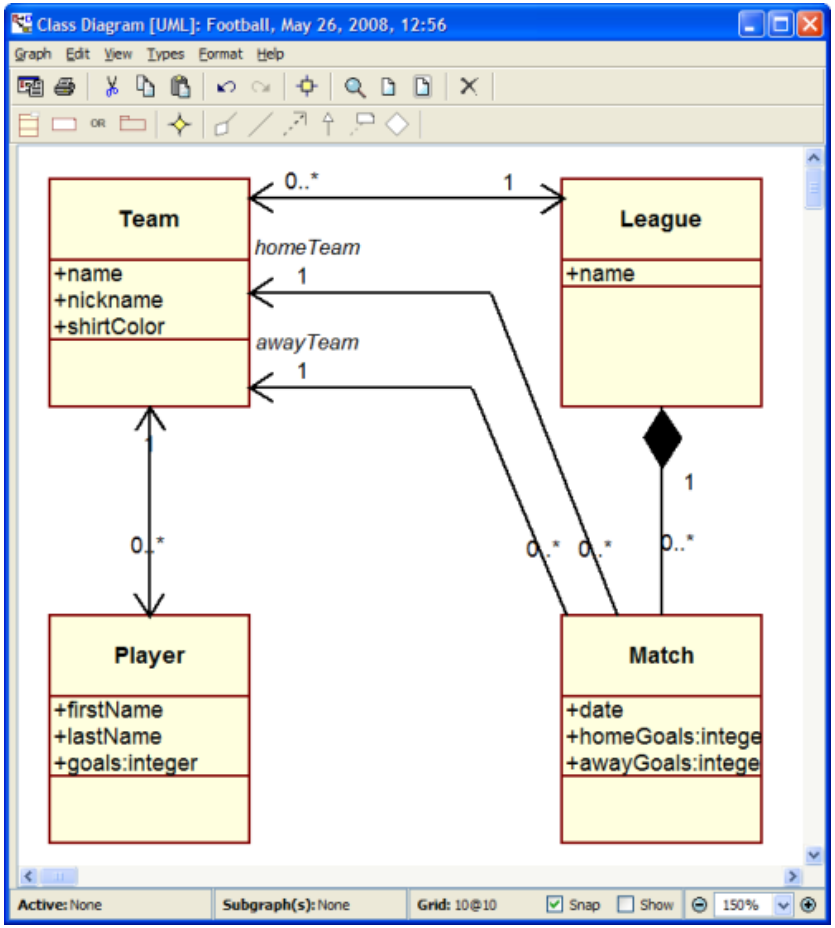
Entwicklung einer Smartphone-Anwendung



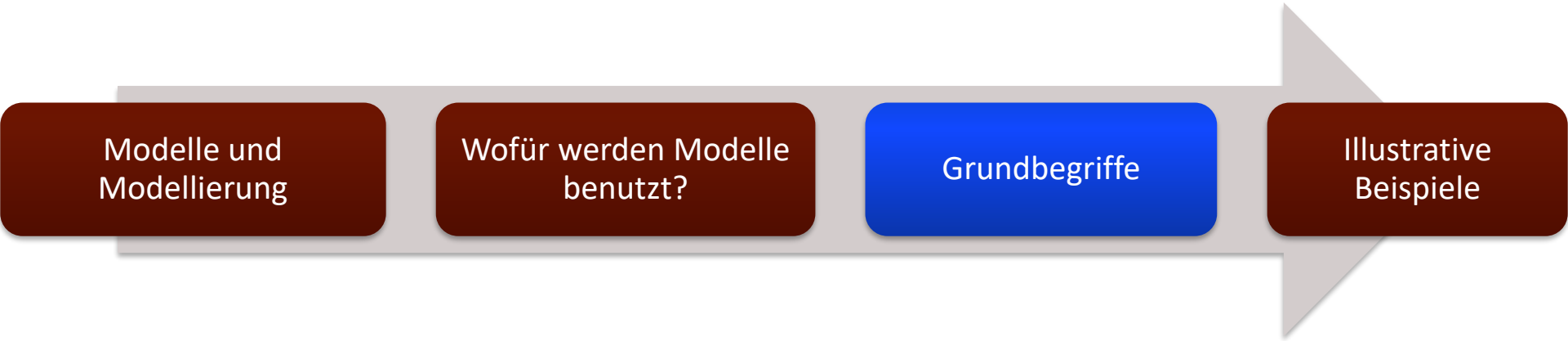
Android Story Board

Reverse Engineering



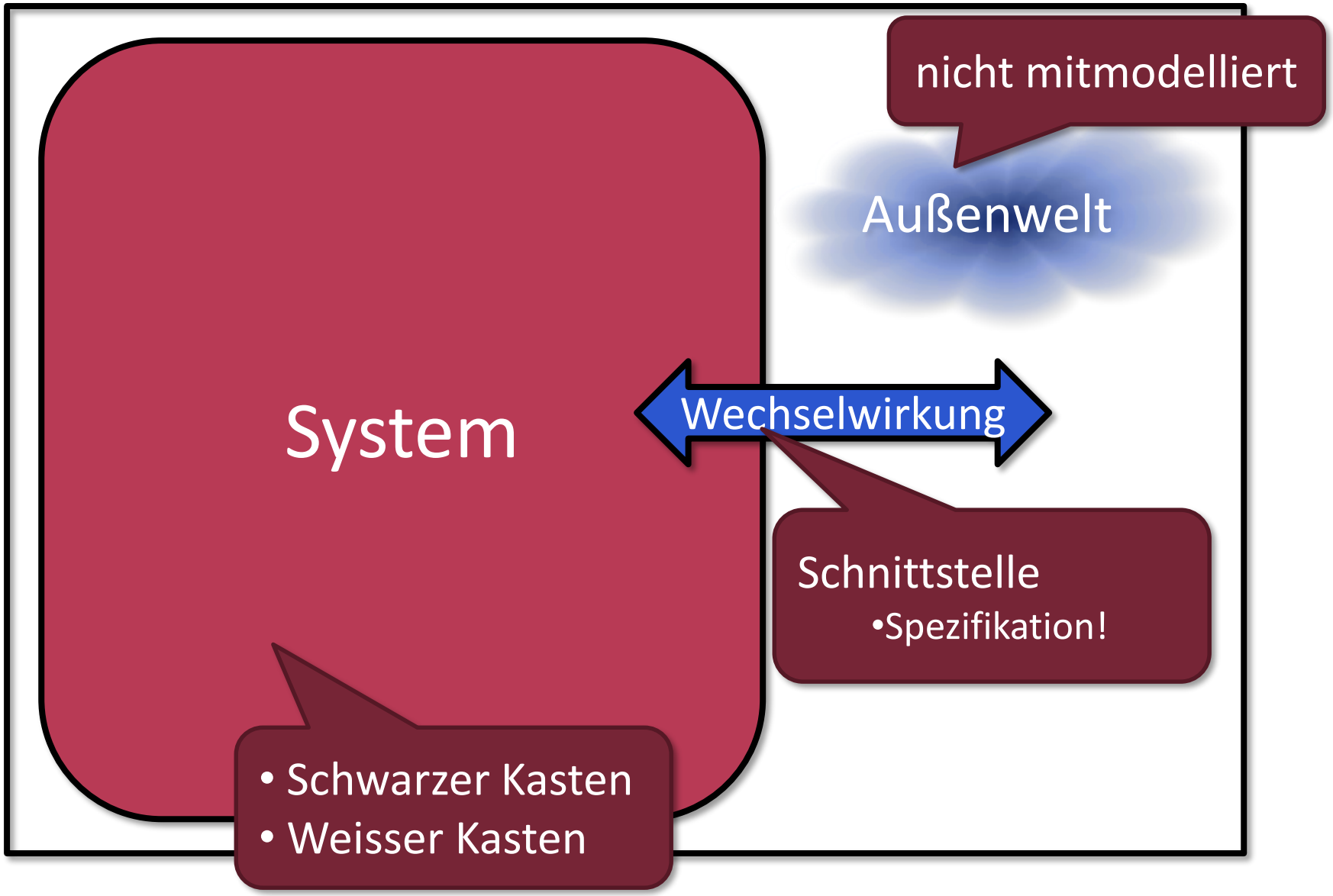


Inhalt

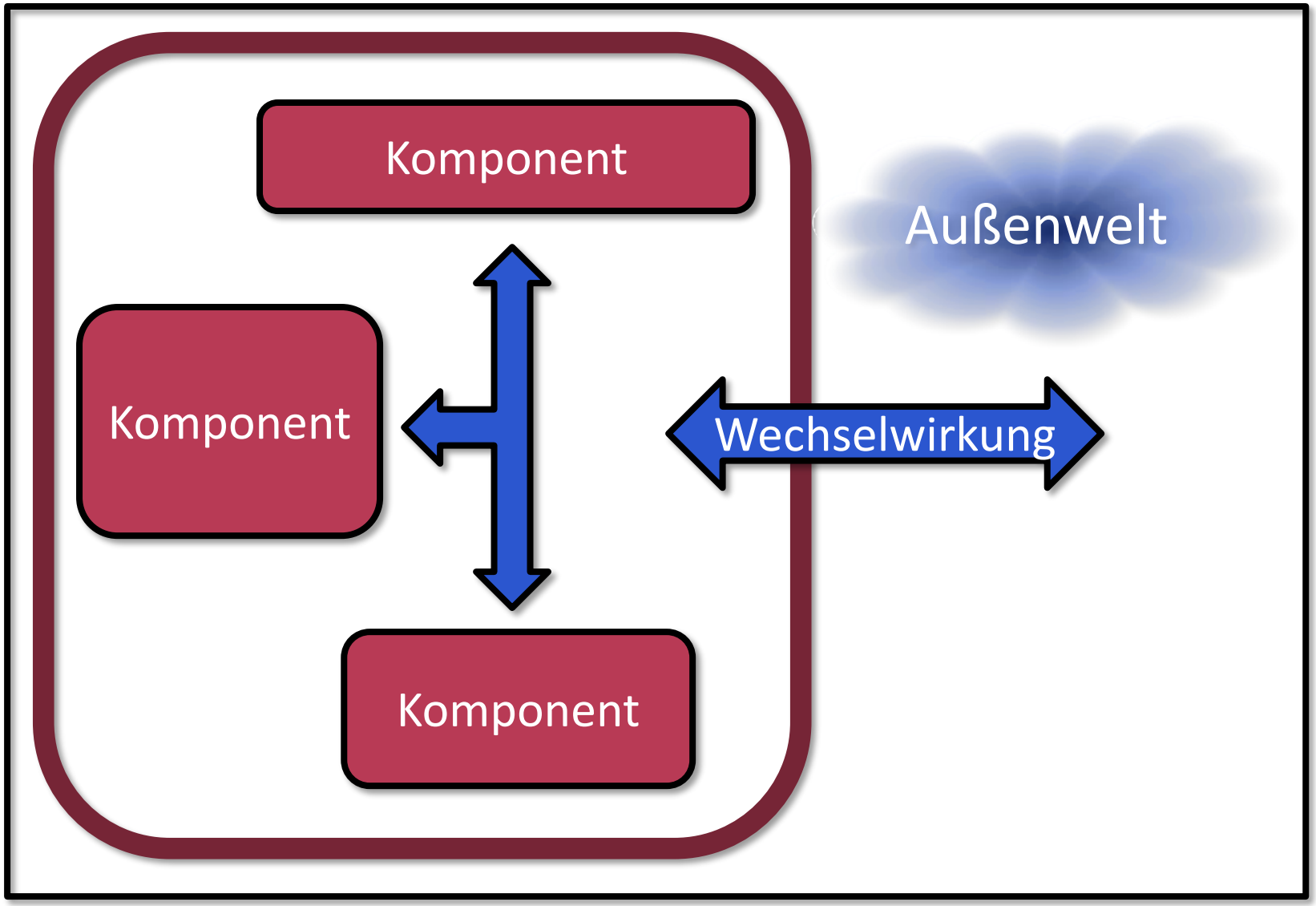


Grundbegriffe der Modellierung

Grundbegriffe – System und Außenwelt



Grundbegriffe – System und Außenwelt

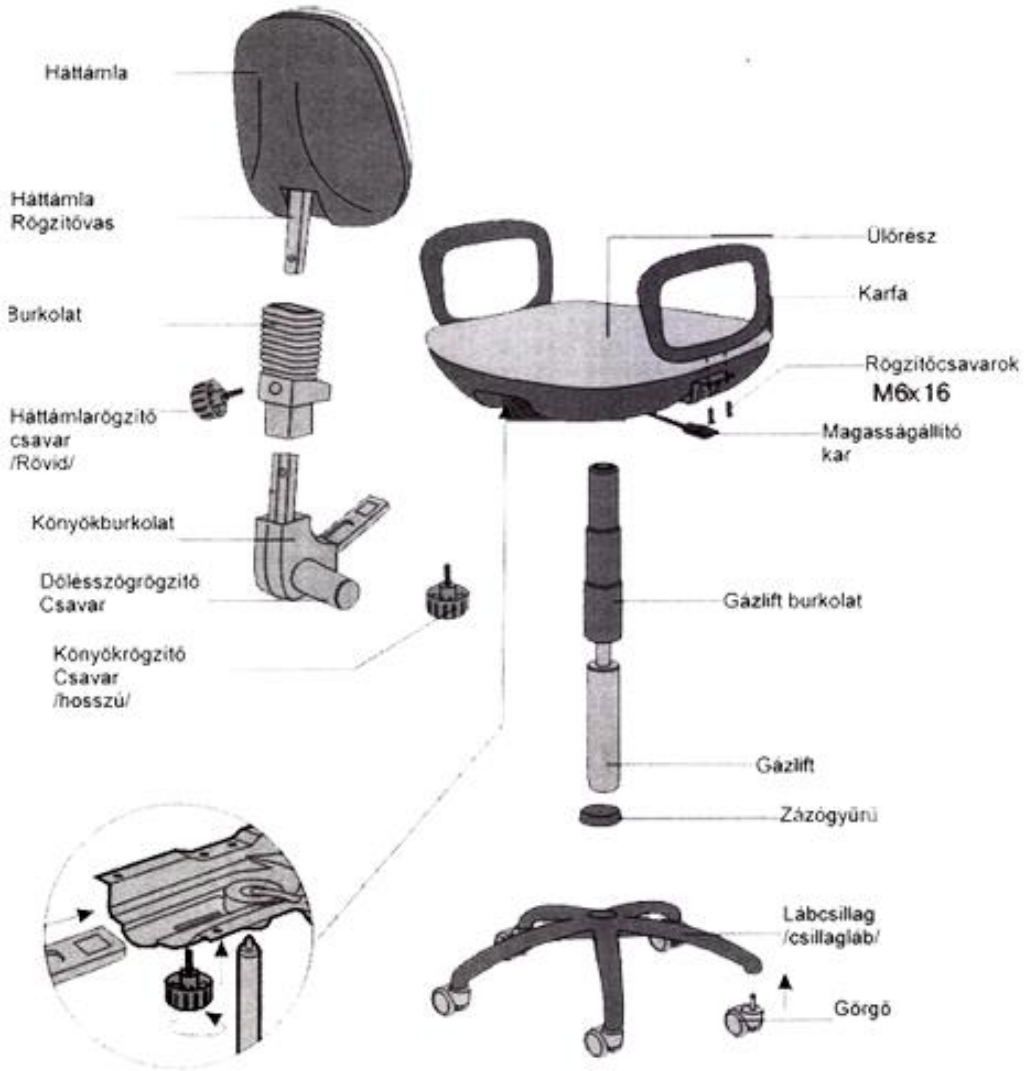


Grundbegriffe – Verfeinerung/Abstraktion

- *Verfeinerung*:
  Bereicherung des Modells mit Einzelheiten
  - … so dass die originale Modellabstraktion erhalten 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**

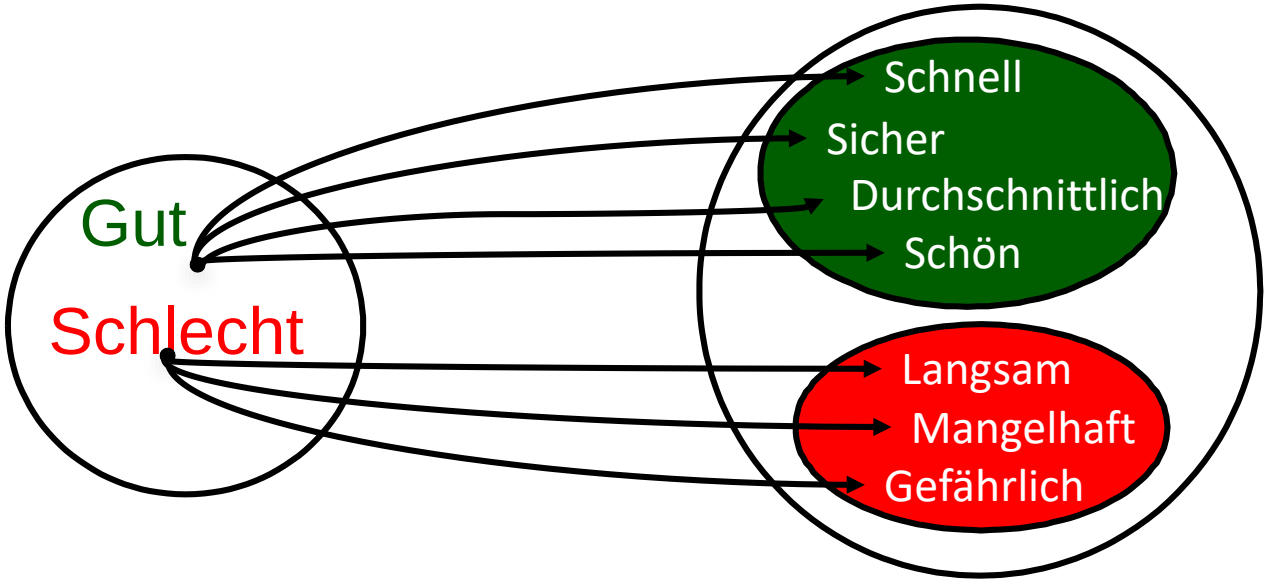
Grundbegriffe – Verfeinerung

Grundbegriffe – Verfeinerung



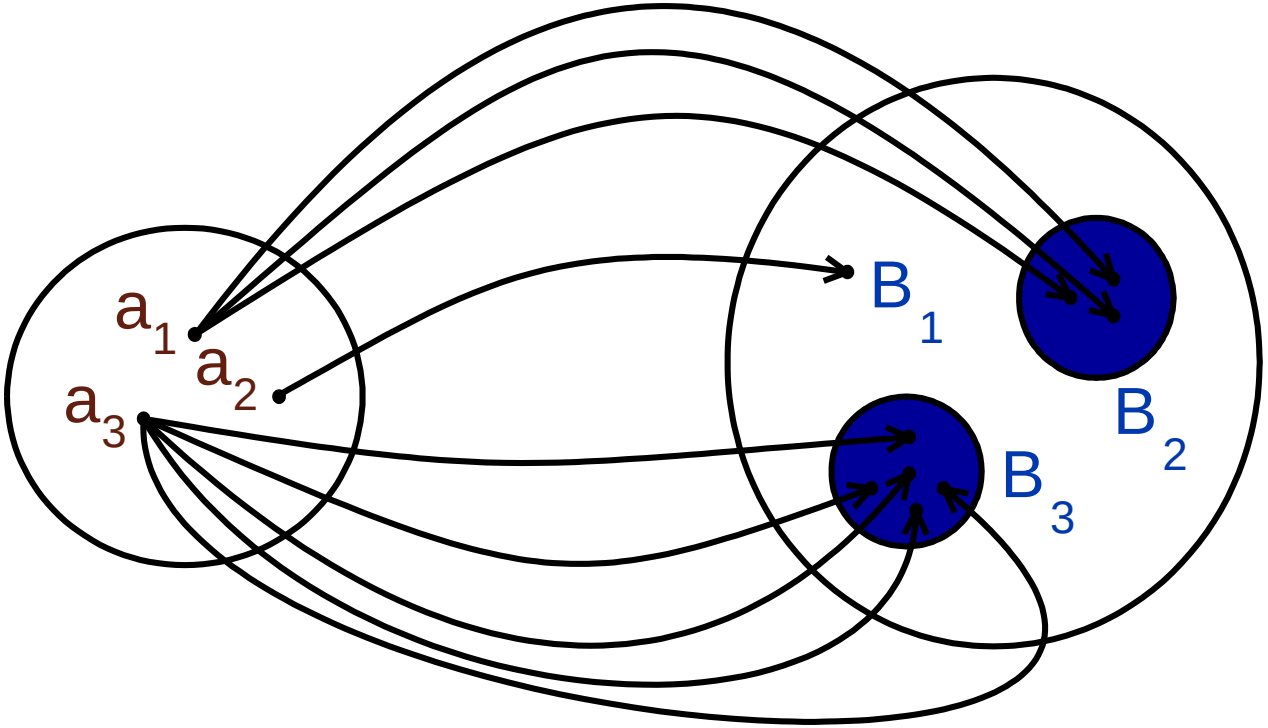
Mengenverfeinerung

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 $R(a_i) \cap R(a_j) = \varnothing \ \forall i, j$

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 Kode 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 Kode noch wiederverwendet werden?

→ Vermeidung der Kodeduplikation

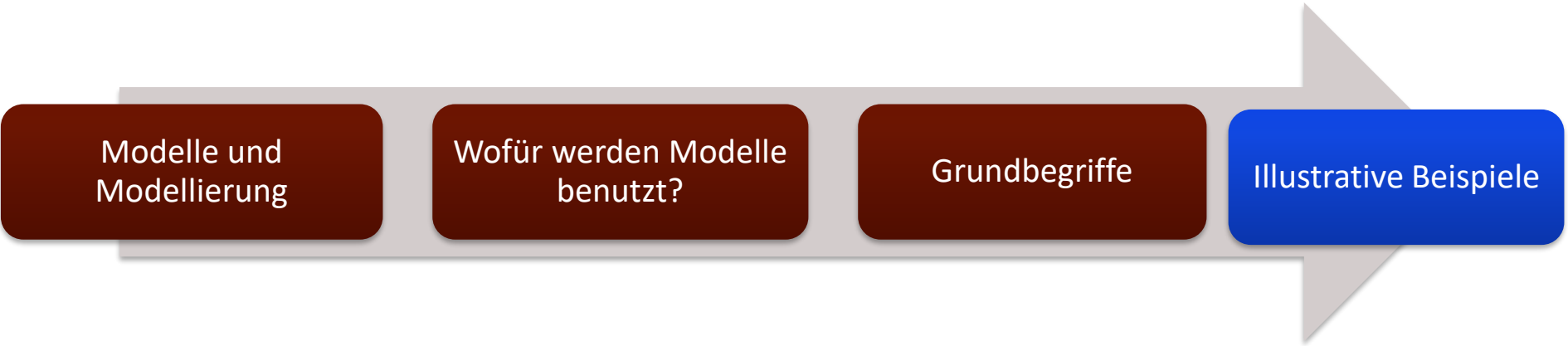
Rolle der Abstraktion in Programmierung

- Anmerkung:
  - Ordnungsalgorithmus mit Auswählen: eine Abstraktion
  - Verschiedene Vergleichfunktionen (→ 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*: Leistungmodellierung
  - *Rechnernetzwerke*: Simulation
  - *Künstliche Intelligenz*: Ontologien
  - *Systemtheorie*: Simulink Kontrollelemente

Inhalt



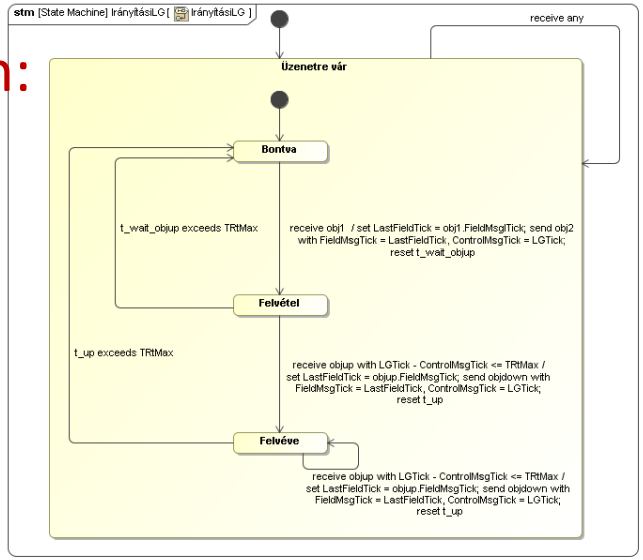
Illustration

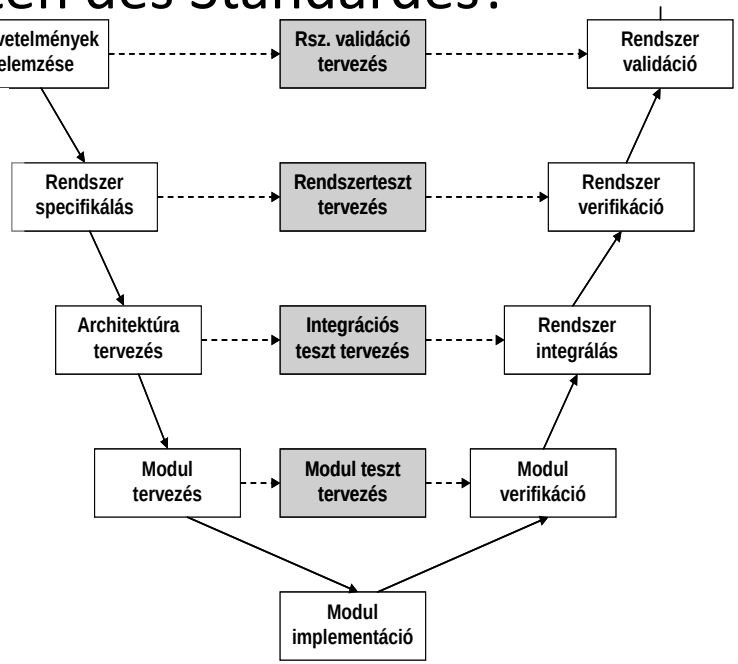
Entwicklung von Bahnsoftware

- 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:

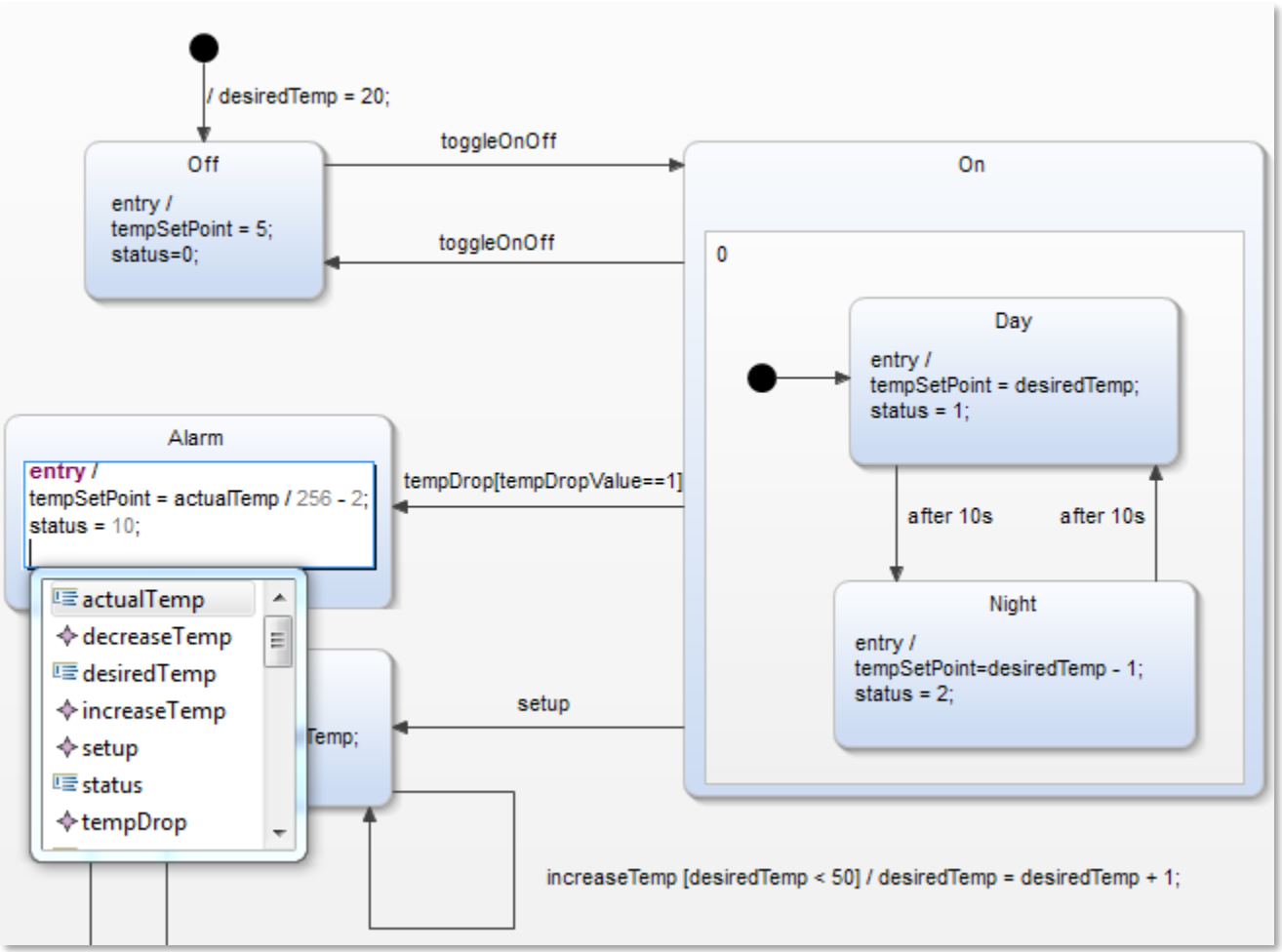
- systematische Überprüfung
- Modellierung und formale Verifikation



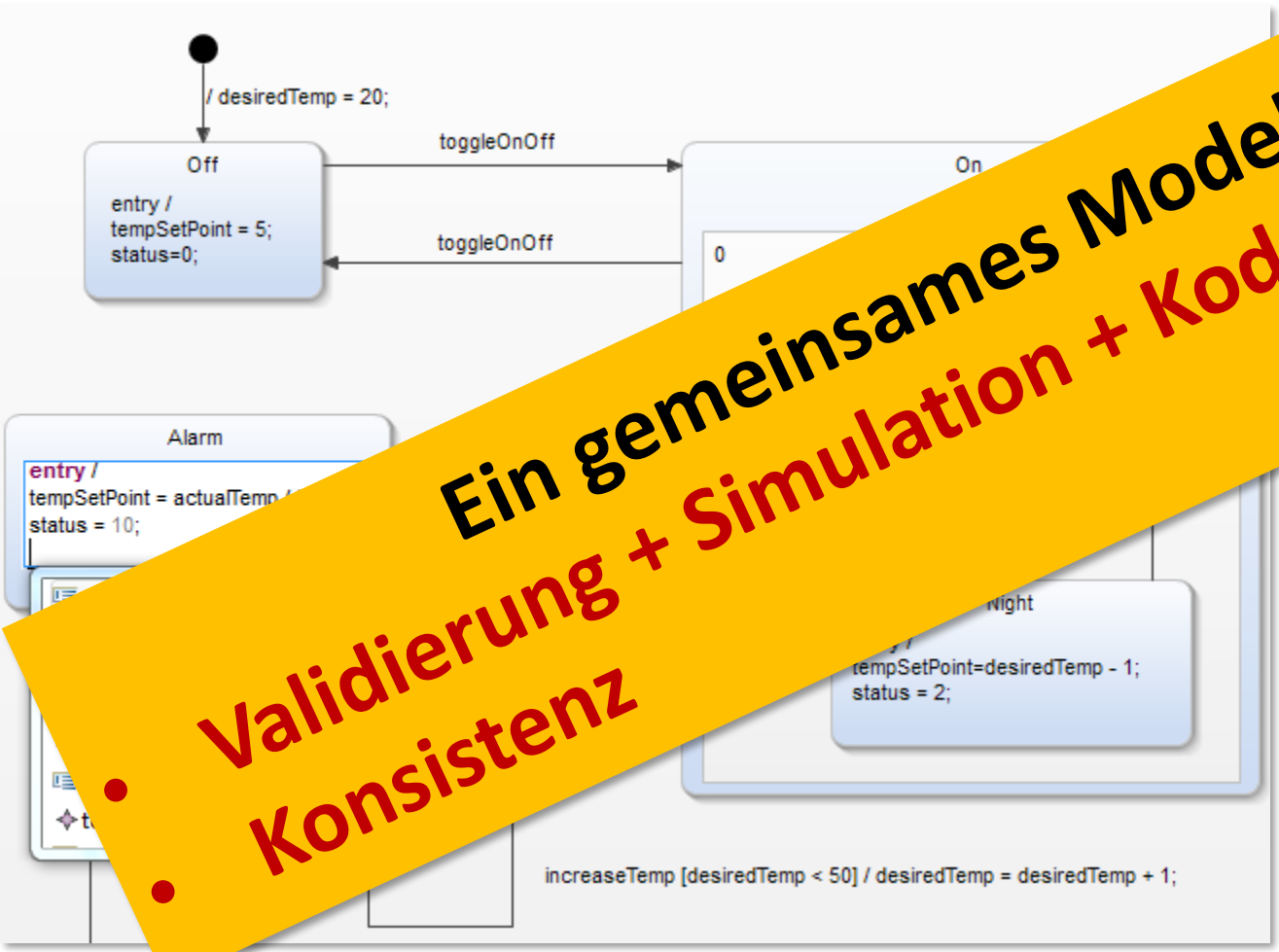


Prozess→ Modell → Qualitätssicherung

Yakindu - Zustandsdiagramm



Yakindu - Zustandsdiagramm



Ein gemeinsames Modell

- **Validierung + Simulation + Kodegenerierung**
- **Konsistenz**