HS Augsburg
Fak. Informatik

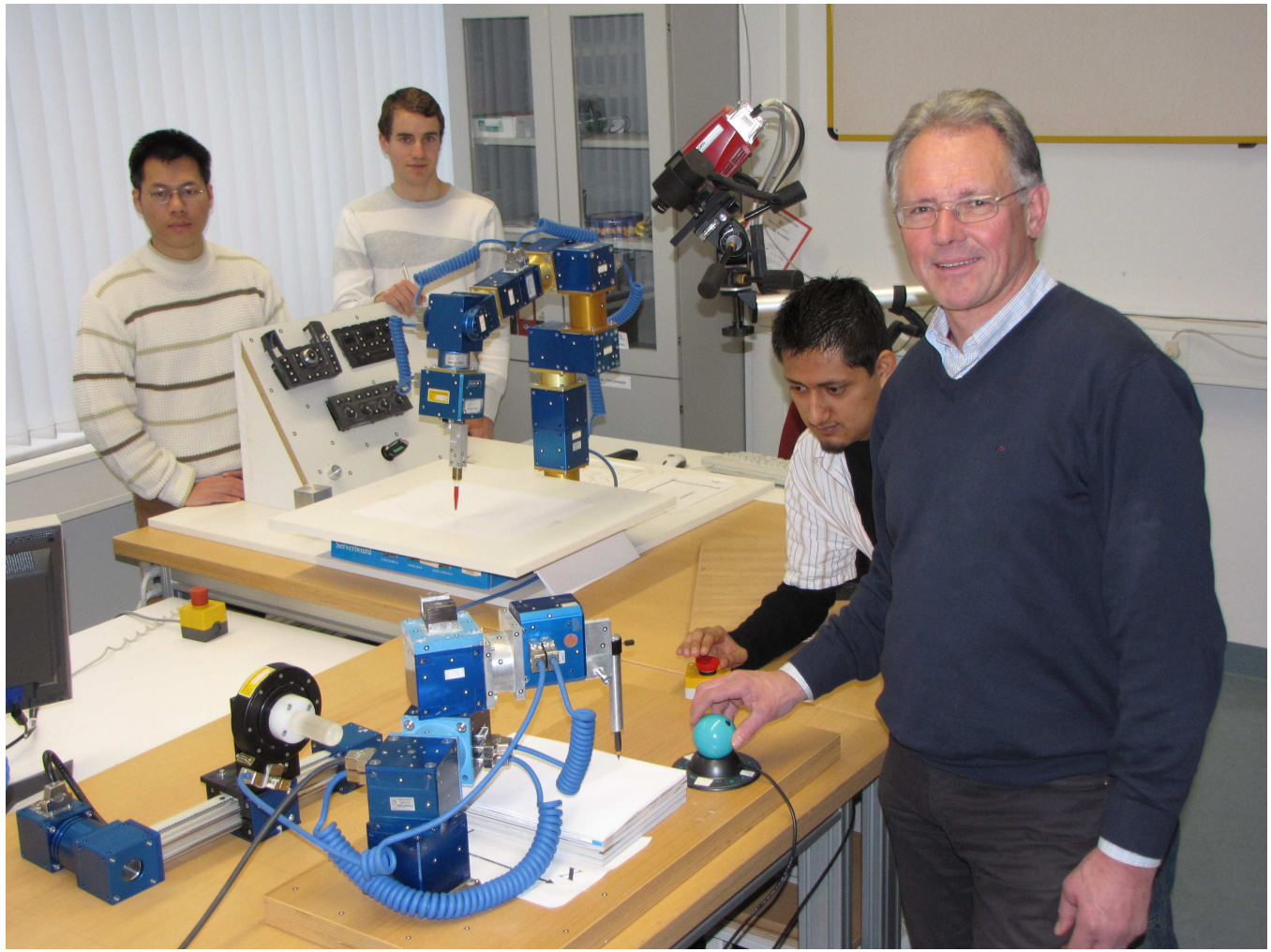
Modellbasierte und komponentenorientierte Softwareentwicklung für Eingebettete Systeme

CIM & Robotik
Prof. G. Stark

1. Vorstellung Labor für CIM & Robotik
2. Zielkriterien für die Softwareentwicklung
3. Programmierparadigmen und Entwicklungsprozess
4. Modellbasierte Entwicklung
5. Komponentenorientierte Entwicklung
6. Anwendungsbeispiele
7. Ausblick auf neue Verfahren und Technologien

Hochschule Augsburg, Labor für CIM & Robotik, Prof. Georg Stark
Email: Georg.Stark@hs-augsburg.de

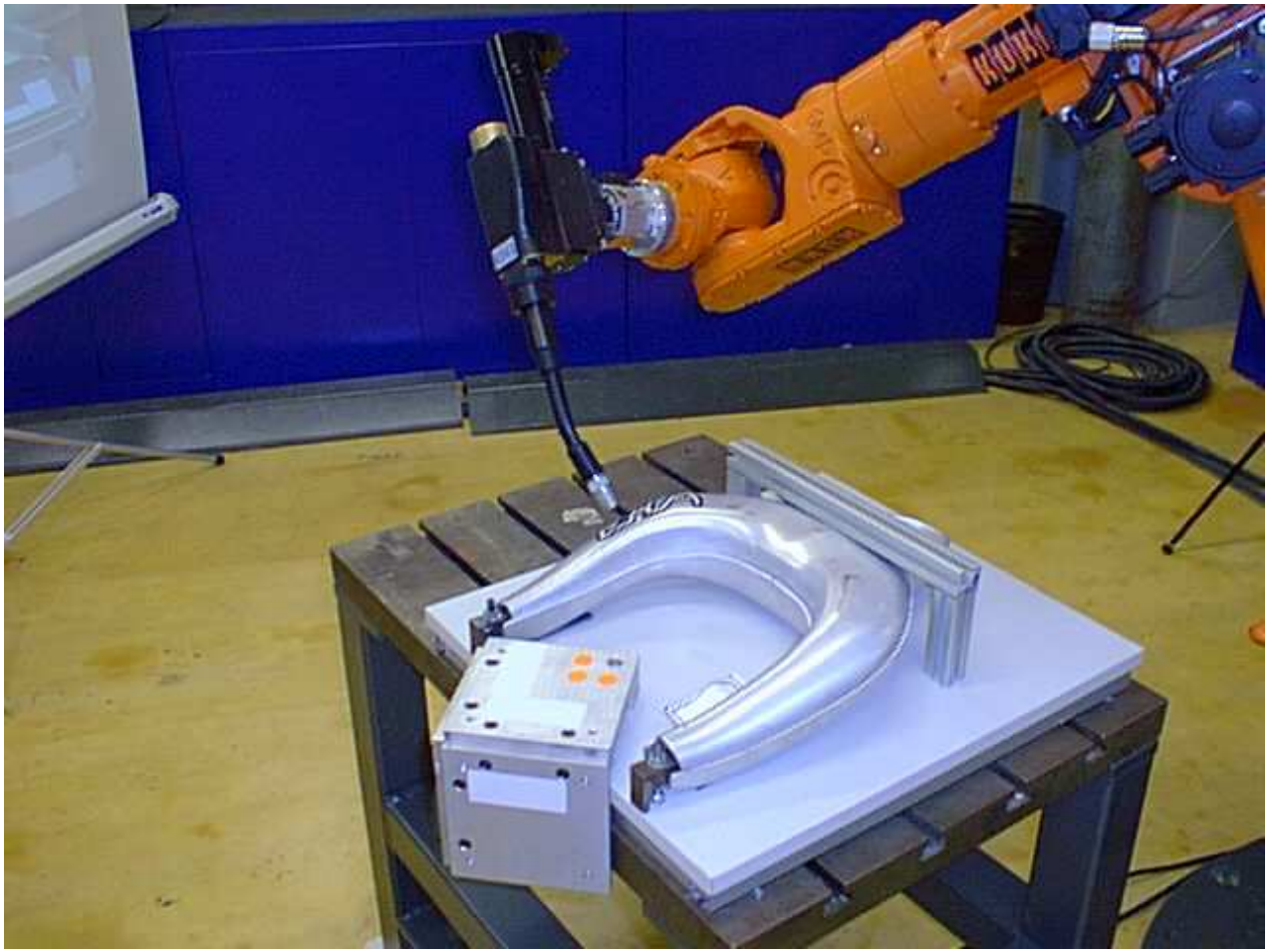
Vorstellung Labor CIM & Robotik



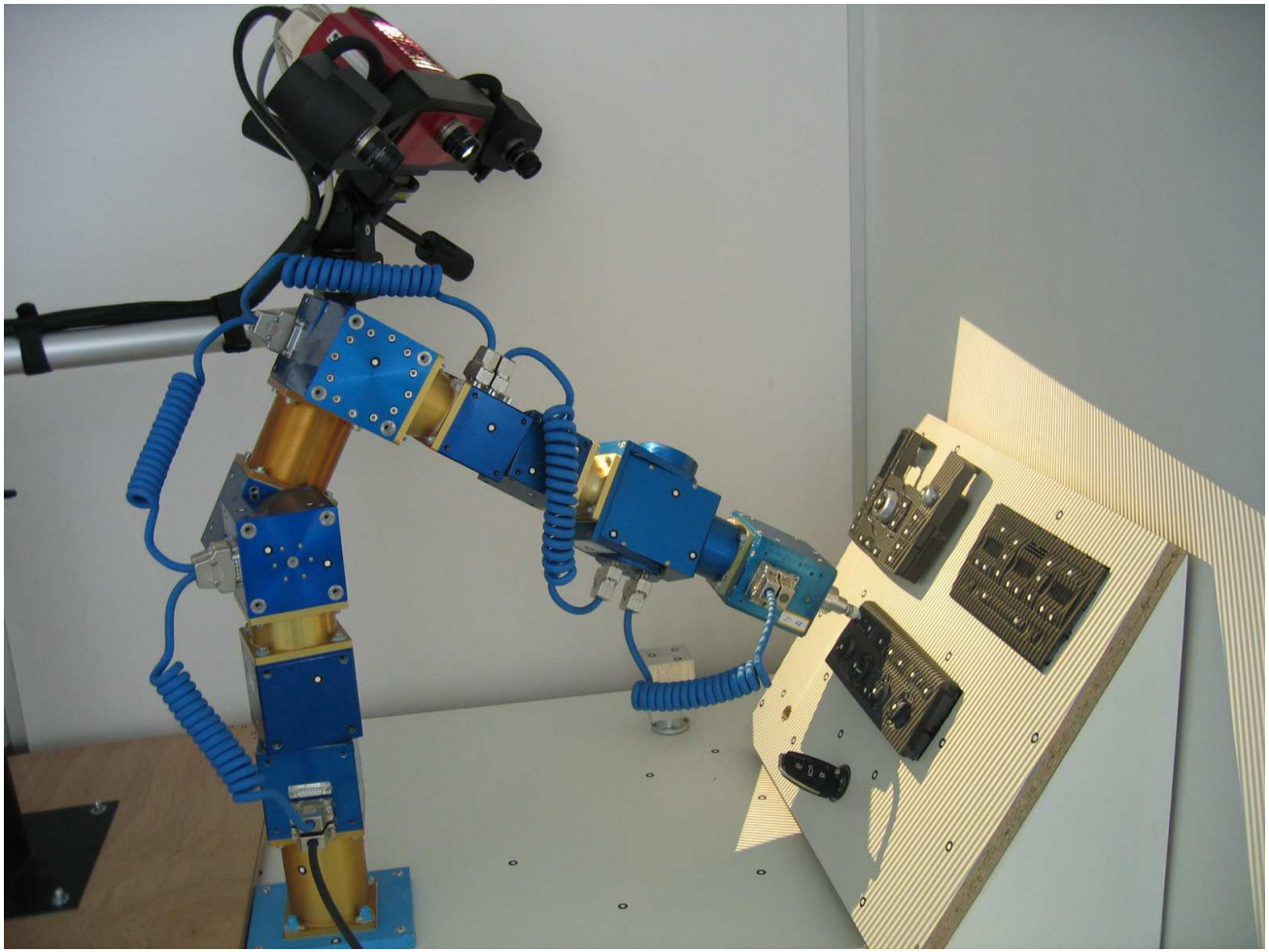
Schwerpunkte F & E

- **Praktische Robotik**
  - Neue Programmierverfahren
  - Weiterentwicklung eigene Software MRobot
  - Sensorbasierte Steuerungsfunktionen
  - Kooperierende Systeme
- **Angewandte Robotik**
  - Mobile Roboter
  - Assistenzroboter für die Fertigung
- **3D-Bildverarbeitung**
  - ATOS-Kamerasystem, Fa. GOM

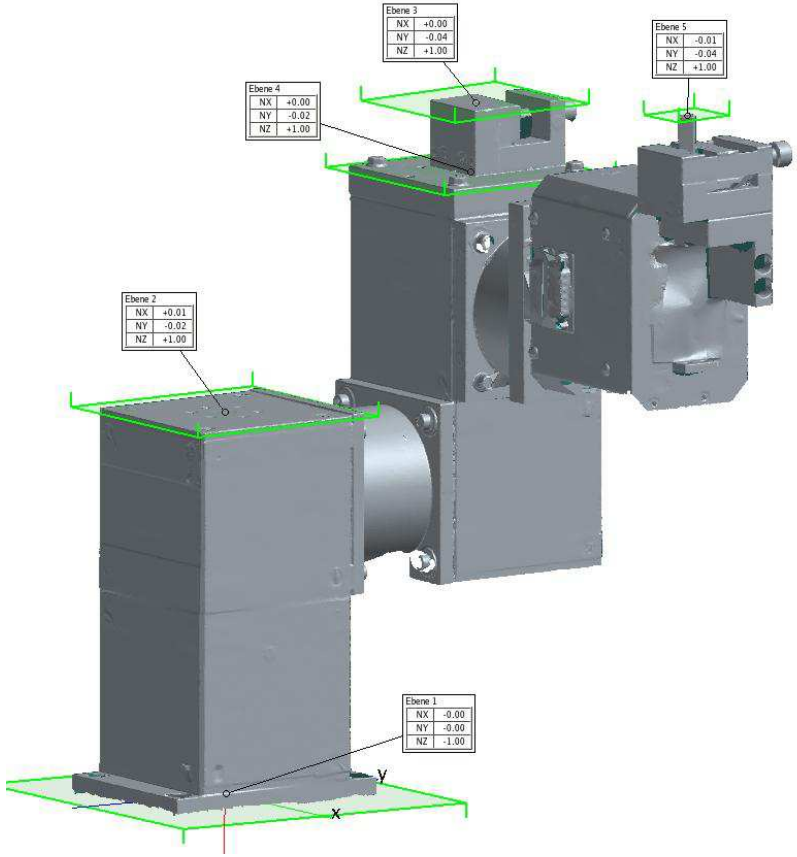
Programmierung mit 3D-Bilddaten

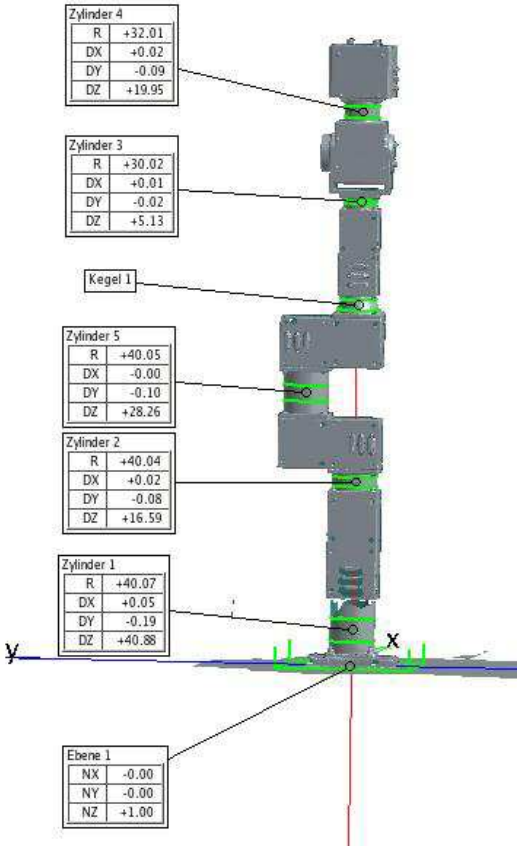


Automatischer Test von Bedieneinrichtungen

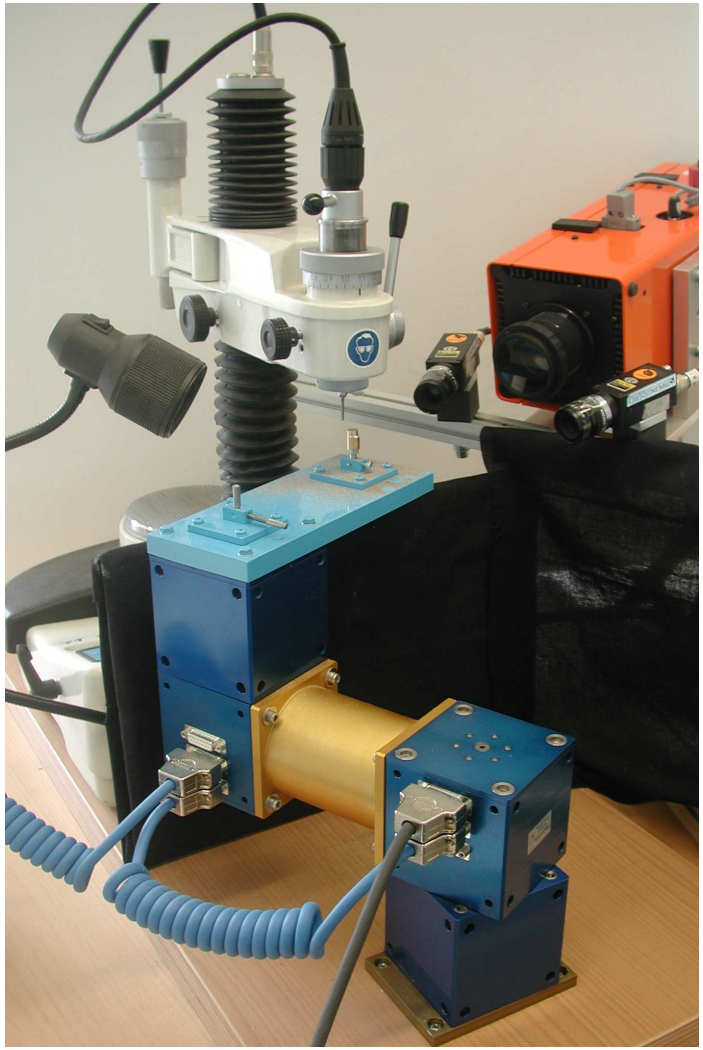


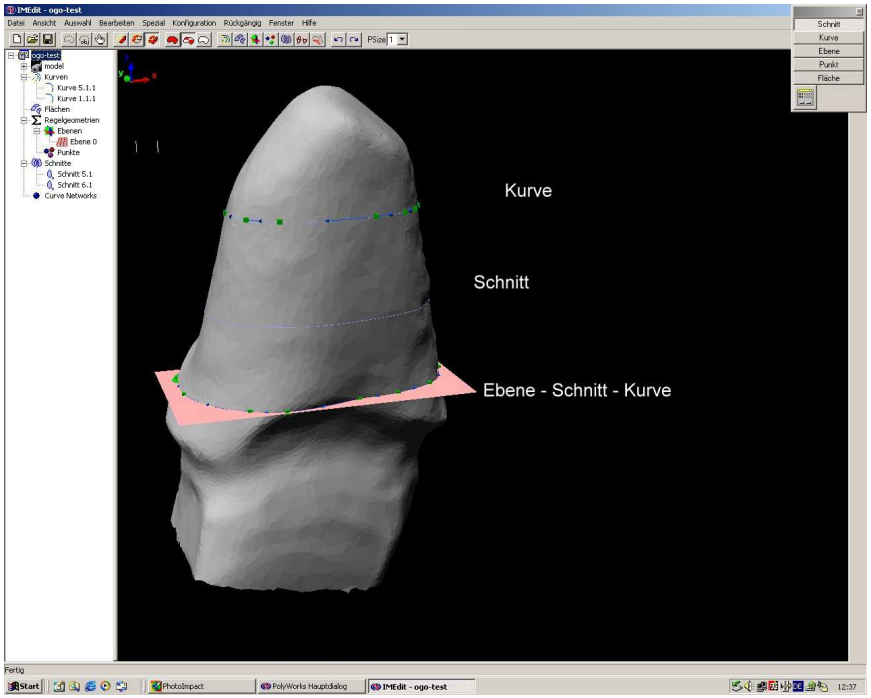
Vermessung mit 3D-Bilddaten





Fräsen von Zahnkronen





3D-Bildverarbeitung: Olympia-Kajak

Messepräsentation mit Mathworks

Forschungsprojekt ECHORD-KANMAN

- **Greiferkamera mit eigener Software *MRobot***
  Andocken und Objekterkennung
- **Kanban-Fertigungsprozess**
  Unterstützung durch mobile Robotersysteme

Allgemeine Zielkriterien für Softwareentwicklung

- Gute Wartbarkeit der Software
  - Fehlerbeseitigung
  - Erweiterungen
- Optimaler Informationsfluss zwischen allen beteiligten Personengruppen
- Kosteneffizienz
- Hohe Funktionalität der Software

Besonderheit bei Eingebetteten Systemen

Rechnersysteme, fest integriert in einen technischen Kontext

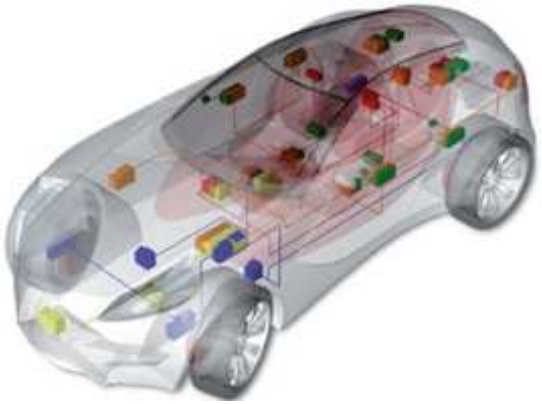
Was sind Eingebettete Systeme?

- Enge Kopplung zum technischen Prozess
- Verteilte Systeme mit hoher Dynamik
- Hohe Zuverlässigkeit gefordert
- Wartungsfreundlichkeit und Portierbarkeit der Software

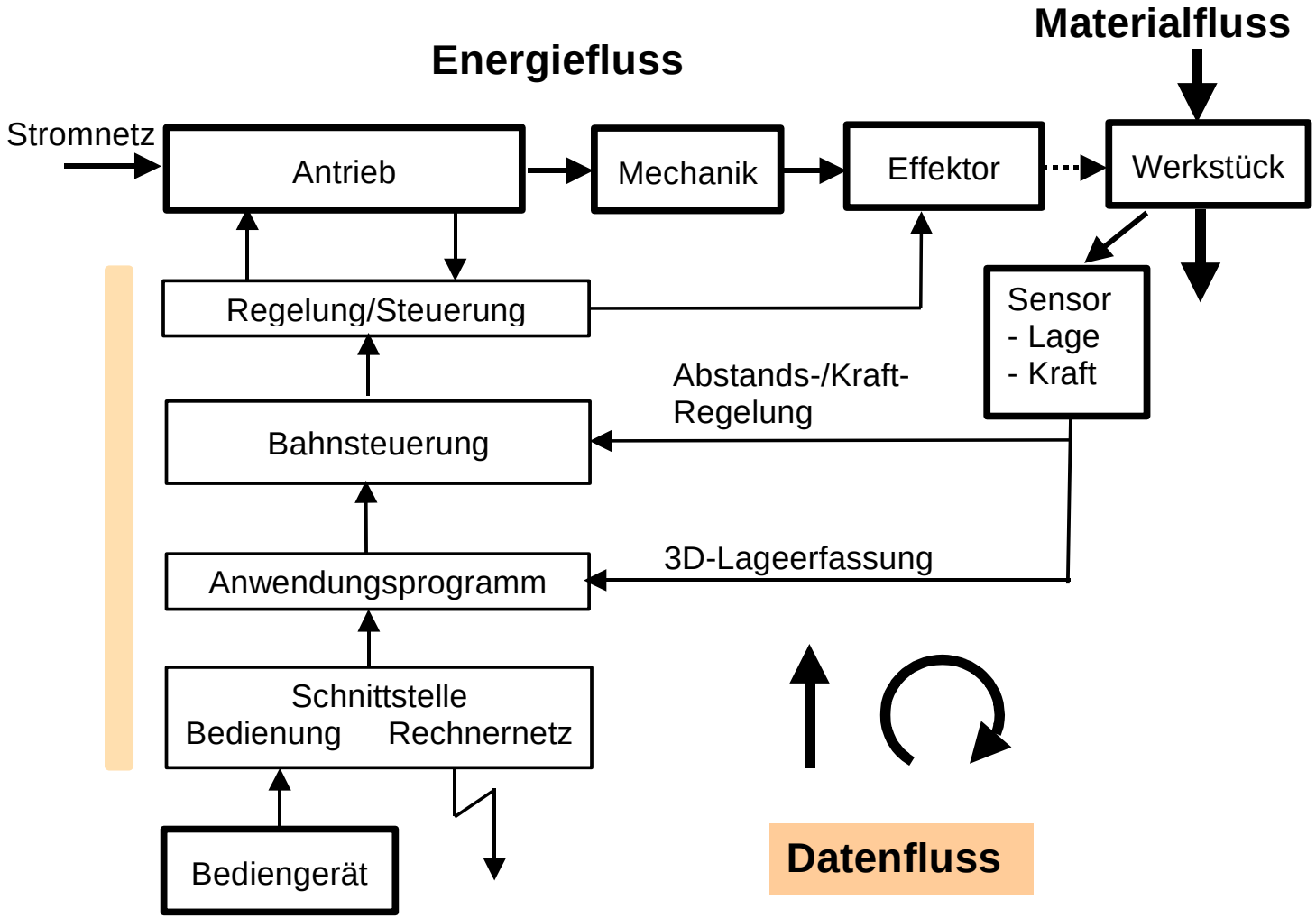
Wichtige Anwendungsgebiete

Verkehr | **Automation** | **Konsum** | **Medizin ...**

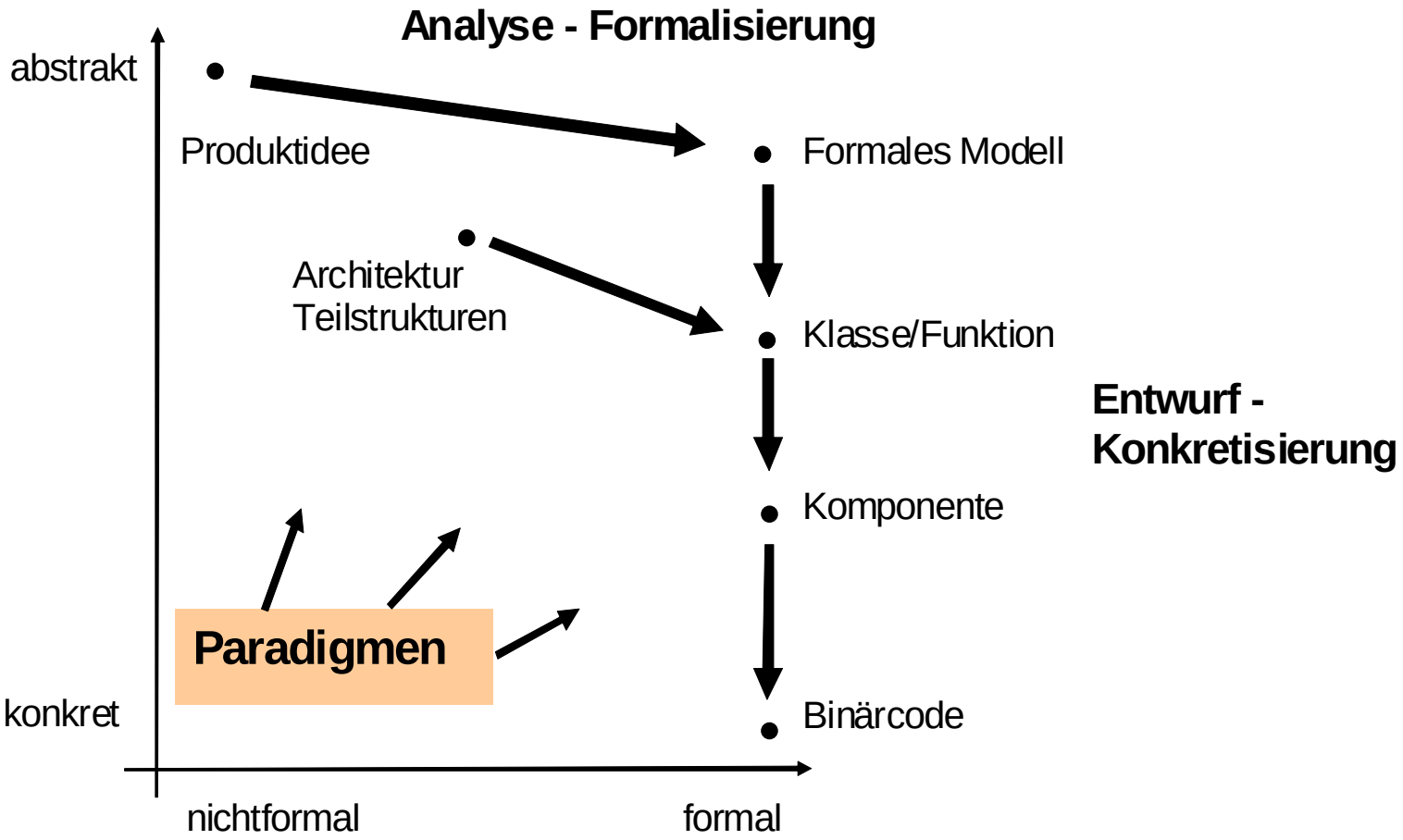
Mechatronik
Robotik



Anwendung - Mechatronische Systeme



Darstellung des Programmwissens in der Entwurfsebene



Alternative Ansätze bei Programmier-Paradigmen

- Primäre Darstellung
  - prozedural
  - deklarativ
- Unterstützung der Wissensdomäne
  - Struktur des Wissens
  - Inhaltliche Darstellung des Wissens
- Abstraktionsebene (Semantische Lücke)

Vergleich der Paradigmen

Paradigma	Primäre Darstellung	Wissensdomäne	Abstraktions-ebene (Semantik)
Imperativ	prozedural	Prozess (parallel, sequentiell)	niedrig
Objektorientiert	deklarativ	Struktur	mittel
Komponenten-orientiert	deklarativ	Struktur	hoch
Datenfluss-orientiert	deklarativ	Inhalt	hoch
Modellbasiert	deklarativ	Inhalt	hoch

Modellbasierte Entwicklung

- Formale, abstrakte Darstellung der Wissensinhalte
  - keine Realisierungsdetails
  - weitgehende Plattformunabhängigkeit
- Unterschiedliche Modellbegriffe
  - Reale Systeme / Prozesse
  - Datenstrukturen
- Domänenspezifische Sprache
  - einfach und prägnant
  - Benutzung ohne umfangreiche Programmierkenntnisse
- Simulation und Codegenerierung

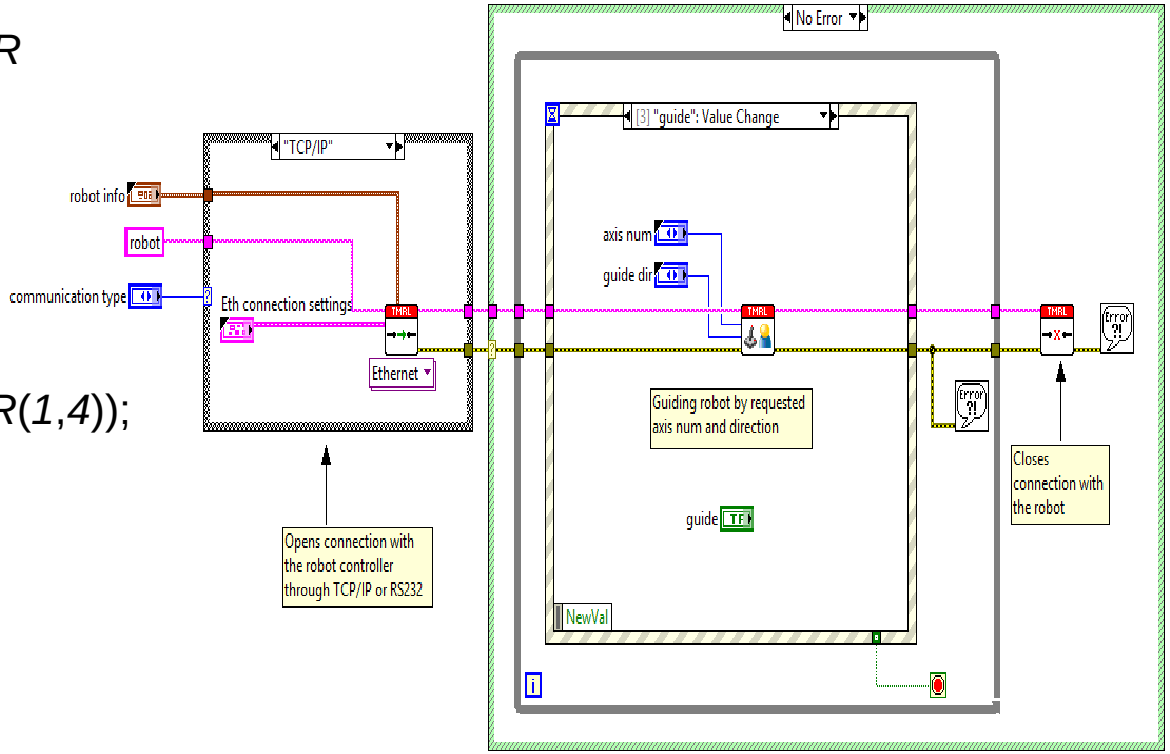
Was ist die Besonderheit ?

Domänensprachen Text/Grafik

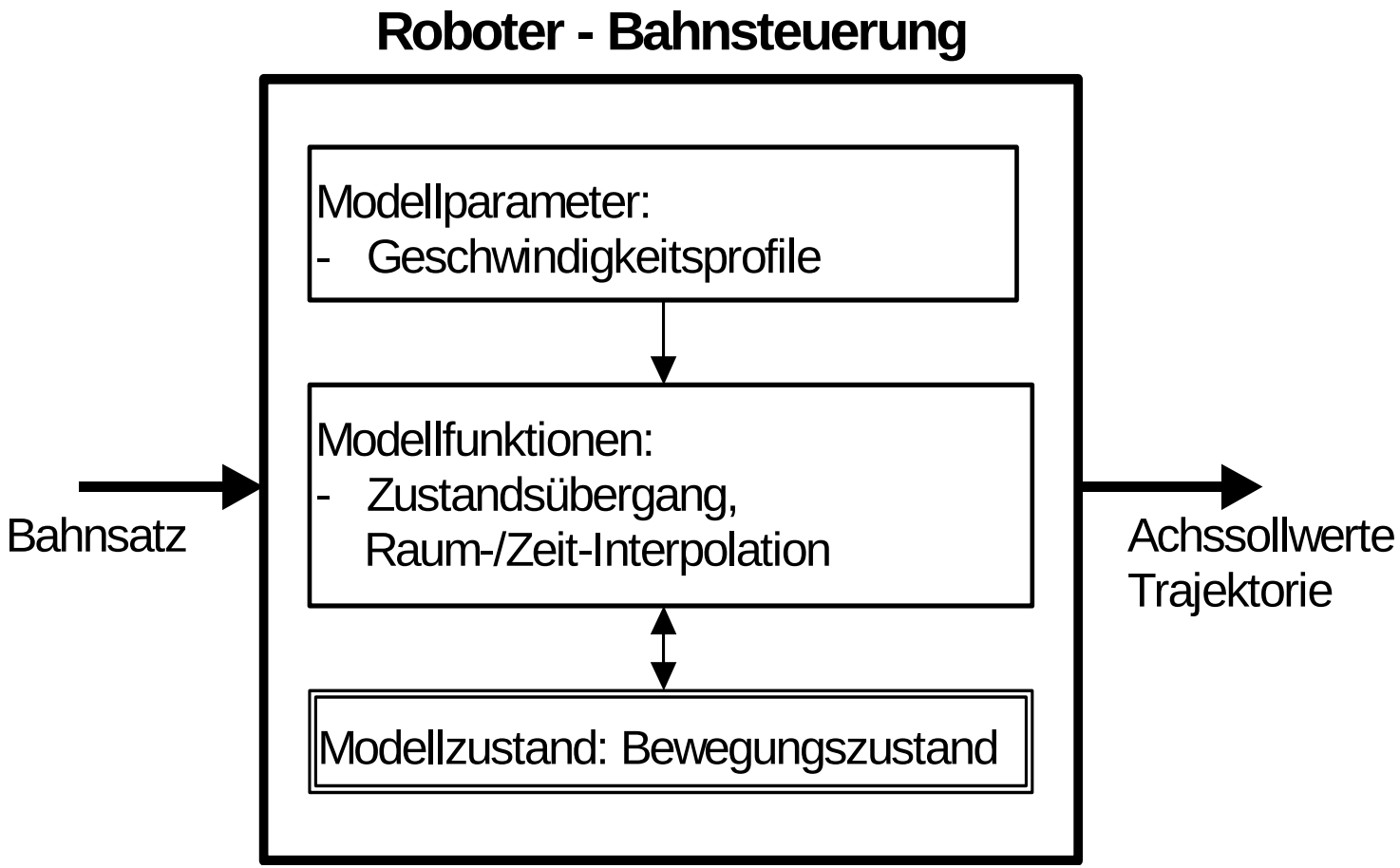
Matlab-Skript

```
% Transformation P2 nach KR
xk=u1-mk;   xk=xk/norm(xk);
yk=cross(zk, xk);
KR=[xk yk zk mk; 0 0 0 1];
P2_KR=inv(KR) * P2;
phi=atan2(P2_KR(2,4),P2_KR(1,4));
```

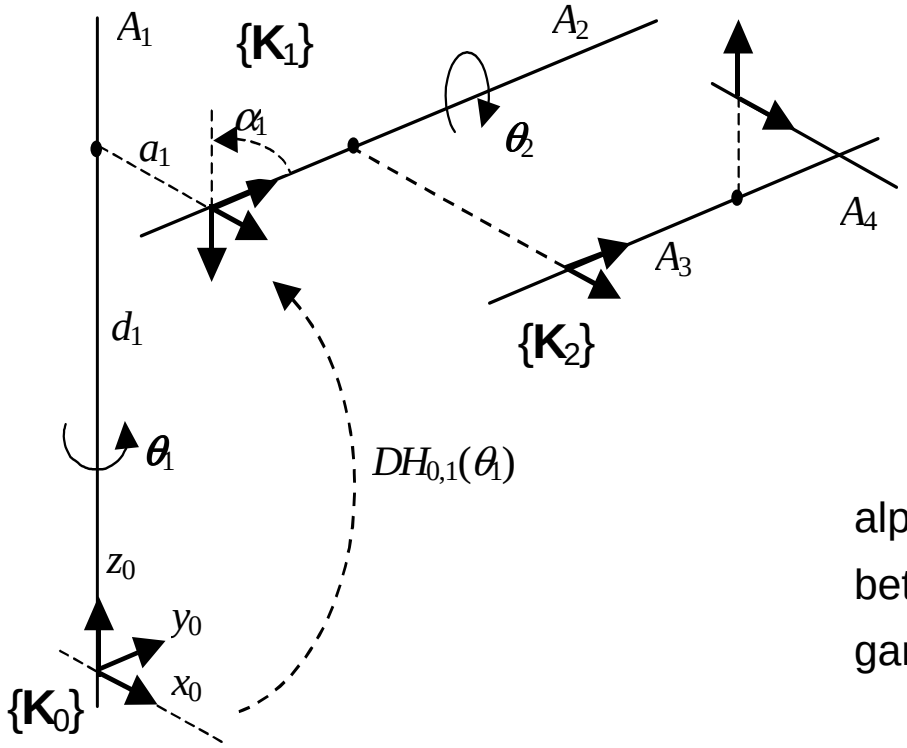
Labview - Grafik



Beispiel - Zustandsmodell Bewegung



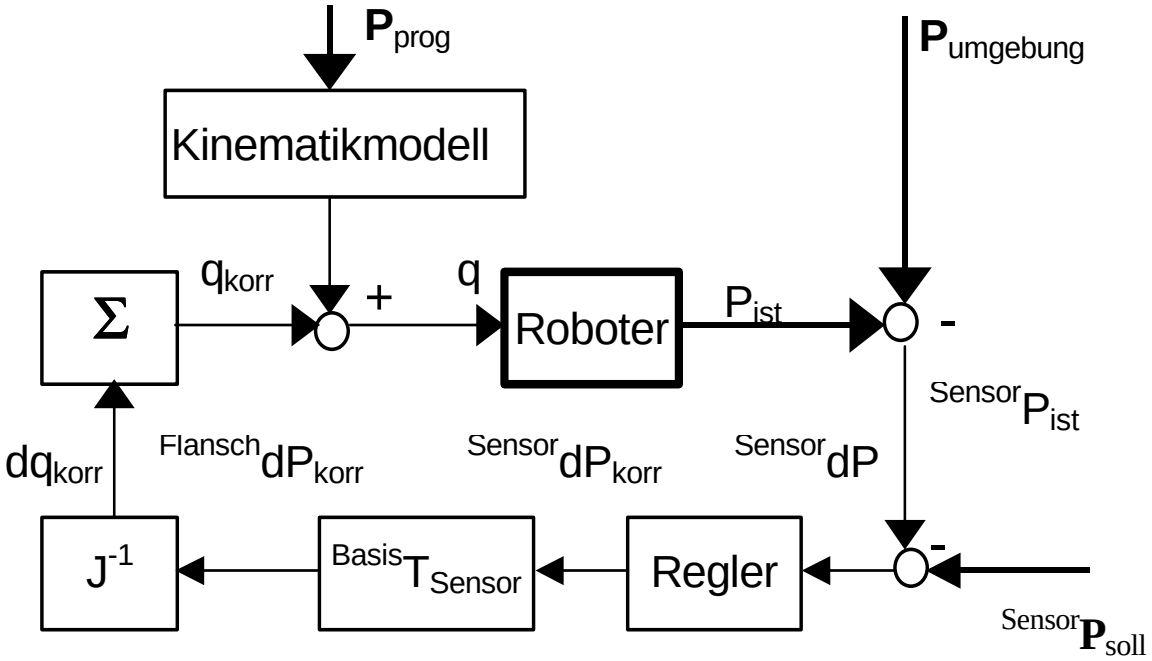
Roboter - Kinematikmodell

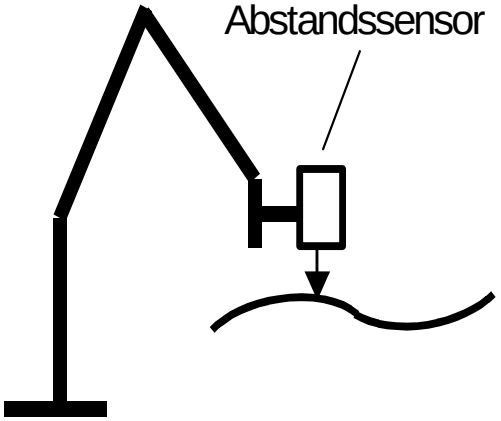



```
dhp = [  0      0.4    0     -pi/2;
         0      0      0.2    0;
        -pi/2   0      0     -pi/2;
         0      0.3    0     -pi/2;
         0      0      0      pi/2;
         0      0.1    0      0  ];
```

```
alpha  = atan2(hy1,hx1);
beta   = acos( (oa^2 - ua^2 + l^2)/(2 * l * oa) );
gamma= acos( (oa^2 + ua^2 - l^2)/(2 * oa * ua) );
```

Modell Abstandsregelung



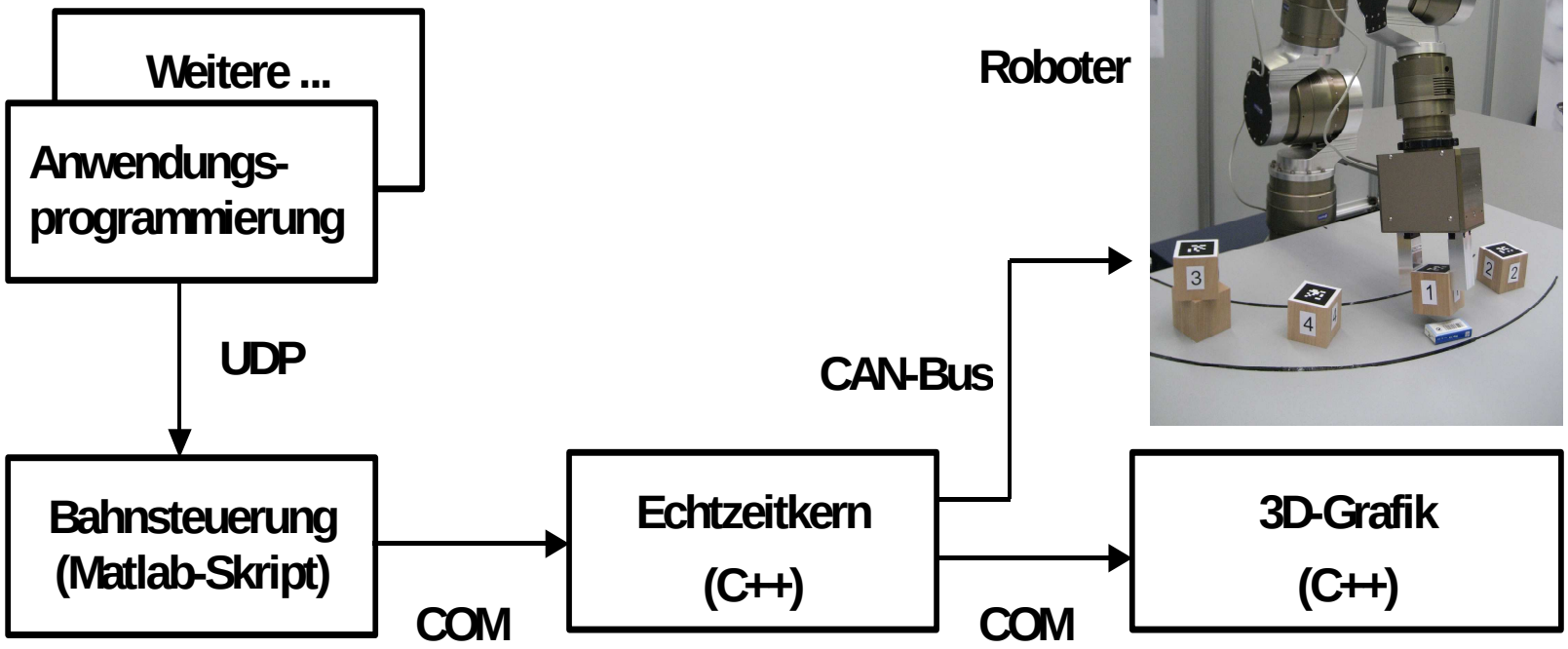


P: kartesische Werte, q: achsbezogene Werte

Komponentenorientierte Programmierung
Leitgedanken

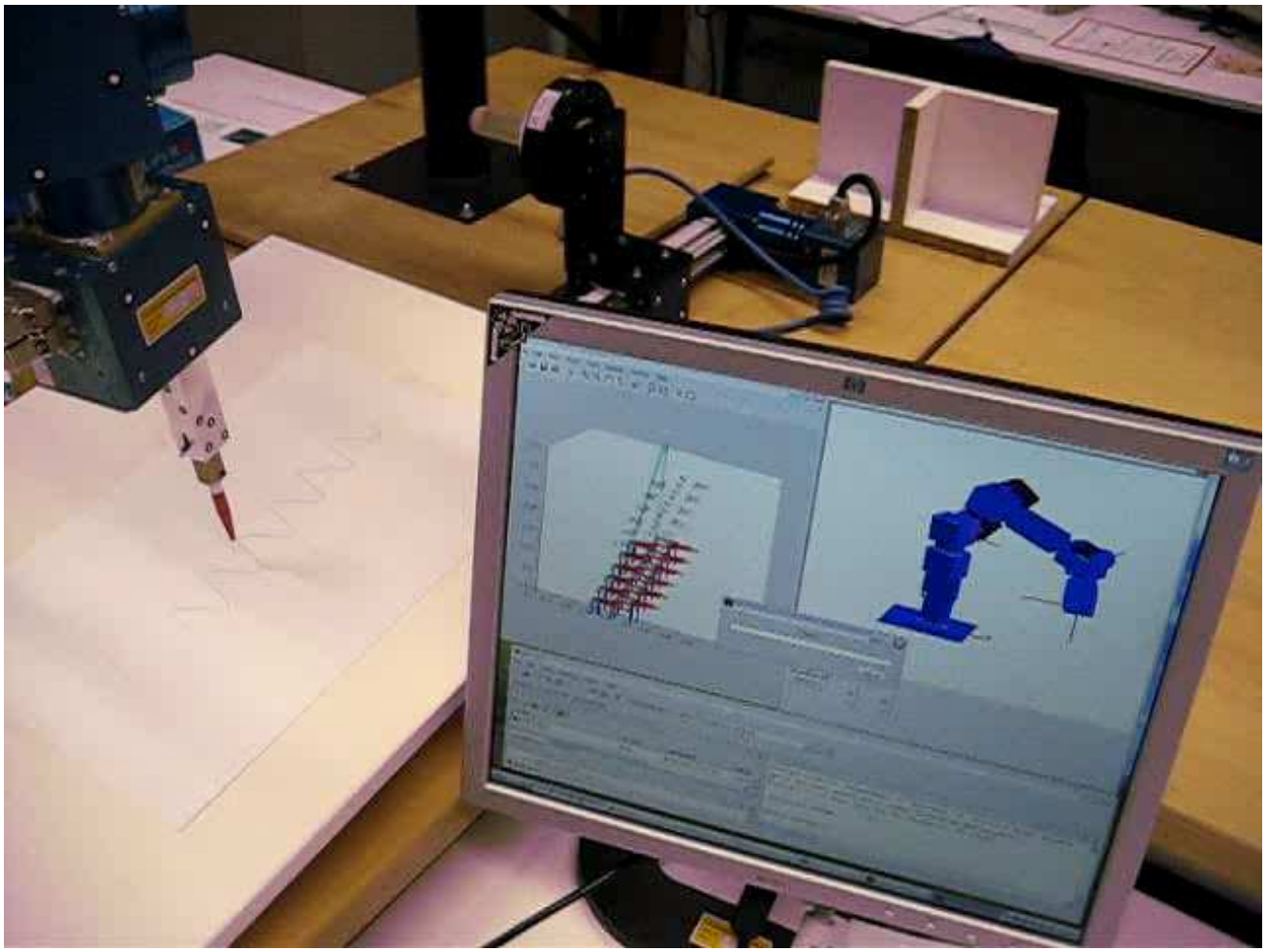
- Softwarekomponenten sind ausführbare Softwareteile, die über standardisierte Schnittstellen benutzt werden.
- Mit Hilfe von Komponenten kann eine Framework-Plugin-Architektur realisiert werden.
- Softwarekomponenten können mit beliebigen Programmiersprachen implementiert werden.
- Softwarekomponenten können dynamisch erzeugt werden.
- Beispiele für standardisierte Schnittstellen:
  - COM von Microsoft
  - CORBA von der OMG (www.omg.org),
  - JavaBeans
  - TCP/IP, UDP (allg. Netzwerkprotokolle)

Komponentenstrktur MRobot / SunSim

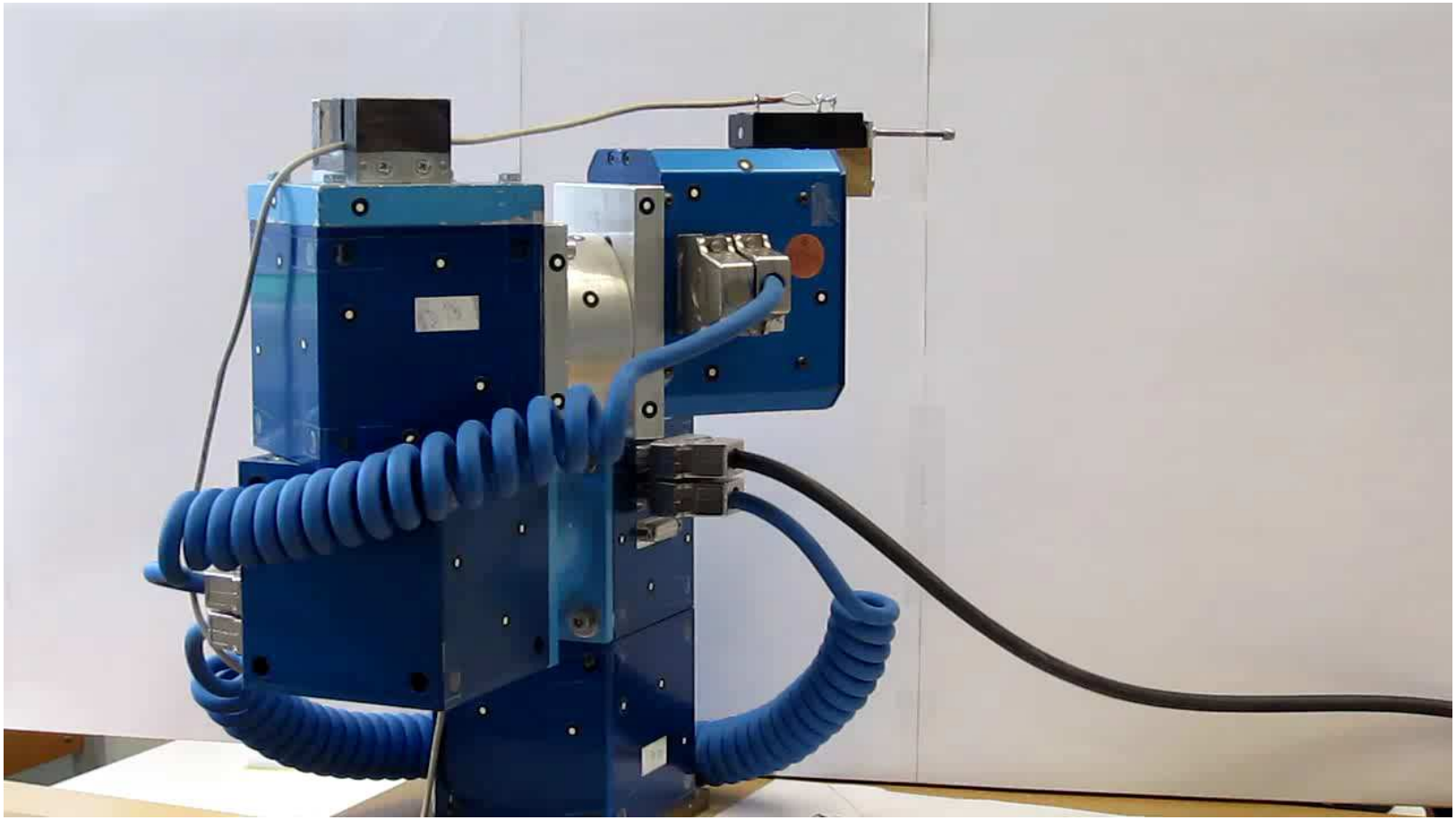


Anwendungsbeispiele

Synchrone Robotersteuerung und Simulation

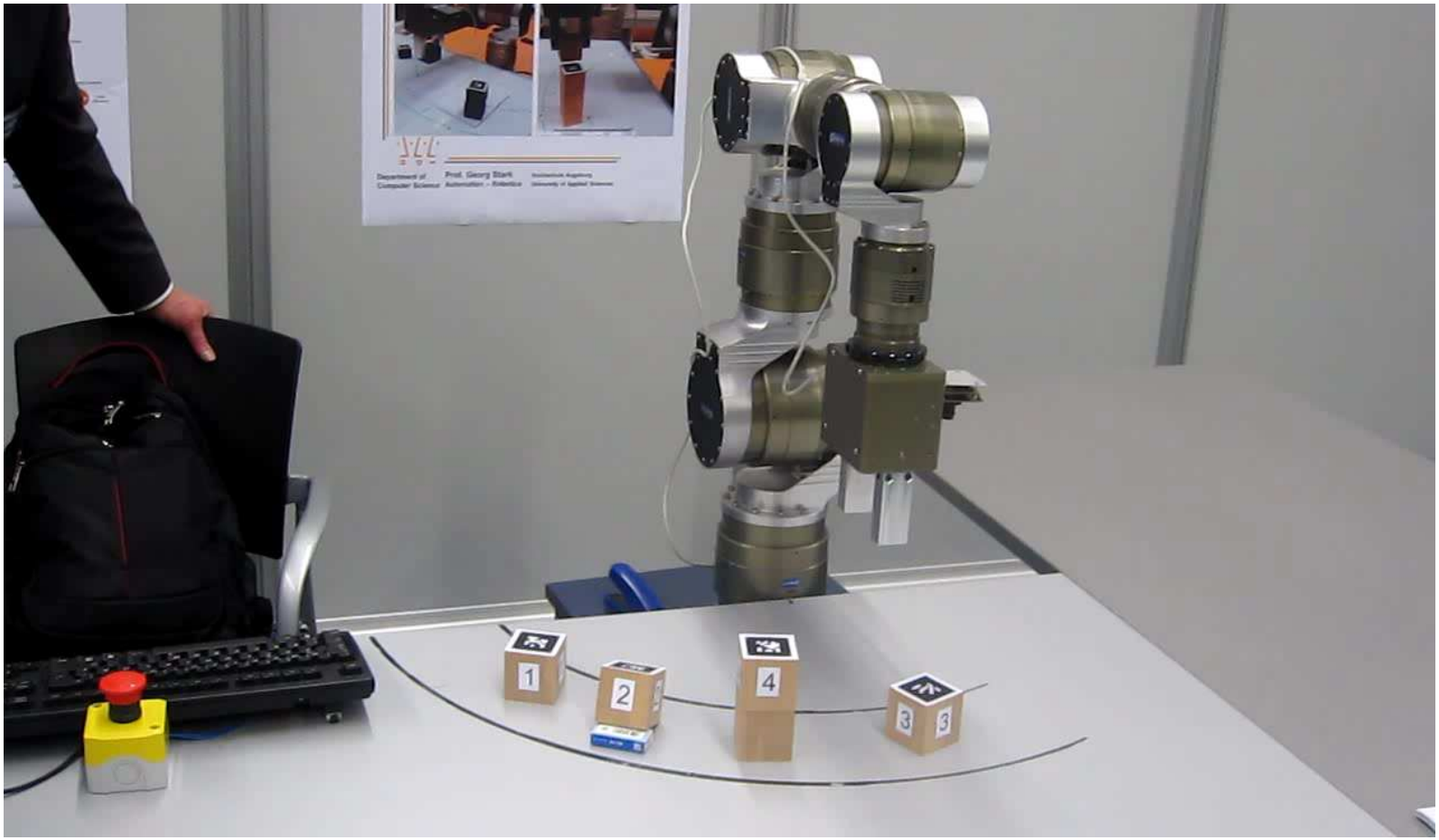


Sensorsteuerung

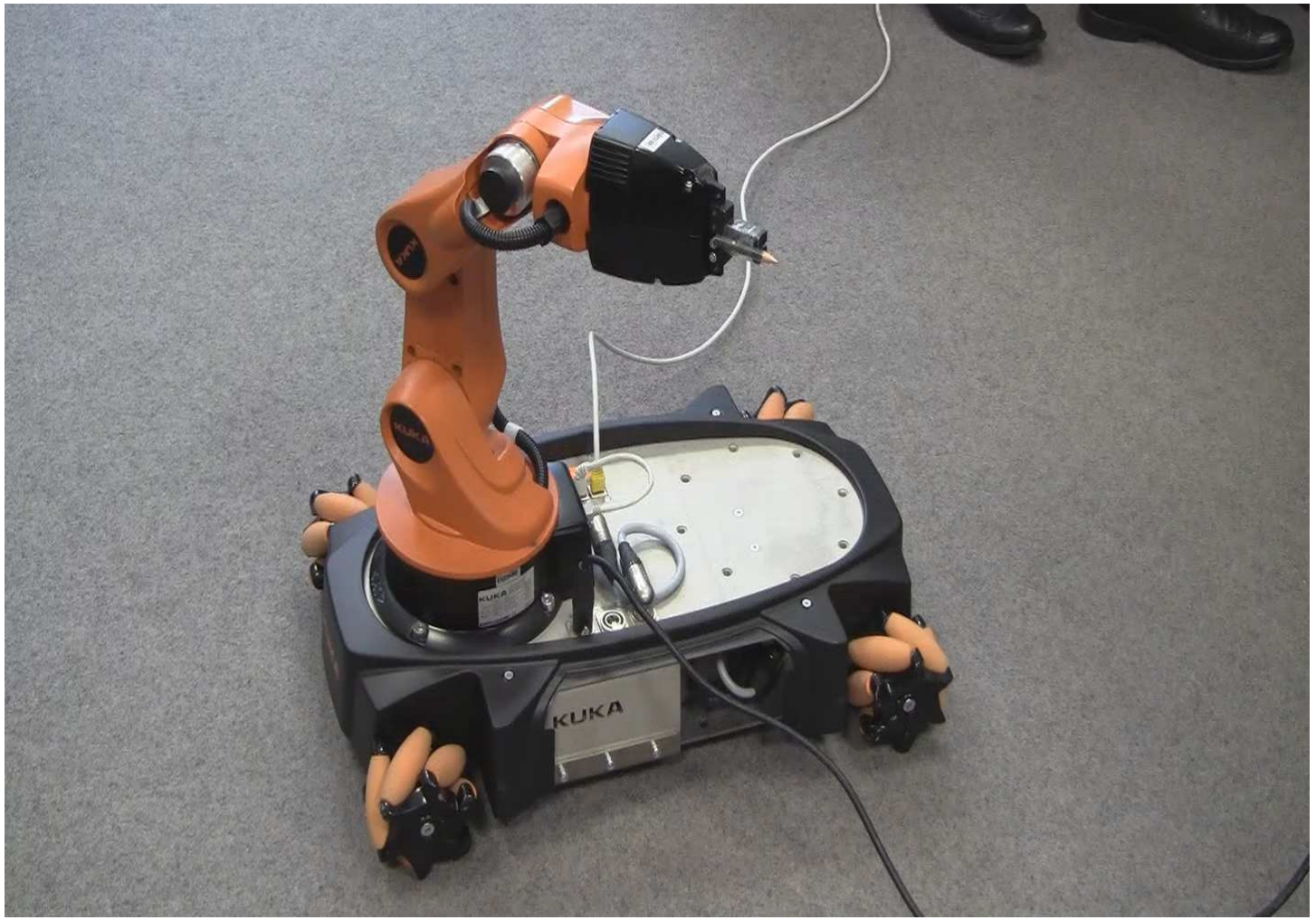


Schreiben – ELECTRONICA 2012

Objektlokalisierung mit Greiferkamera



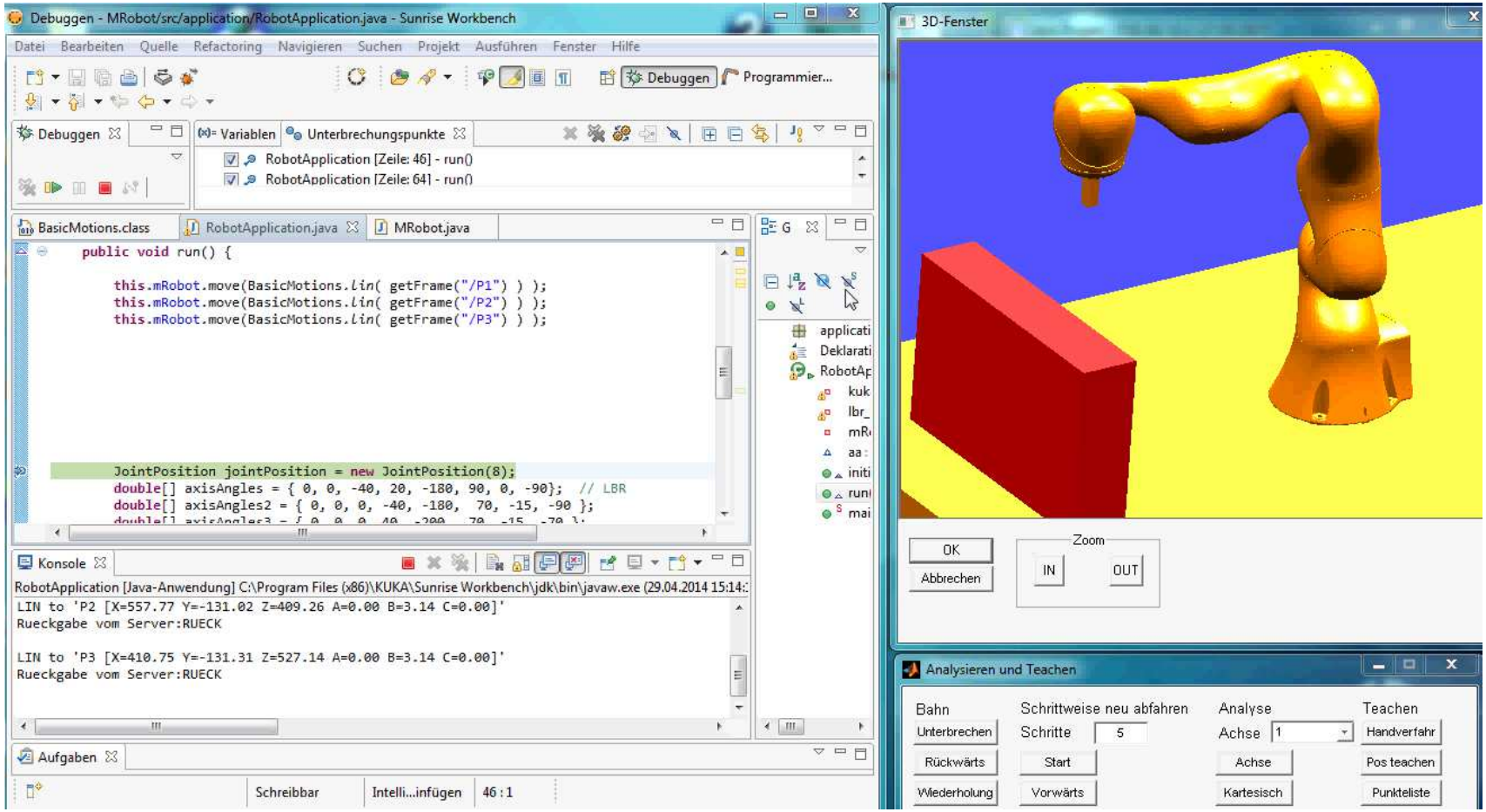
Nullraum-Steuerung KUKA youBot



HS Augsburg
Fak. Informatik

Simulationskomponente für KUKA Sunrise Workbench

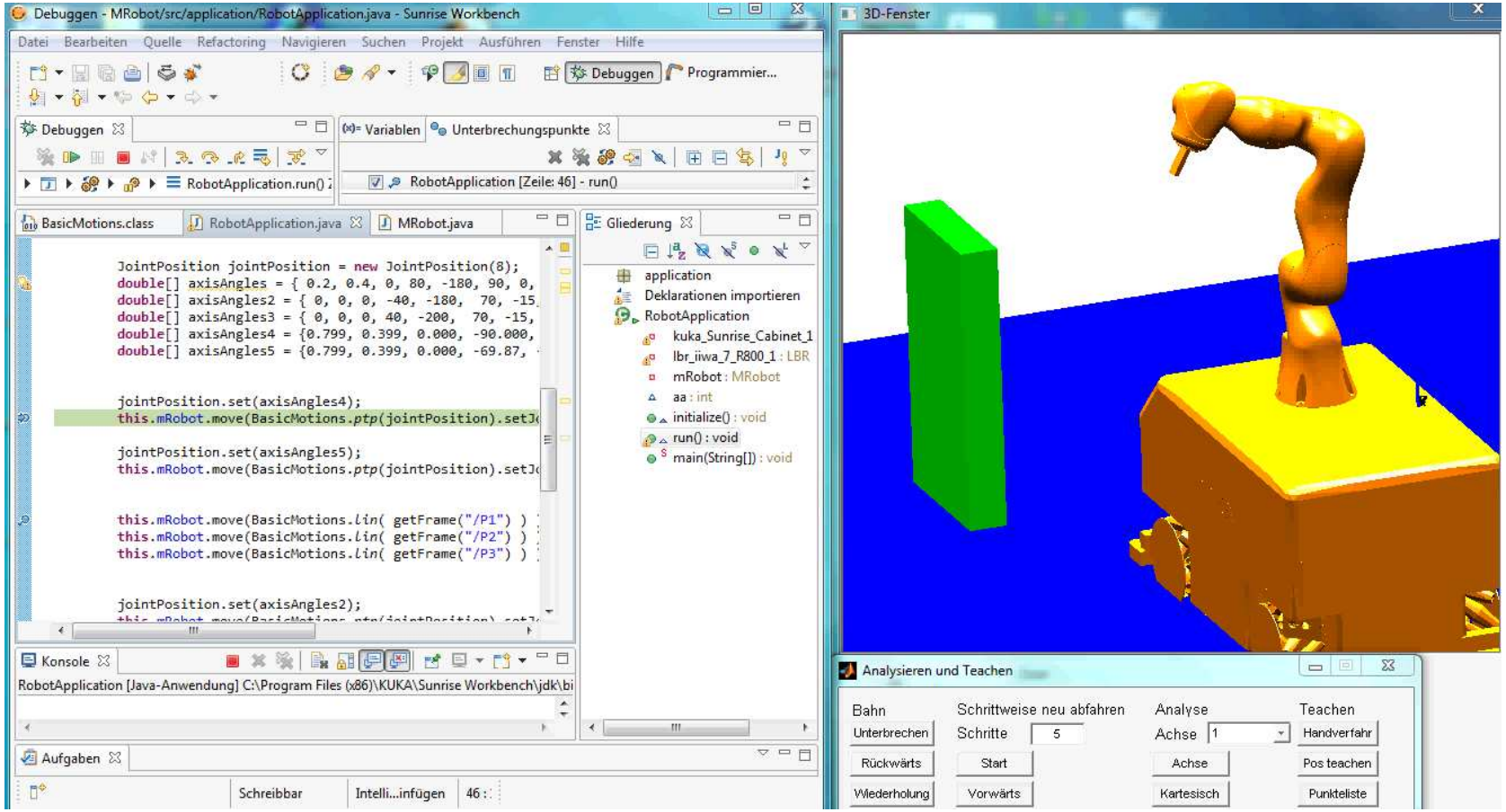
CIM & Robotik
Prof. G. Stark



Simulation KUKA omniRob

HS Augsburg
Fak. Informatik

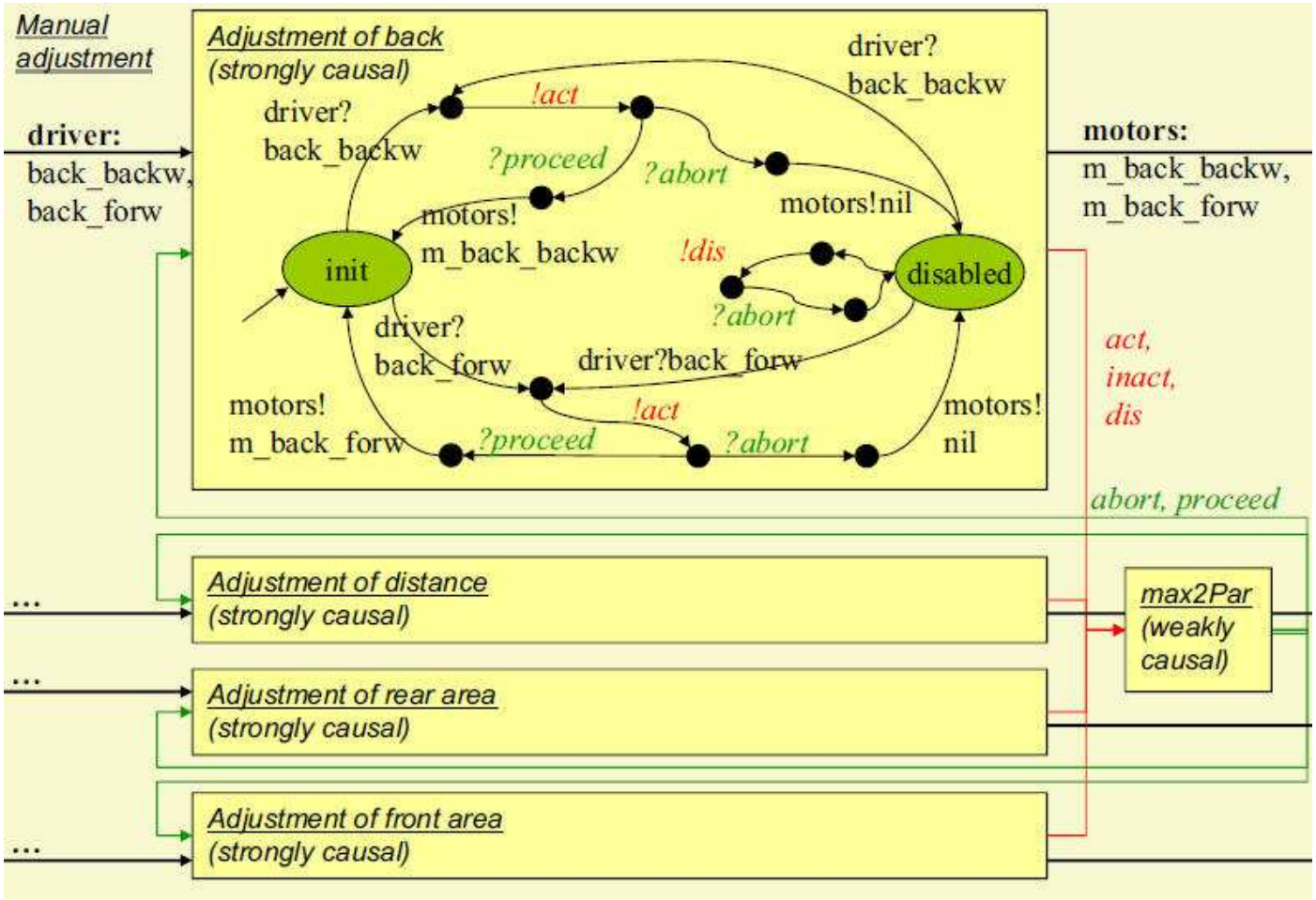
CIM & Robotik
Prof. G. Stark



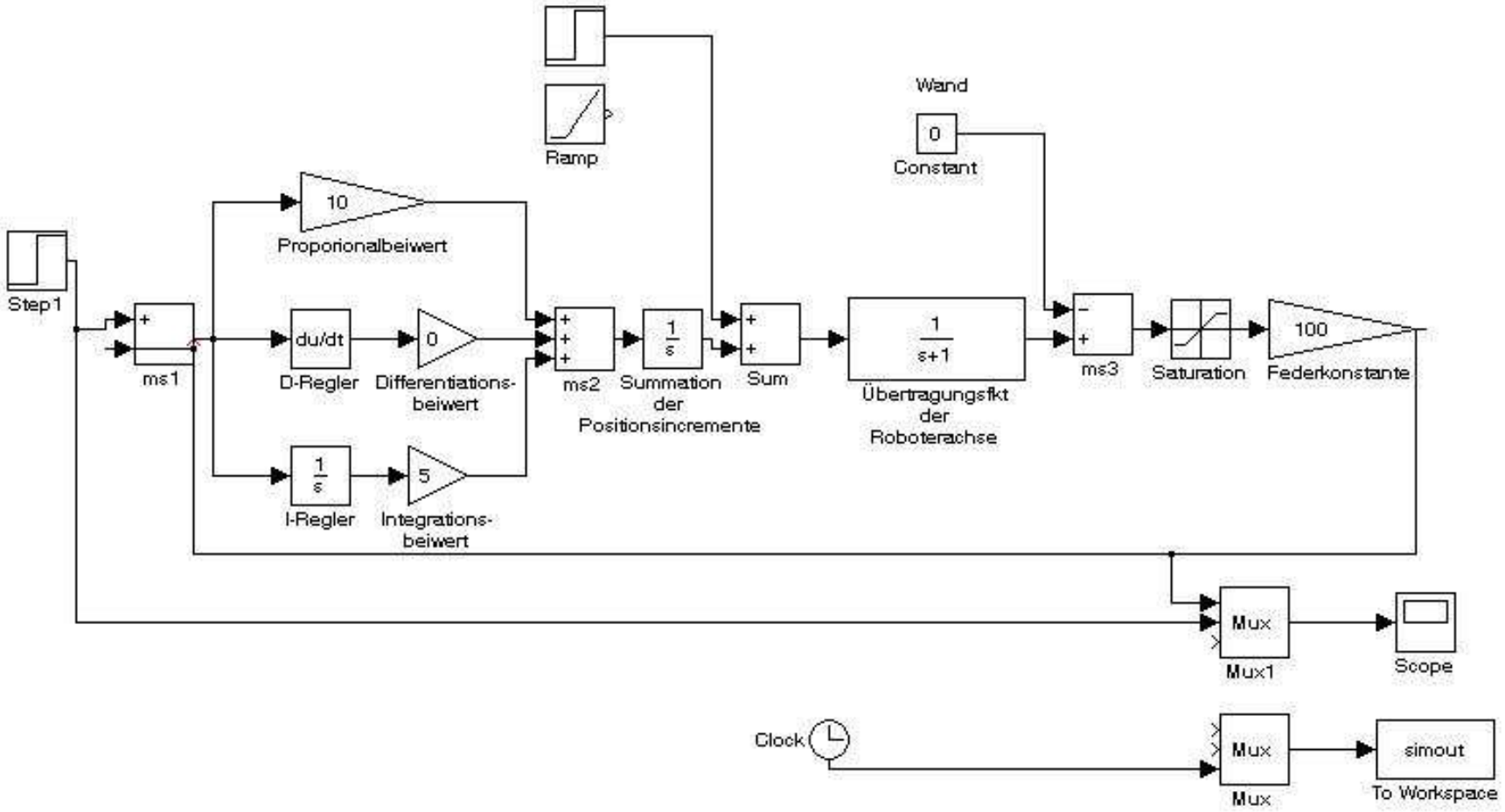
Neue Verfahren und Technologien

- Modellierung von realen Systemen, Prozessen
  - Matlab/Simulink, Realtime Workshop
  - Modelica
  - Labview
  - ASCET SD
  - SCADE
- Datenmodelle
  - Eclipse, Modelling Framework (EMF)
  - Visual Studio, Modellierungs-SDK

Beispiel - Prozessmodell



Matlab/Simulink - Signalfluss



Modelica – Modellierung phys. Systeme

Example: Hierarchical Modelica Model

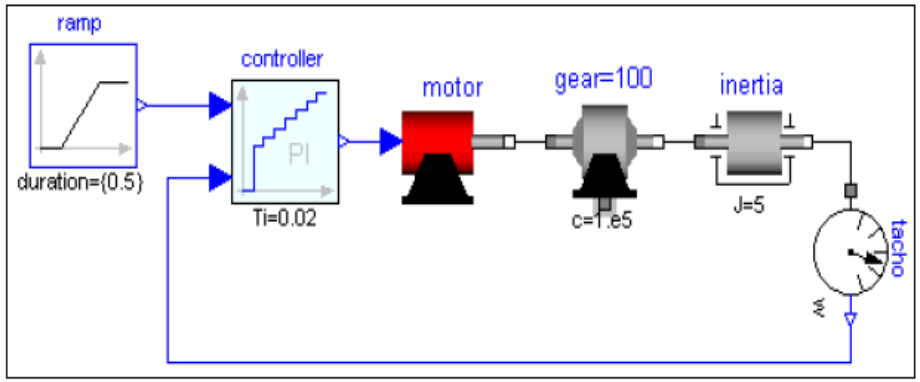
textual representation

```
model MotorDrive
  PI          controller;
  Ramp        ramp;
  Motor       motor;
  Gearbox     gear(ratio = 100);
  Inertia     inertia(J = 10);
  SpeedSensor tacho;
equation
  connect(controller.y    , motor.i_ref);
```

Class name

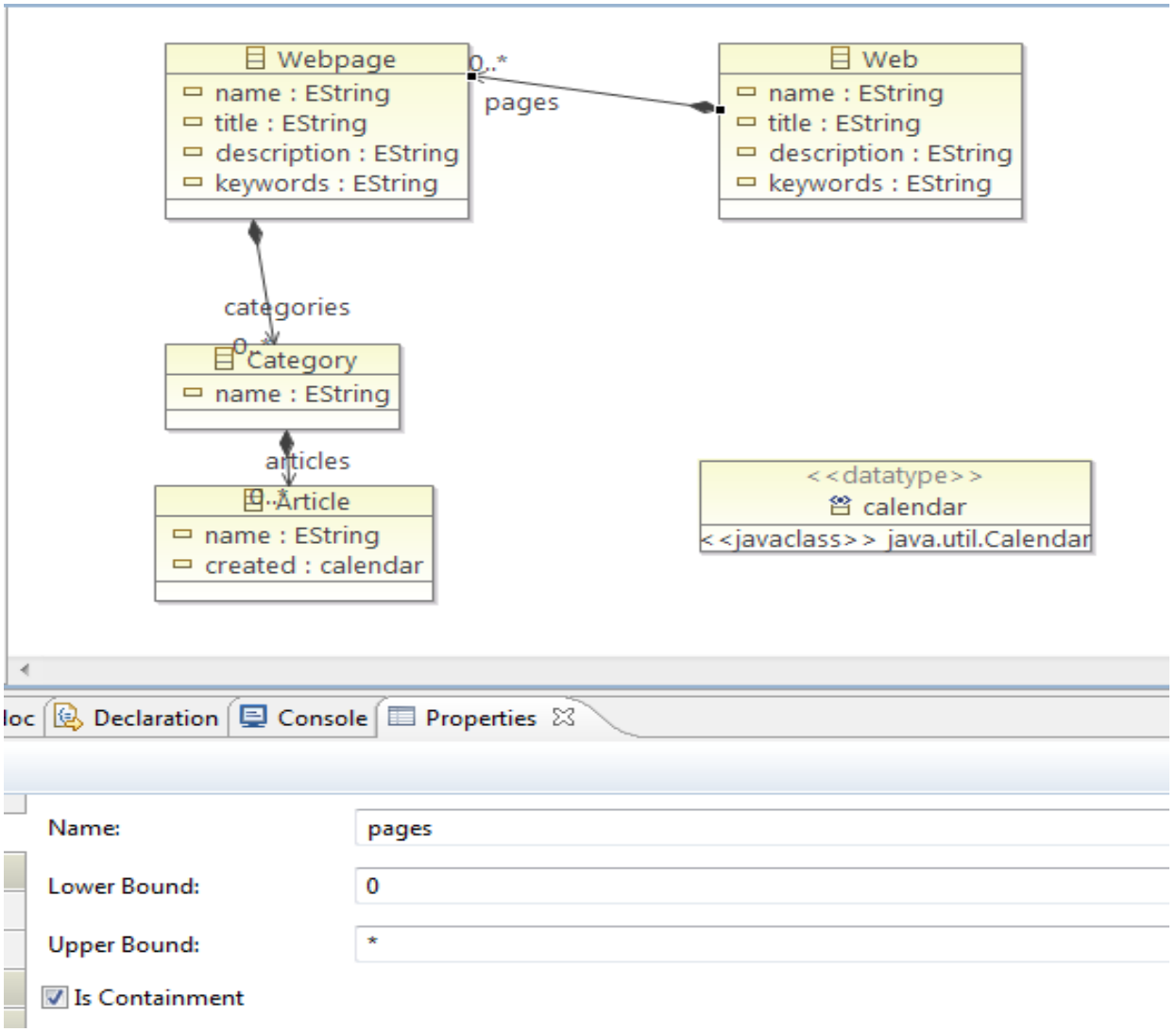
Modifier

Instance name



graphical representation

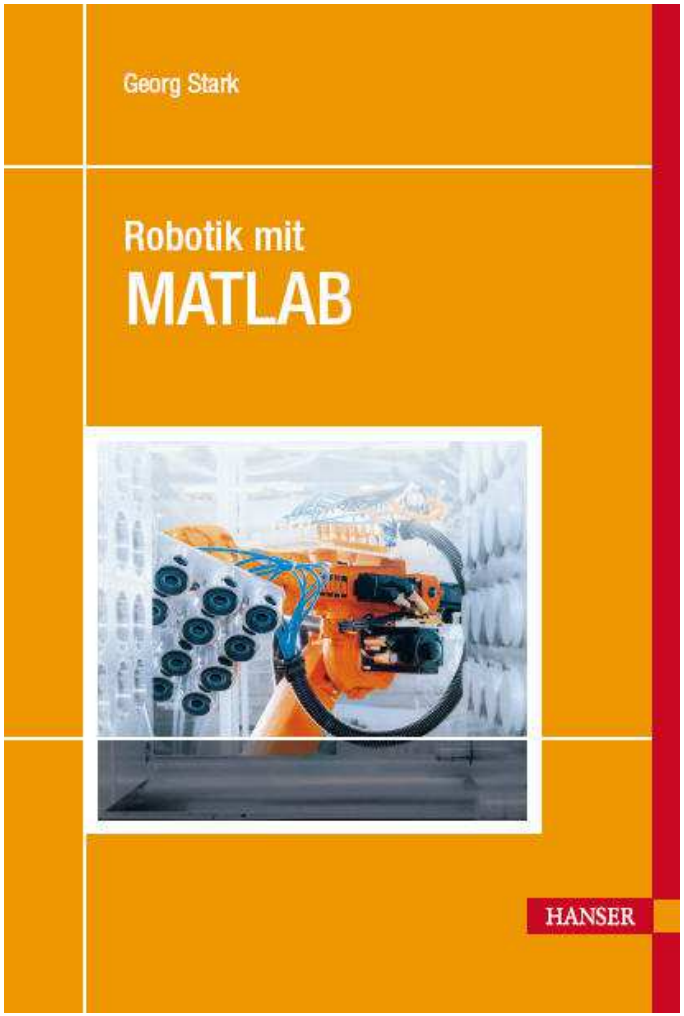
Datenmodell in EMF



Zusammenfassung

- Die Programmierung von Eingebetteten Systemen erfordert Wissen über das
  - umgebende System,
  - die auszuführenden Prozesse.
- Die modellbasierte Programmierung unterstützt die
  - formale Darstellung ohne Realisierungsdetails,
  - weitgehende Plattformunabhängigkeit.
- Die dargestellten Modelle können sofort simuliert und zur Codegenerierung benutzt werden.
- Die Aufteilung der Software auf binäre Komponenten erleichtert Installation und Wartung.
- Das Ergebnis ist kosteneffiziente Software mit guter Wartbarkeit und hoher Qualität

Modellbasierte Prog: Robotik mit Matlab



Vielen Dank für ihre Aufmerksamkeit!