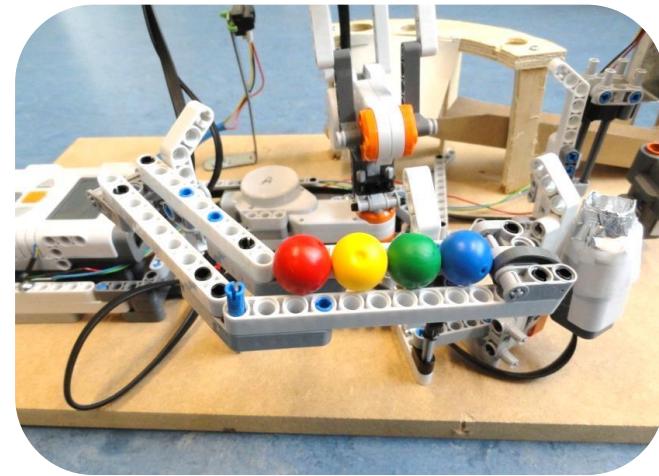
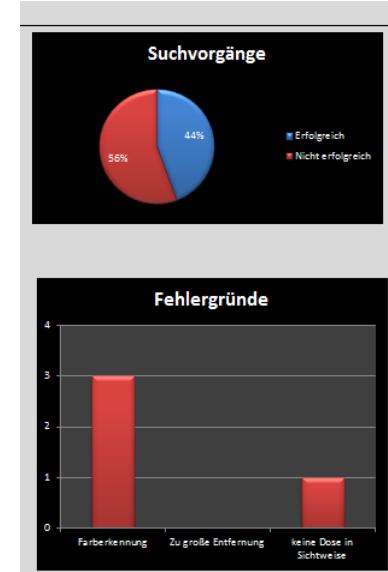
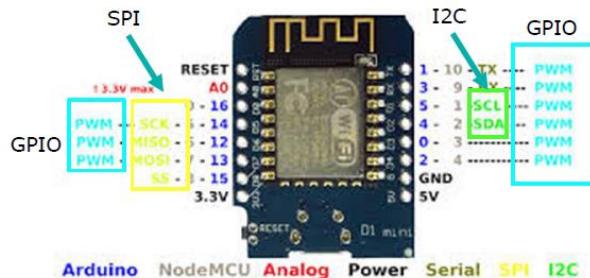


Datenmanagement, Leittechnik (und statistische Prozesslenkung-> Data Science!)



Prof. Dr.-Ing. Stephan Kallweit
Fachbereich Maschinenbau & Mechatronik
Fachgebiet Mess- und Automatisierungstechnik
Raum 01411
Goethestr. 1
52064 Aachen

T +49. 241. 6009 52348
F +49. 241. 6009 52681
kallweit@fh-aachen.de
www.fh-aachen.de

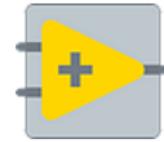


Praktikum

10 Termine

1. Termin ab 12.10.2021

Gruppeneinteilung erfolgt über
campus



Di. 10:15 Gr. A

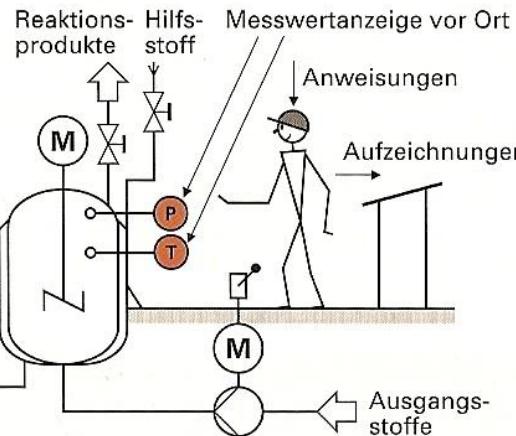
Di. 12:15 Gr. B

PT: LabView, Python, IoT, Machine
Learning
Prüfungsform:
Projektarbeit, Klausur

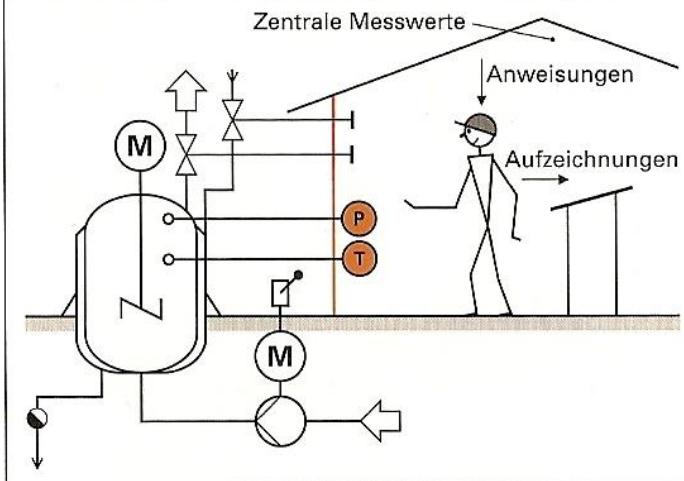
Auszug aus der Beschreibung des Studiengangs

„Vom Bestellprozess, über die Logistikströme, bis hin zur Auslieferung der Produkte lernen die Studierenden die einzelnen Schritte des Produktionsprozesses nicht nur kennen, sondern lernen auch, diese kompetent zu planen und zu managen. [...] Die Absolventen beherrschen das ganzheitliche Verstehen, Planen und Weiterentwickeln von Produktionsprozessen, so dass sie neuartige Produktionsweisen im Zusammenspiel von Mensch, Material und Maschine aufbauen können.“

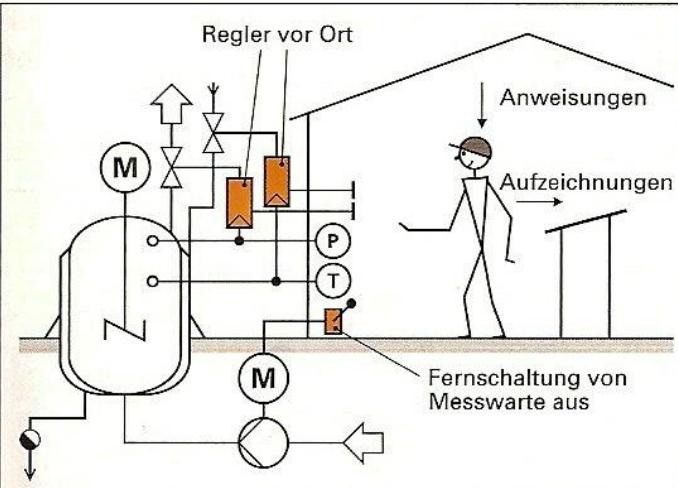
Historie 1900 - 1955



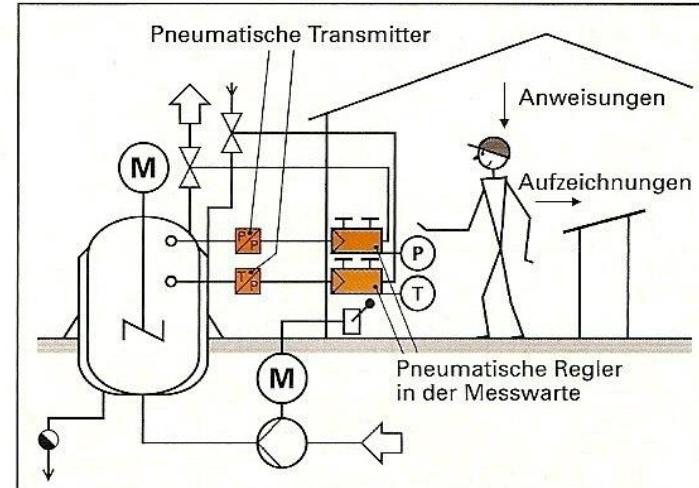
Prozessleittechnik bis ca. 1915



Prozessleittechnik von 1915 bis ca. 1935



Prozessleittechnik von 1935 bis ca. 1945

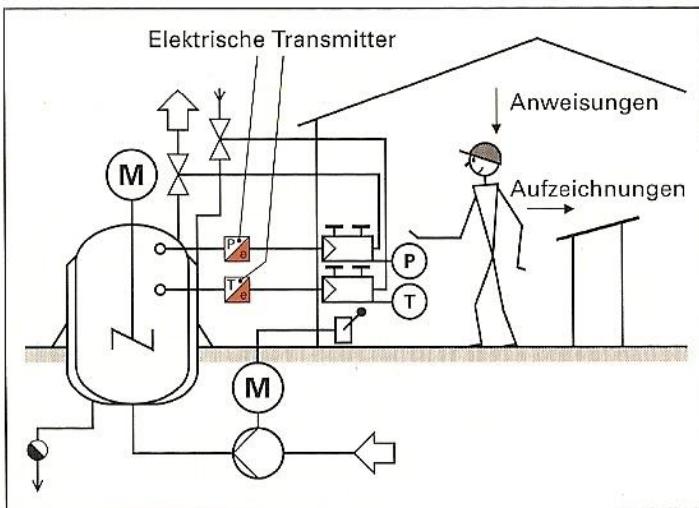


Prozessleittechnik von 1945 bis ca. 1955

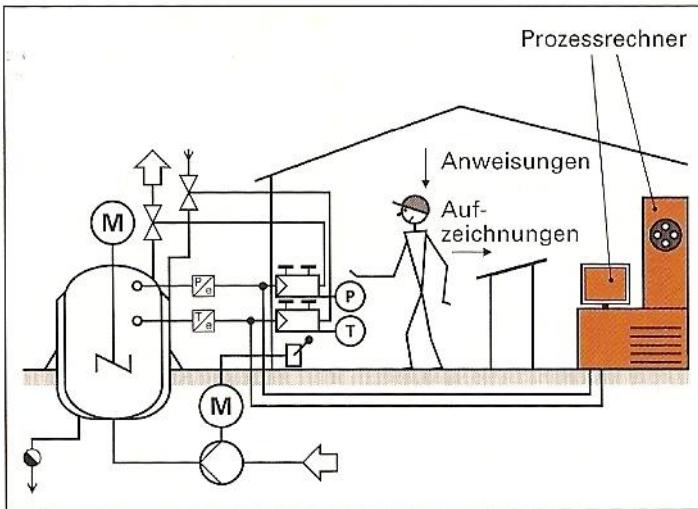
Offline

Online-
Open Loop

Historie 1955 - 1995

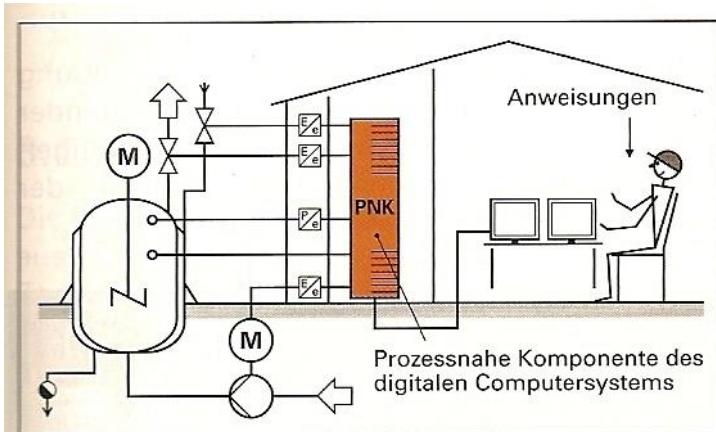


Prozessleittechnik von 1955 bis ca.1965

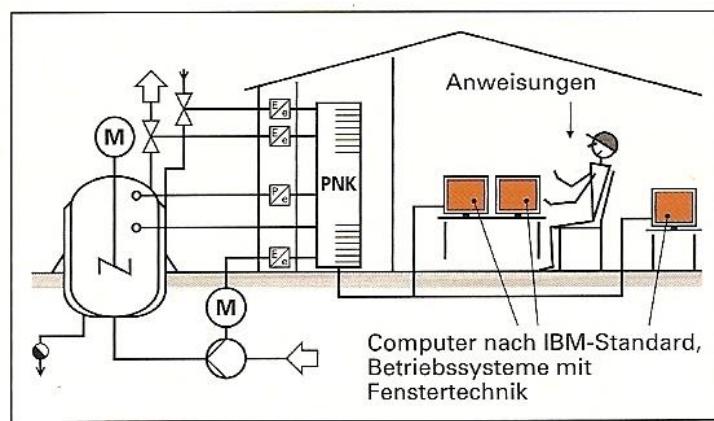


Prozessleittechnik von 1965 bis ca.1975

Online-
Open Loop



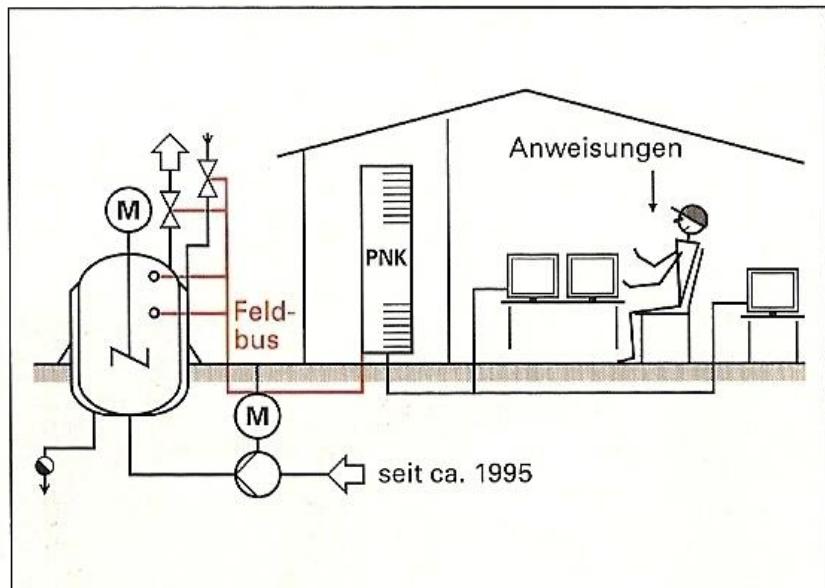
Prozessleittechnik von 1975 bis ca.1985



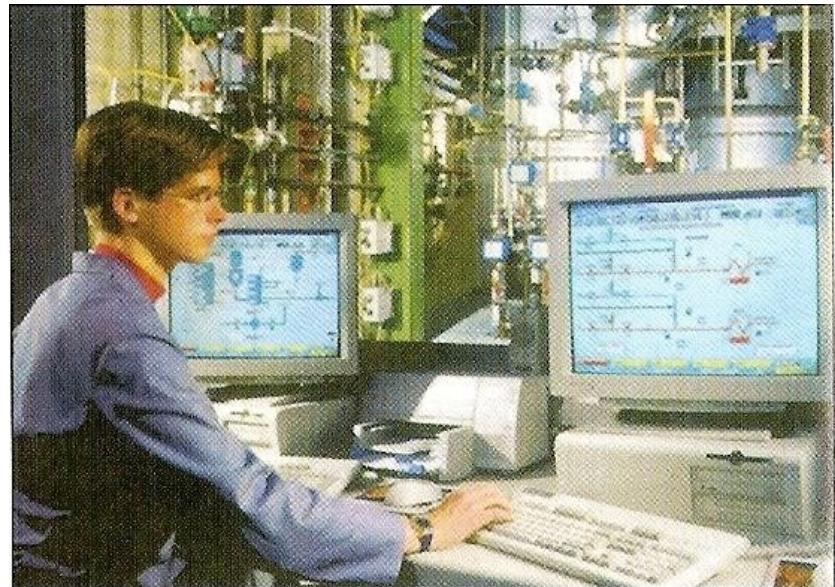
Prozessleittechnik von ca. 1985 bis ca.1995

Online-
Closed Loop

Historie 1995 - heute

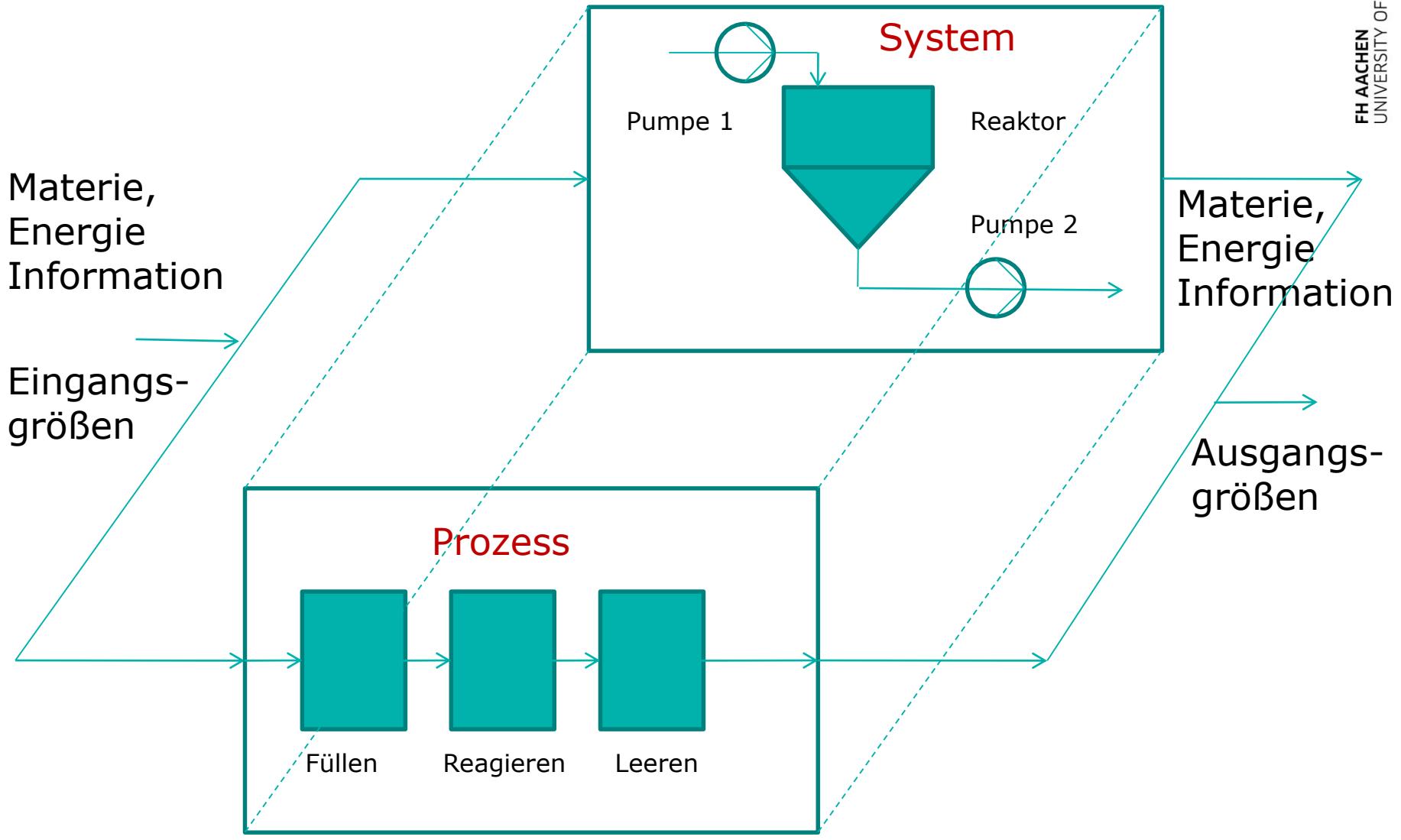


Tendenz in der Prozessleittechnik seit ca. 1995

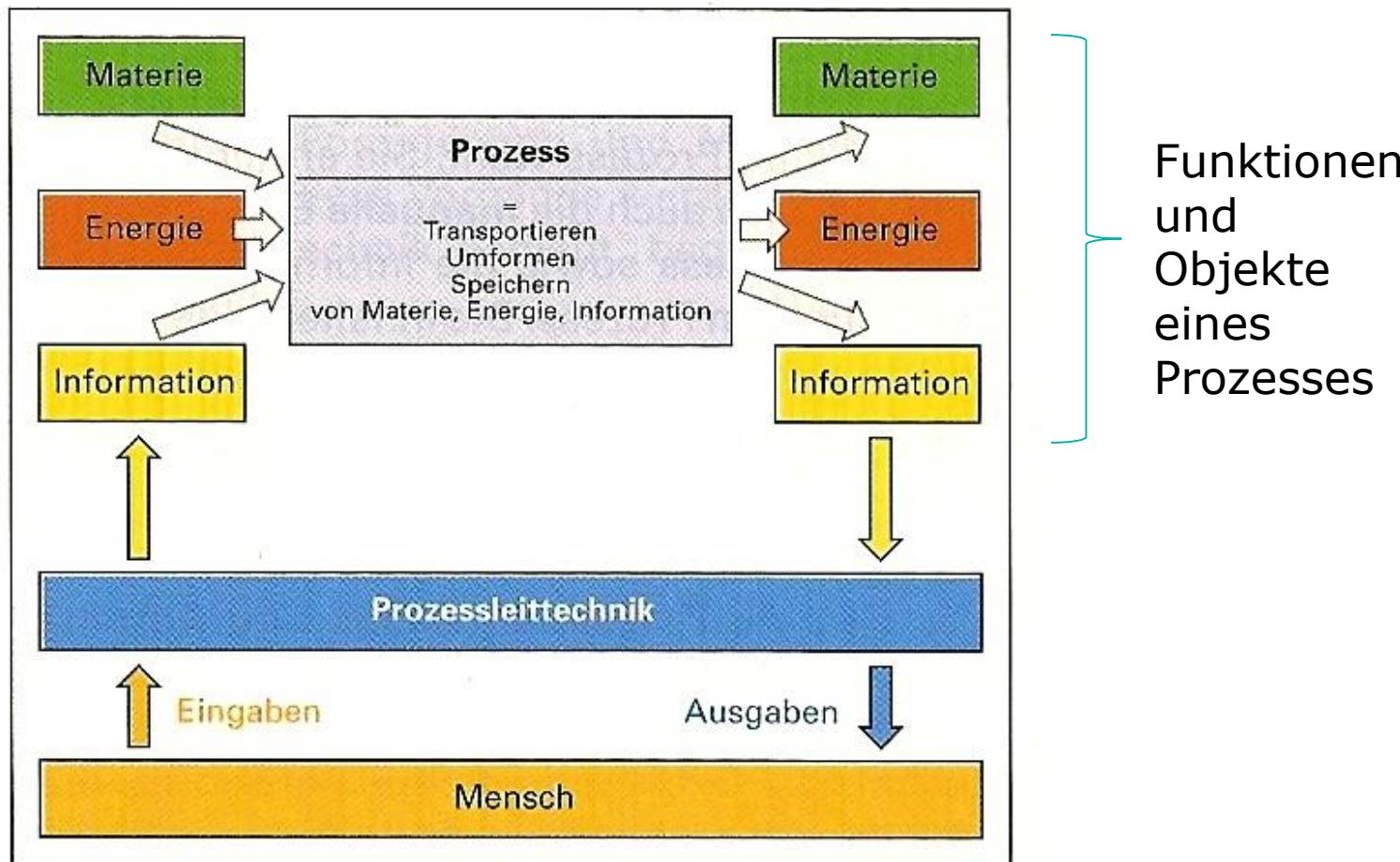


Bedienung einer Technikumsanlage mit einem modernen Kleinprozessleitsystem

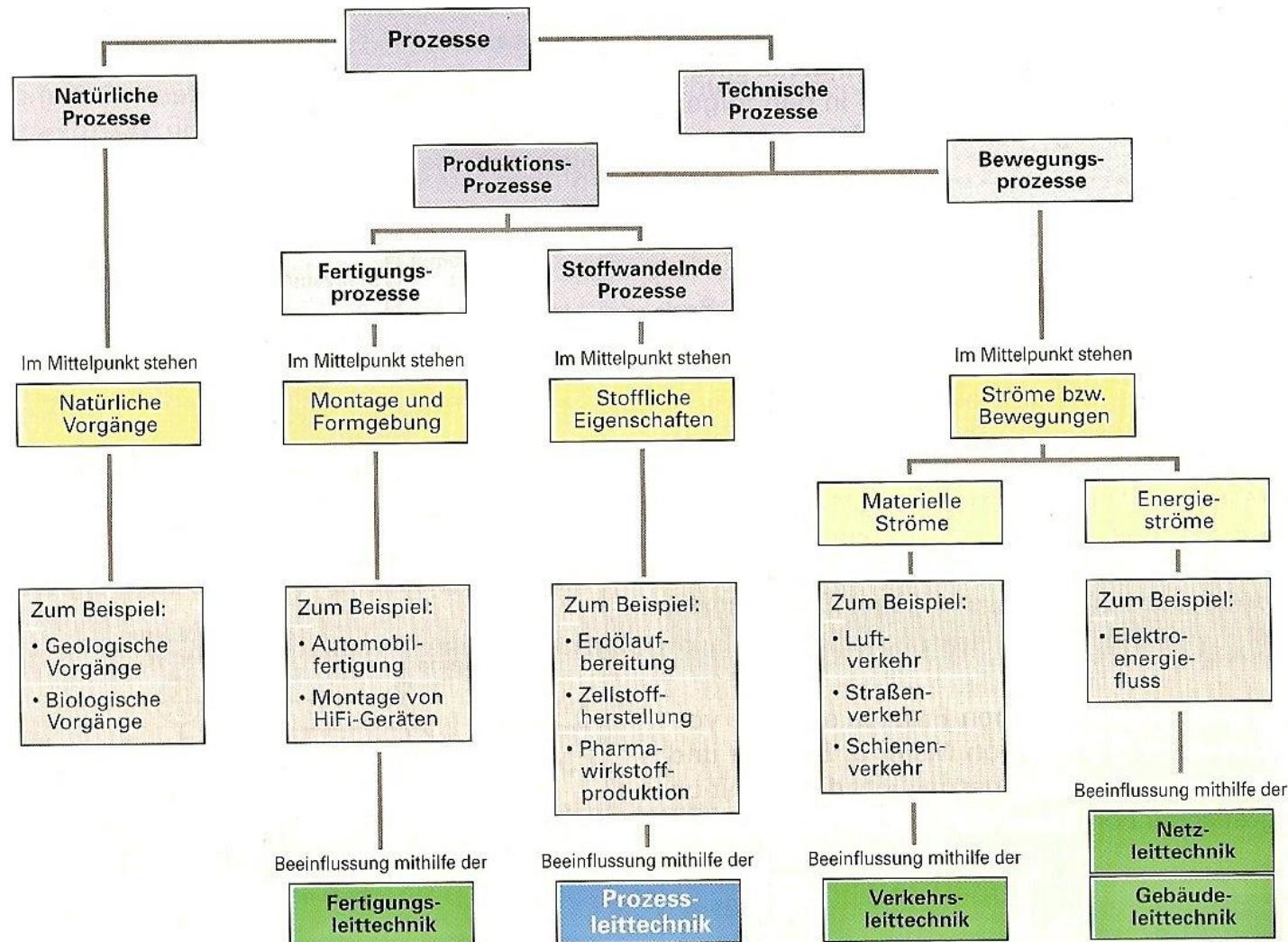
Prozessabläufe



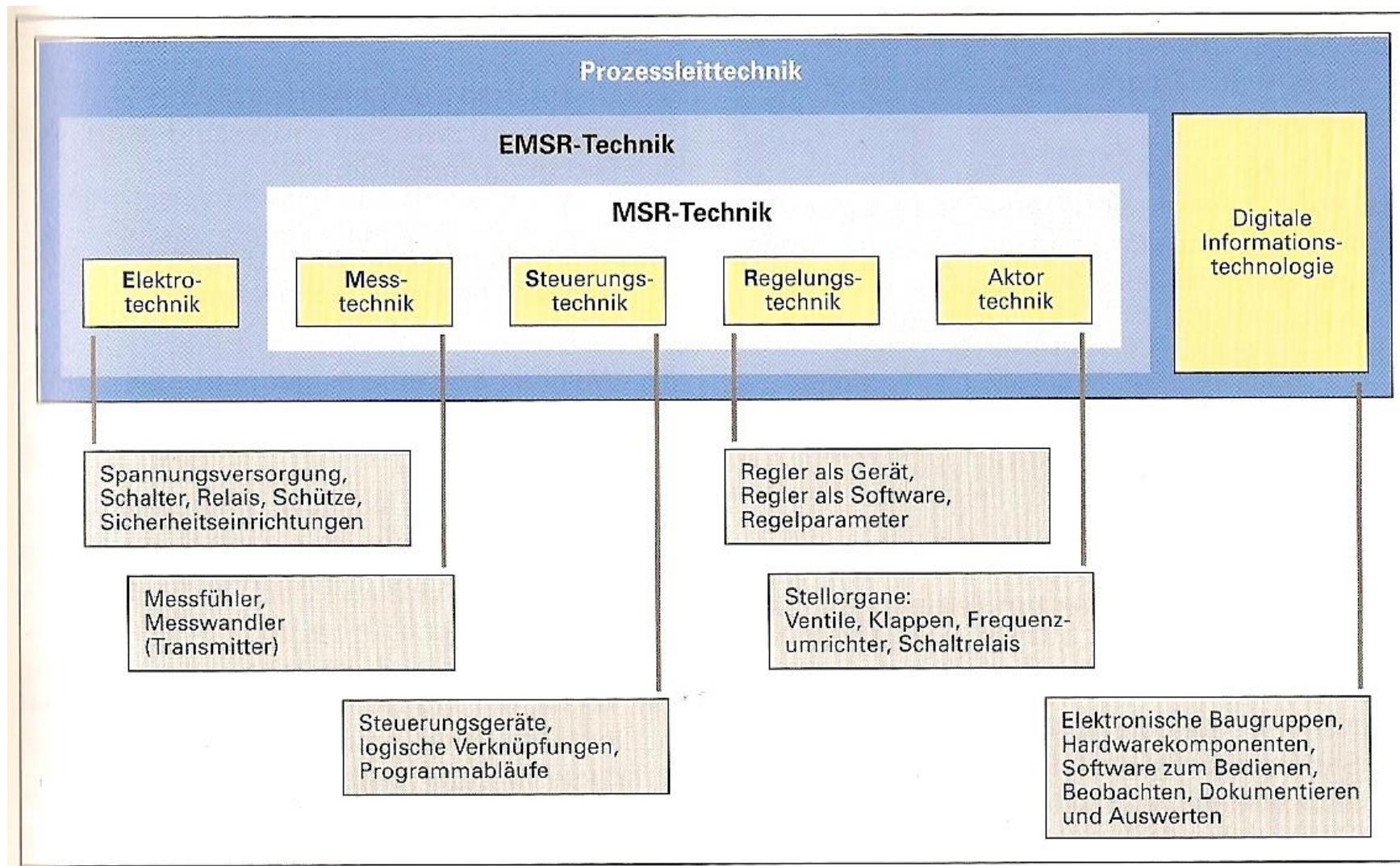
Mensch – Leittechnik - Prozess



Struktur von Prozessen und Leittechnik



Teilgebiete der Prozessleittechnik



Produktionsablauf

Abb. a)

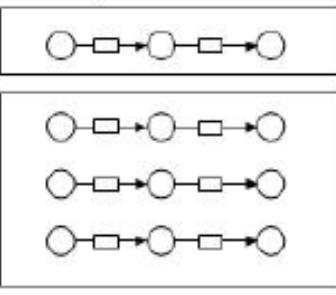


Abb. b)

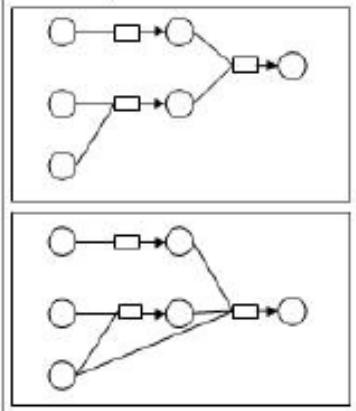


Abb. c)

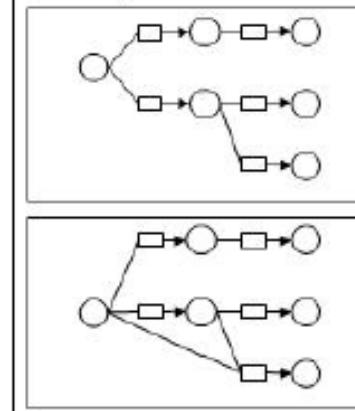


Abb. d)

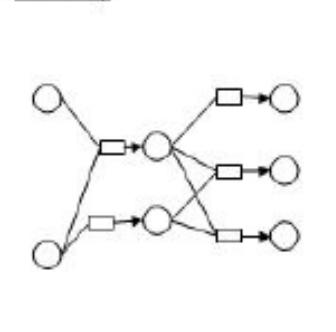


Abb. e)

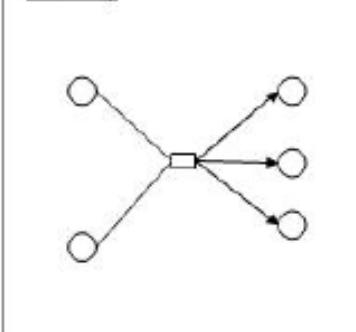
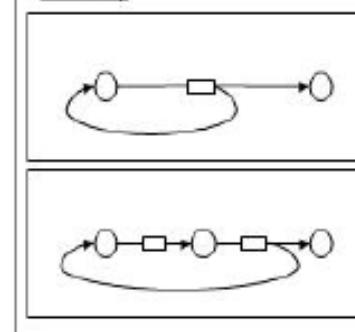


Abb. f)



Legende:

(O) Produkt

→ Reihenfolgebeziehung

(Pfeil) Herstellungsprozeß

- a) durchgängig
(seriell, parallel)
- b) konvergierend
- c) divergierend
- d) allg. Netz
(a – c
kombiniert)
- e) umgruppierend
(Nebenprodukte)
- f) zyklisch
(Katalysatoren,
Lösungsmittel)

Produktionsablauf

Einstrang/Mehrstrang-Anlage:

Eine bzw. mehrere gleiche Produktionsstraßen parallel:
z.B. Spinnereibetrieb

Einprodukt/Mehrprodukt-Anlage:

Mehrere Zuführungen und Abführungen an einem Hauptapparat
z.B. Destillationskolonne

Batch/Konti-Produktion:

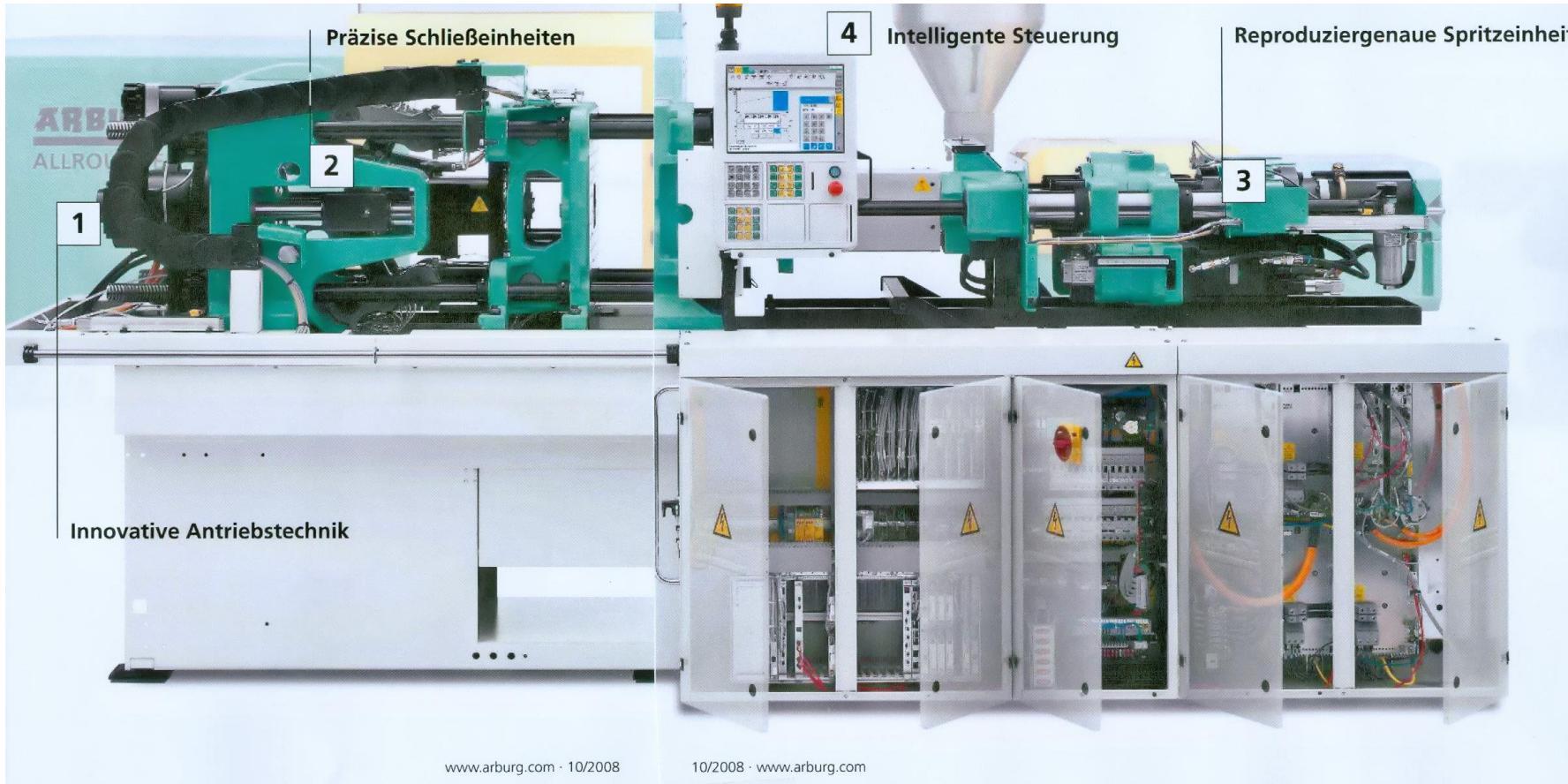
Konti:

längere Produktionszeiträume, An- und Abfahrhilfen,
Protokollierung je Schicht
z.B. Rohr-Extrusion

Batch: Produktion in Zyklen, flexible Umstellung,
Protokollierung je Batch (Chargenprotokoll)
z.B. Lack-, Schmierölproduktion

Prozessleittechnik

Kunststoffverarbeitungsmaschine

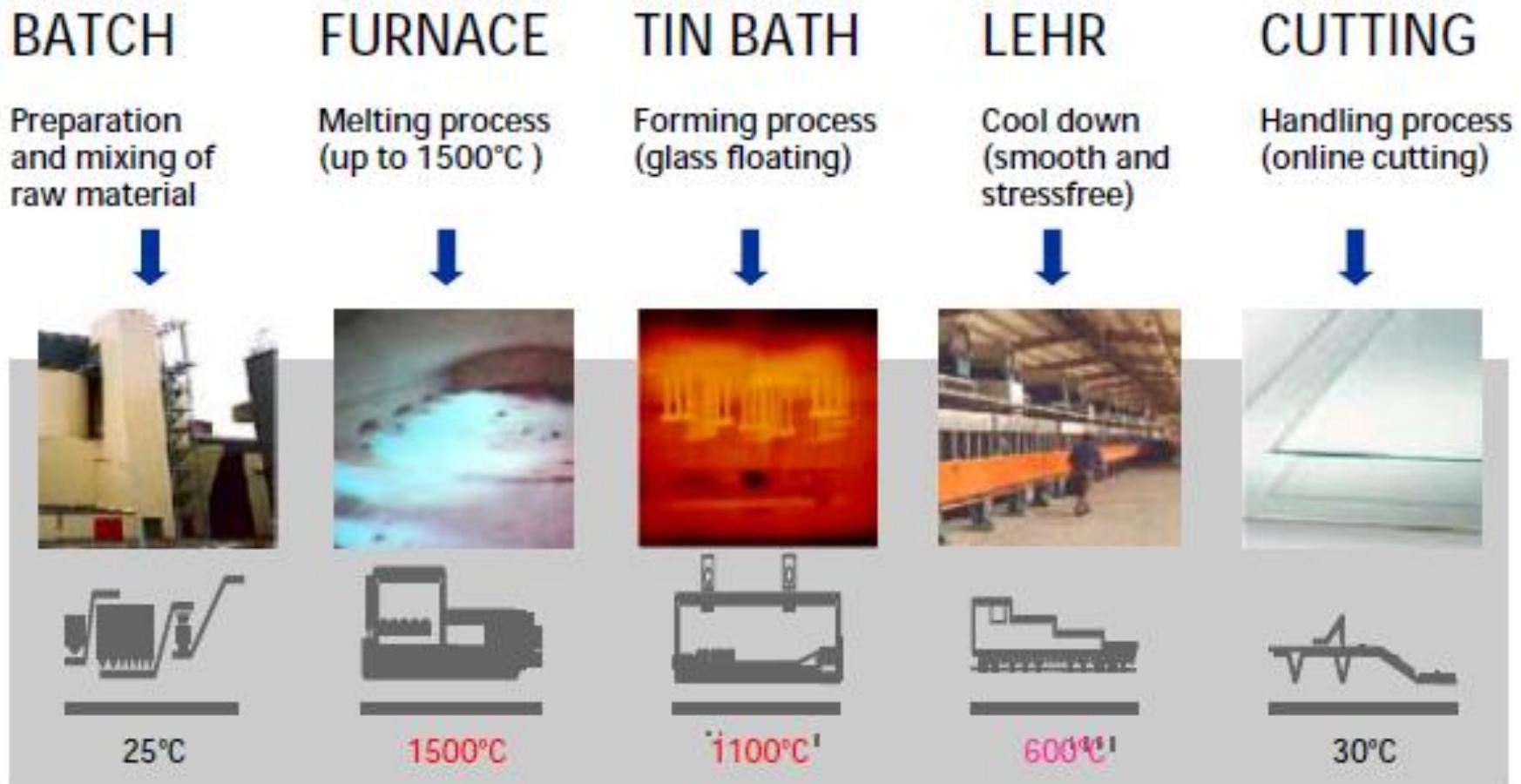


Wickelprozess

Gewebeproduktion

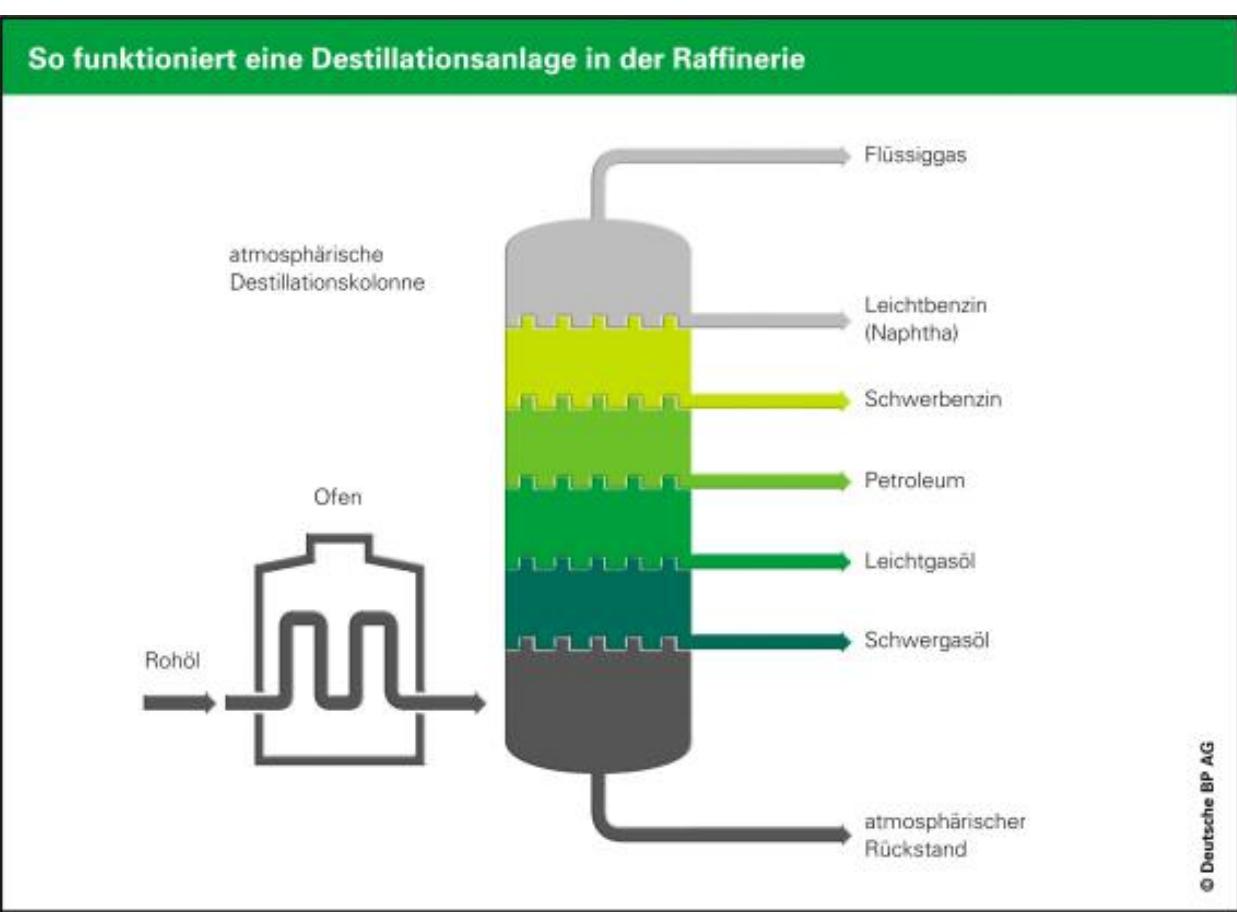
Prozessleittechnik

Floatglasanlage



Prozessleittechnik

Rohöldestillation



Fertigungsleittechnik

Montage Hinterachsgehäuse Sprinter



[mit freundlicher Genehmigung der Daimler AG Nutzfahrzeuge]

Vielfalt

- mehr als 80 verschiedene kundenspezifische Getriebevarianten je nach Fahrstrecke/Beladung/Fahrleistung

Produktivitätssteigerung

- 2003: 6800 Mitarbeiter, 20.000 Achsen pro Jahr
- 2008: 3200 Mitarbeiter, 270.000 Achsen pro Jahr

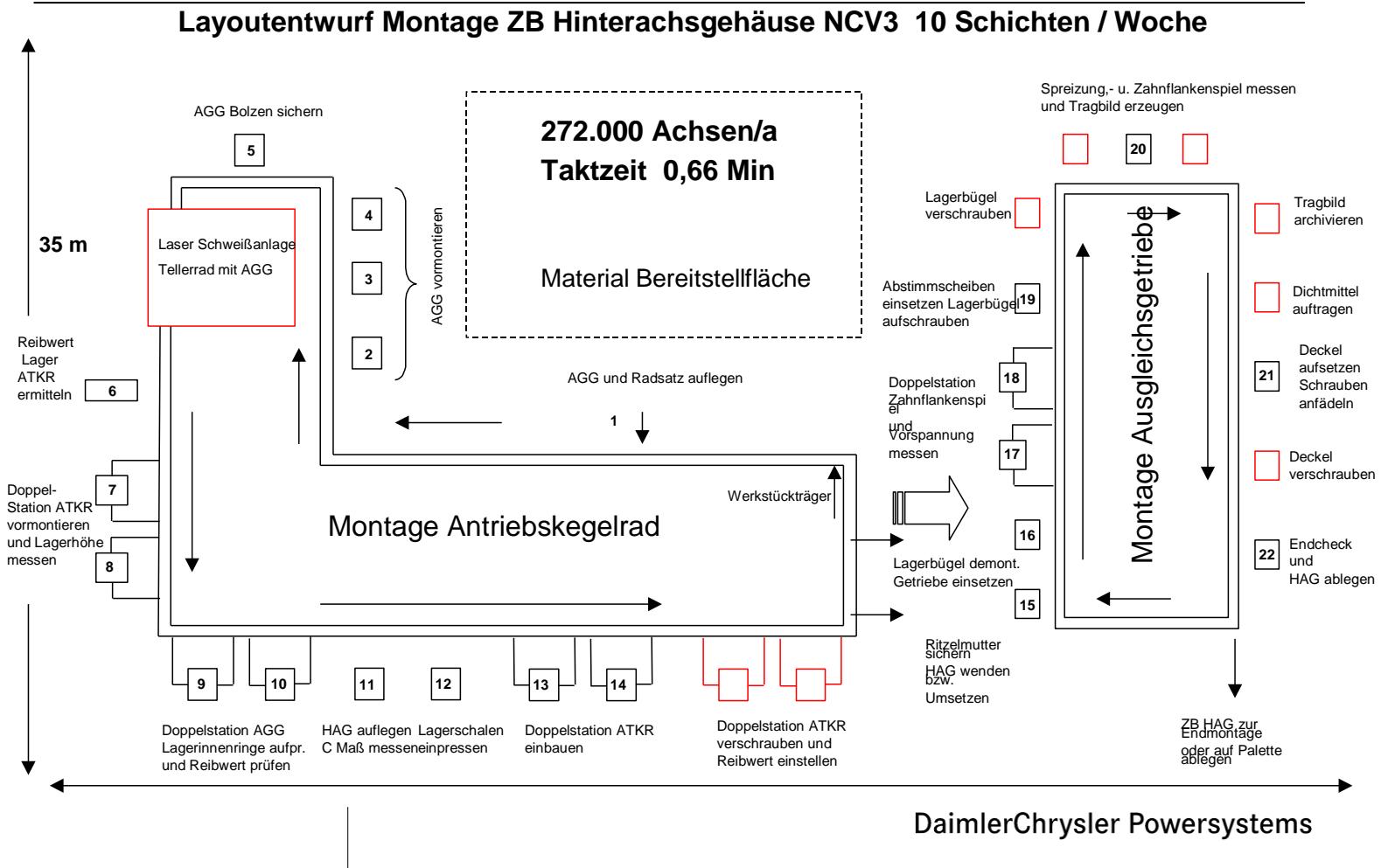
Qualitätssteigerung

- mehr als 500.000 km Laufleistung

Fertigungsleittechnik

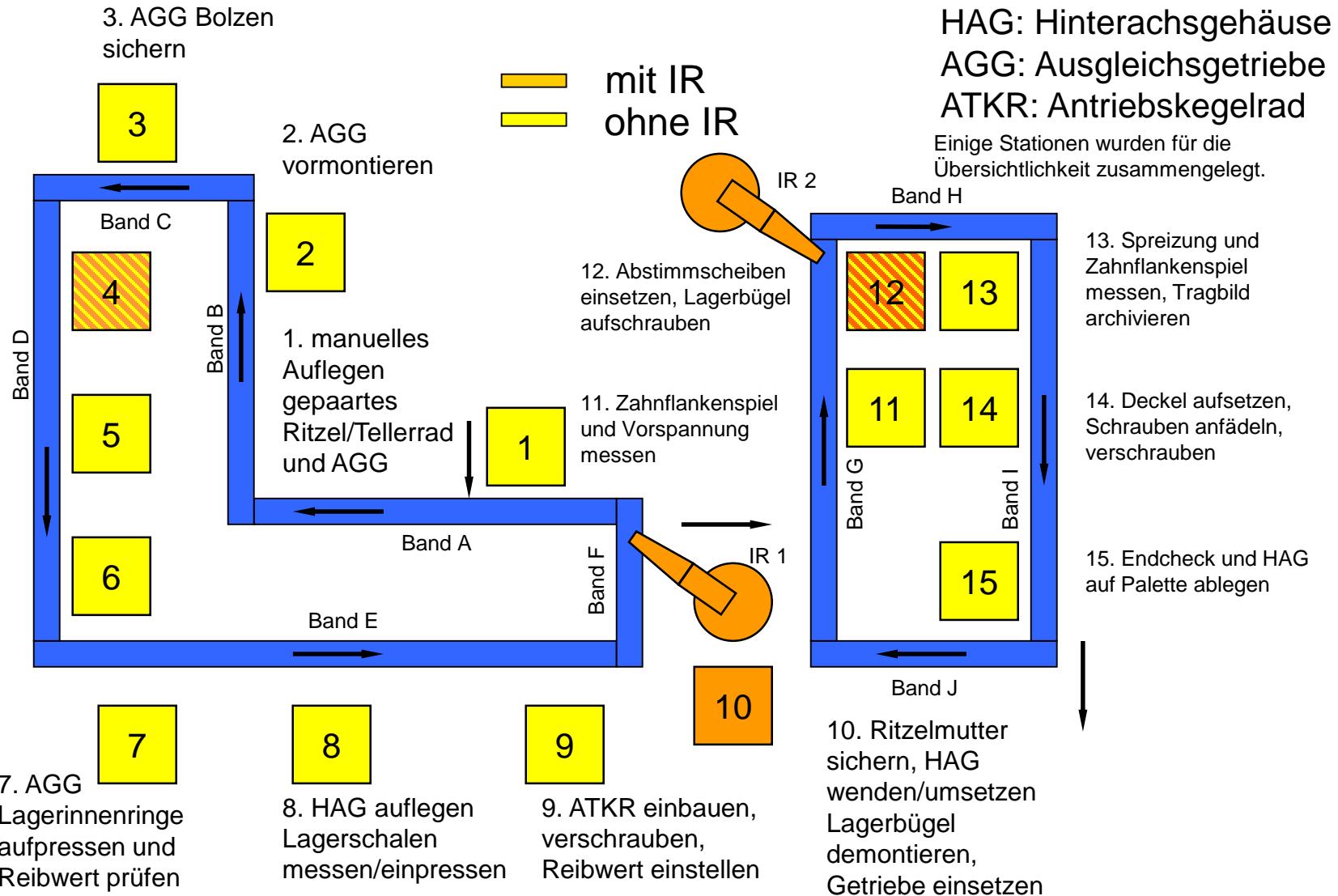
Montage Hinterachsgehäuse Sprinter

DAIMLERCHRYSLER



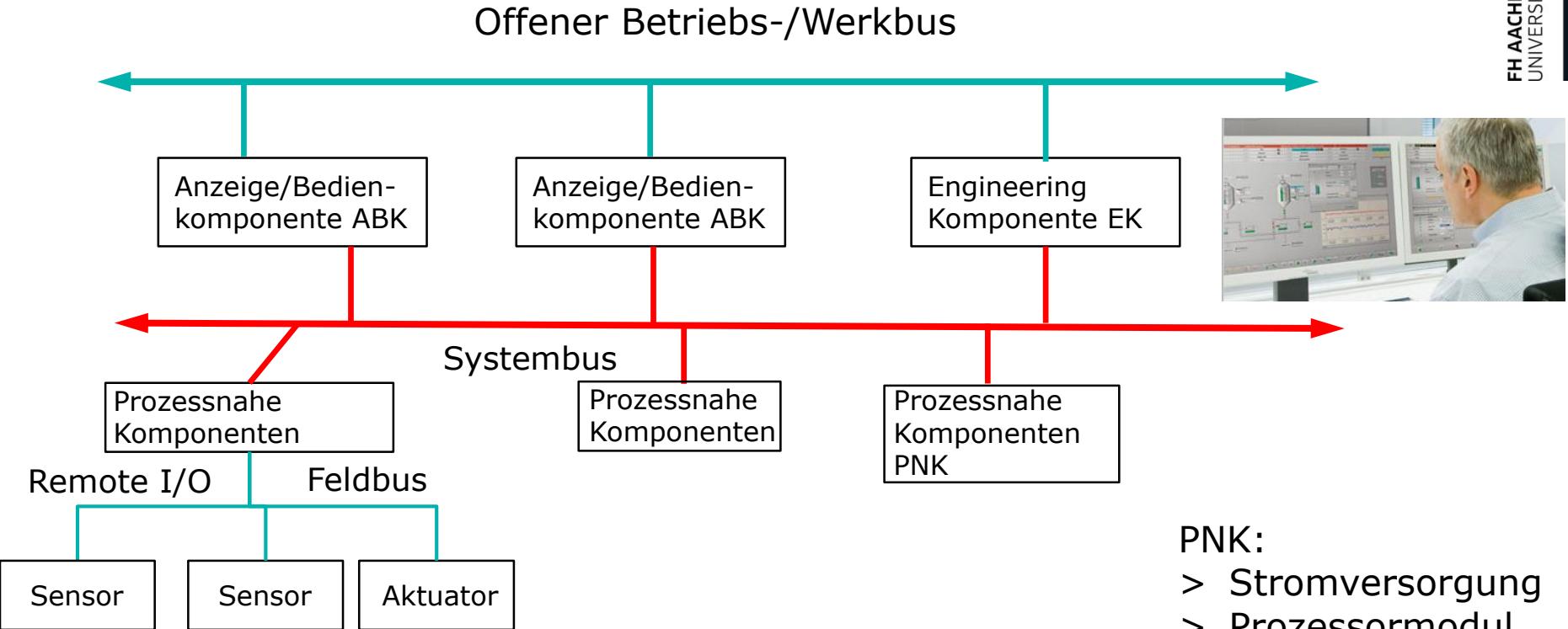
Fertigungsleittechnik

Montage Hinterachsgehäuse Sprinter



Prozessleittechnik

Dezentrales Prozessleitsystem



- PNK:
- > Stromversorgung
 - > Prozessormodul
 - > Schnittstellen
 - > SPS (Anbindung)
 - > I/O Module

Feldbus und Systembus: echtzeitfähig
Betriebsbus: OPC, Internet-Technologien (TCP/IP)

Prozessleittechnik aus der Sicht eines Herstellers



Prozessleittechnik

Power and productivity
for a better world™ 

Prozessleitsystem (PLS)

Eigenschaften

Echtzeitfähigkeit

Innerhalb einer vorgegebenen Zeitspanne muss auf Ereignisse eines technischen Prozesses reagiert werden.

Hohe Verfügbarkeit durch Redundanz

Zur Erhöhung der Verfügbarkeit müssen Teile des PLS gezielt redundant ausgeführt werden, so dass im Störungsfall die Funktion der Komponente unterbrechungsfrei von einer Reservekomponente übernommen werden kann.

Prozessleitsystem (PLS)

Eigenschaften

Offenheit und Interoperabilität

Schnittstellen werden offengelegt und Komponenten unterschiedlicher Hersteller können ohne Zusatzaufwand miteinander betrieben werden.

Durchgängigkeit

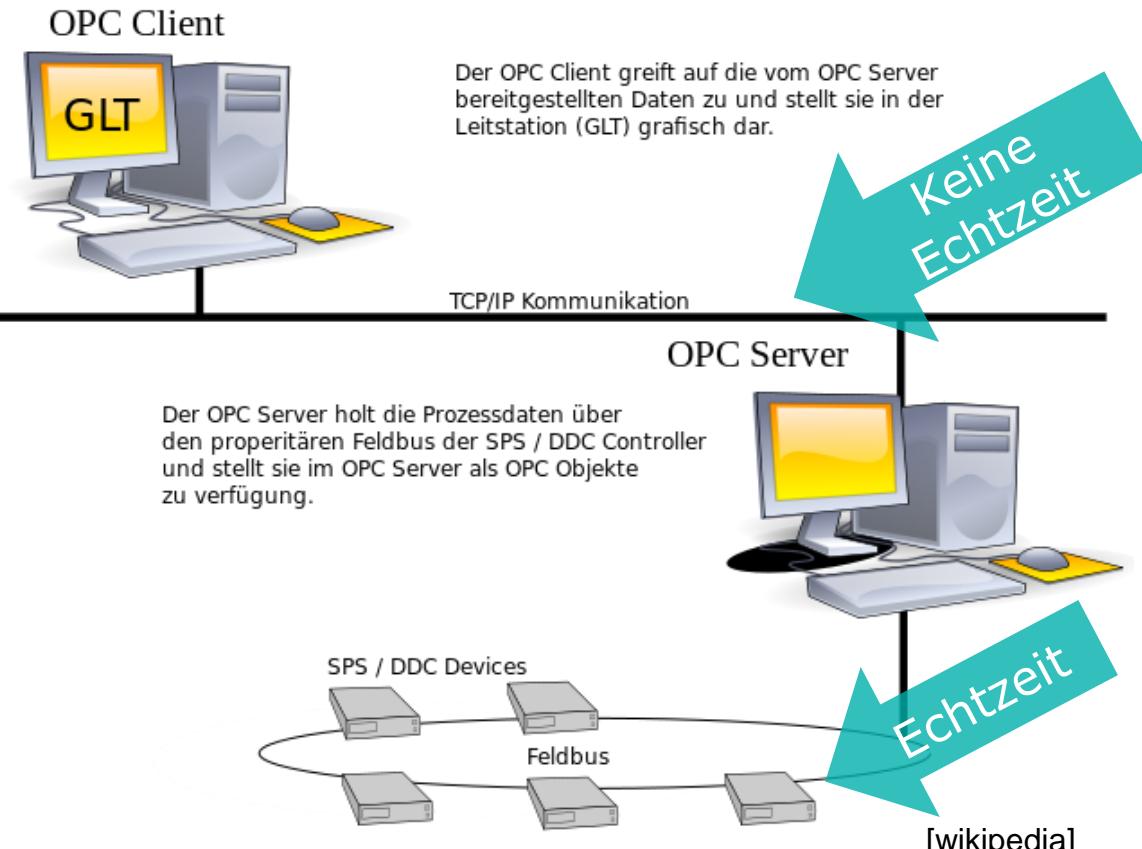
Eine Prozessinformation muss ohne Zusatzarbeit jederzeit zugänglich sein.

Und noch ein Hersteller...



Automatisierungstechnik

Datenaustausch über OPC, OPC UA



[https://de.wikipedia.org/wiki/OPC_Unified_Architecture]

Open Data Communication

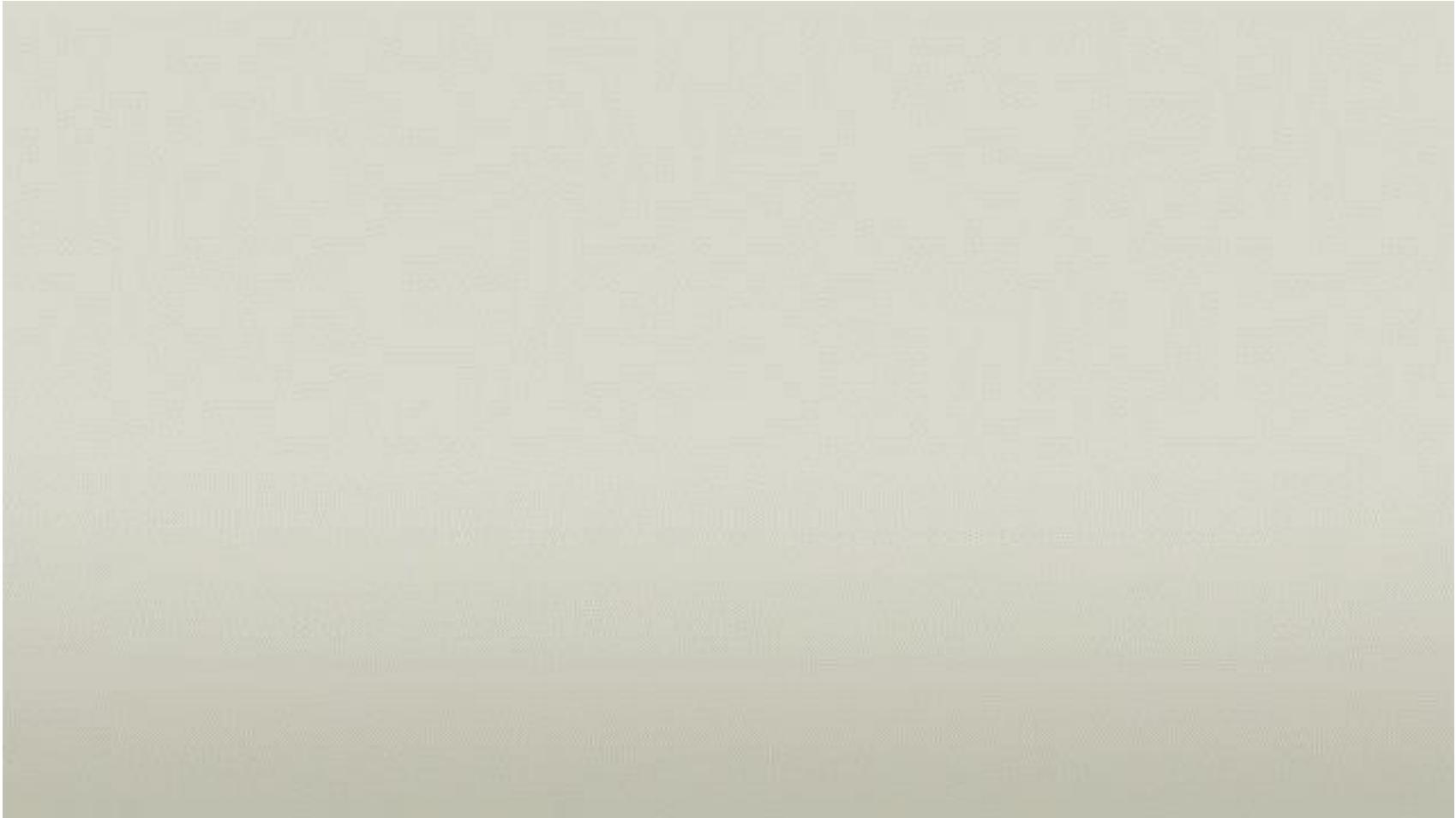
Früher: OLE for process control, später: Open process control

Datenaustausch mit Komponenten der Automatisierungstechnik

OPC UA: Unified Architecture

OPC UA TSN: probiert den Standard auf Echtzeitfähigkeit (Server/Client) zu erweitern. TSN: time sensitive networking

Ein paar Erläuterungen zu OPC UA

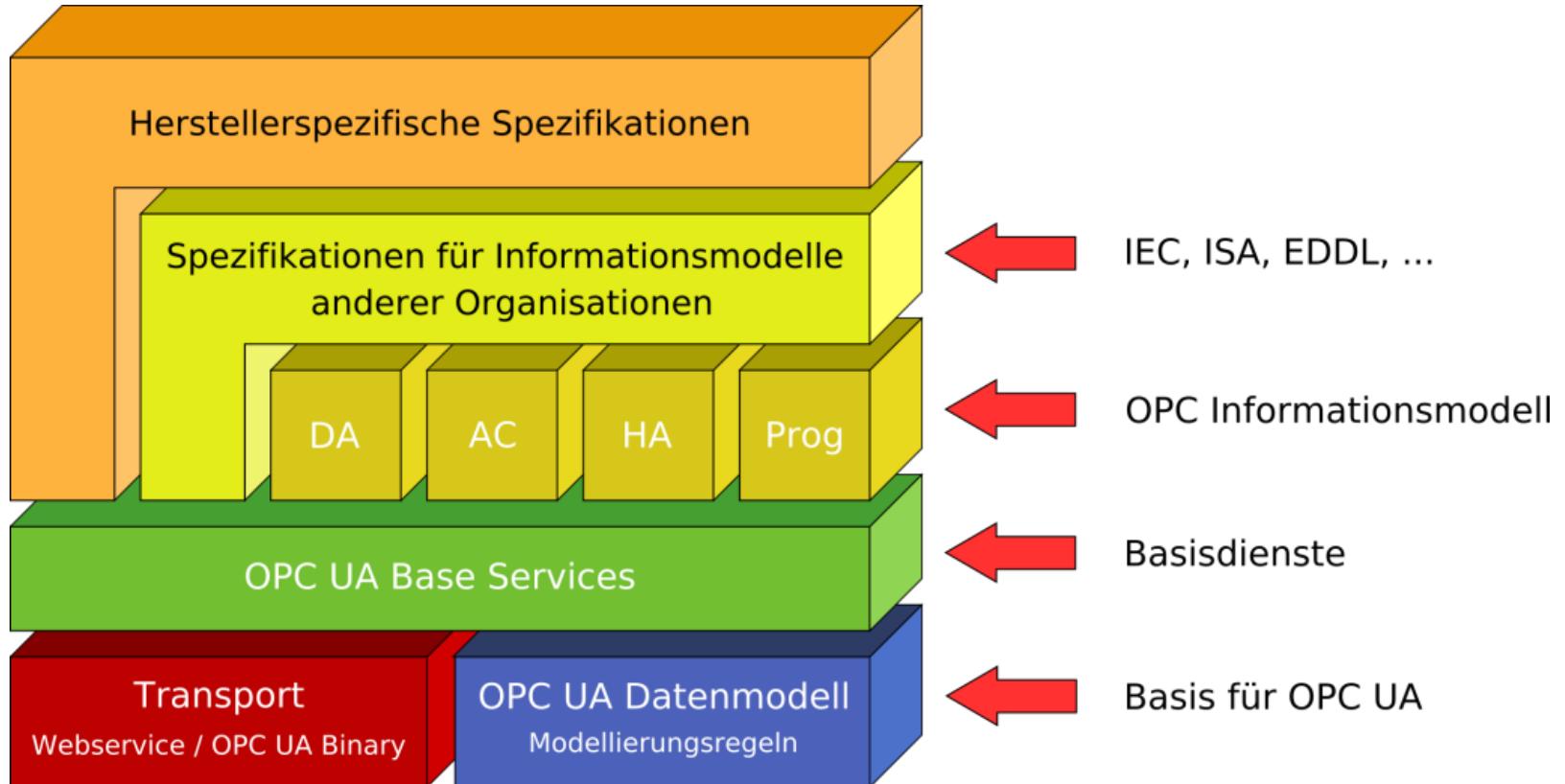


Weitere Infos zu OPC UA



Automatisierungstechnik

Datenaustausch über OPC, OPC UA



Von Gerhard Gappmeier - ascolab GmbH, CC BY-SA 3.0,
<https://de.wikipedia.org/w/index.php?curid=1892069>

DA: Data Access, AC: Alert Event

EDDL: Electronic Device Description Language

Zusätzliche „BWL“-Komponenten

MES: Manufacturing Execution System,
Produktionsleitsystem, Betriebsführungssystem:

- > Führung, Lenkung, Steuerung und Kontrolle in Echtzeit (!),
- > Datenerfassung von Maschinen und Personal,
- > Durchsetzung von bestehender, gültiger Planung,
- > arbeitet mit ERP zusammen.

LIMS: Laborinformations- und Managementsysteme

PIMS: Prozessdateninformations- und
Managementsysteme

Prozessleitsystem (PLS)

Betriebsleittechnik

ERP: Enterprise Resource Planning, Planung von:

- > Kapital, Materialfluss, Lager,
- > Auftragsabwicklung,
- > Personal,
- > Informationen und Kommunikationstechnik.

Aufgabe ist der effiziente betriebliche Wertschöpfungsprozess und dessen Optimierung. Keine Echtzeitfähigkeit.

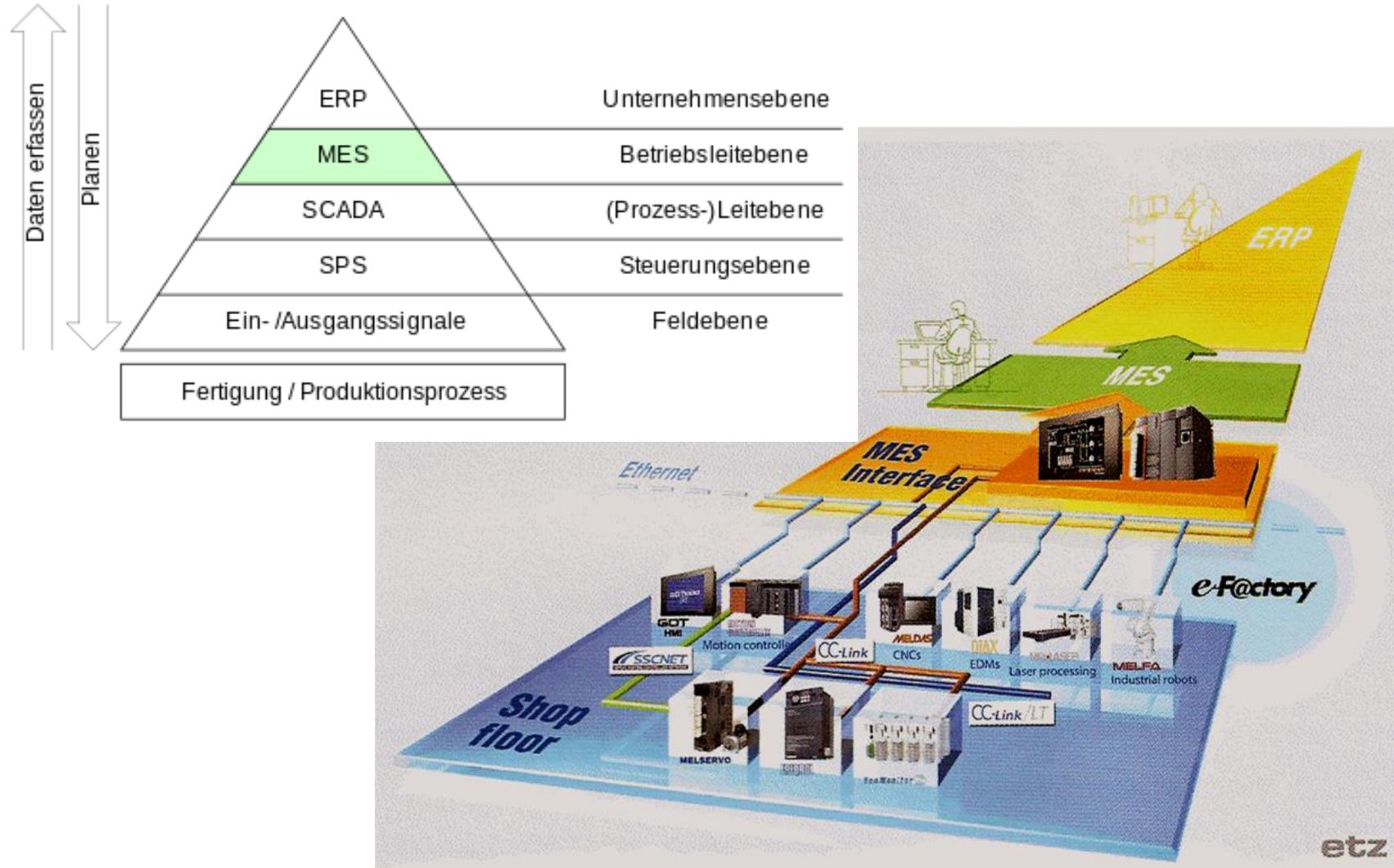
„Nach unten ausgebaut“

Erweiterte Automatisierungssysteme

- > Prozessleittechnik,
- > Materialfluss-,
- > Lagersteuerung

„Nach oben ausgebaut“

Prozessleittechnik Datenhierarchie

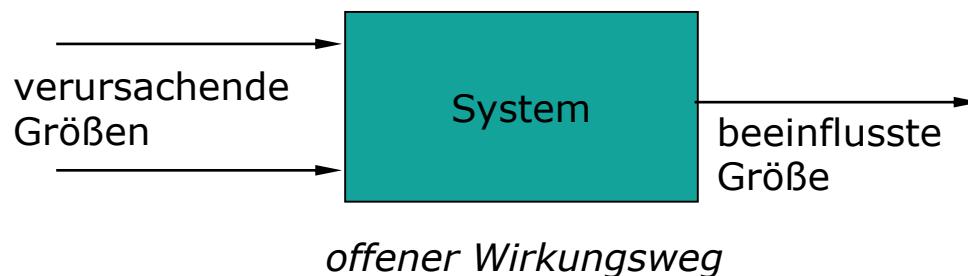


Steuerungstechnik

Definition

„**Steuern** ist der Vorgang in einem System, bei dem **Eingangsgrößen** (verursachende Größen) **Ausgangsgrößen** (beeinflusste Größen) **beeinflussen**“ [DIN 19226-1].

- > offener Wirkungsweg, z.B. Licht einschalten
- > oder geschlossener Wirkungsweg, jedoch ohne fortlaufende Rückwirkung auf dieselben Eingangsgrößen, z.B. Heizöltank füllen bis Vollmeldung, teilweise Übergang zum Regeln
- > **Steuerung** bezeichnet auch die eingesetzte Hard- und Software, z.B. SPS, obwohl auch Regelung mit SPS realisierbar.

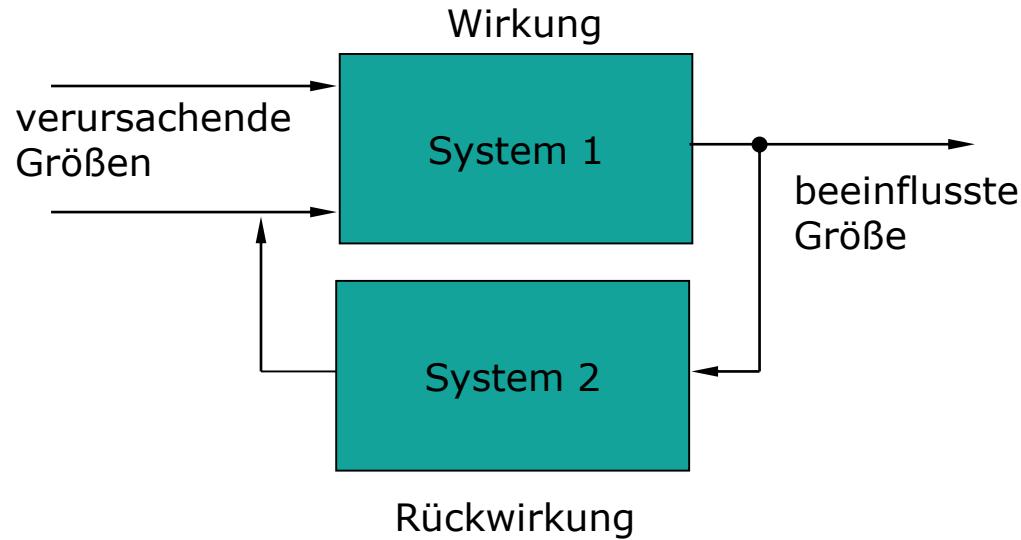


Steuerungstechnik

Definition, Abgrenzung

„**Regeln** ist ein Vorgang, bei dem fortlaufend die **Regelgröße** (beeinflusste Größe, Ist-Größe) mit der **Führungsgröße** (verursachende Größe, Soll-Größe) **verglichen** und angleichend **beeinflusst** wird“ [DIN 19226-1]

- > geschlossener Wirkungsweg (Regelkreis), z.B. PKW-Geschwindigkeit durch Tempomat konstant halten
- > Mensch kann Glied des Regelkreises sein, z.B. PKW durch Gaspedal konstant halten



Steuerungstechnik

Definition, Abgrenzung

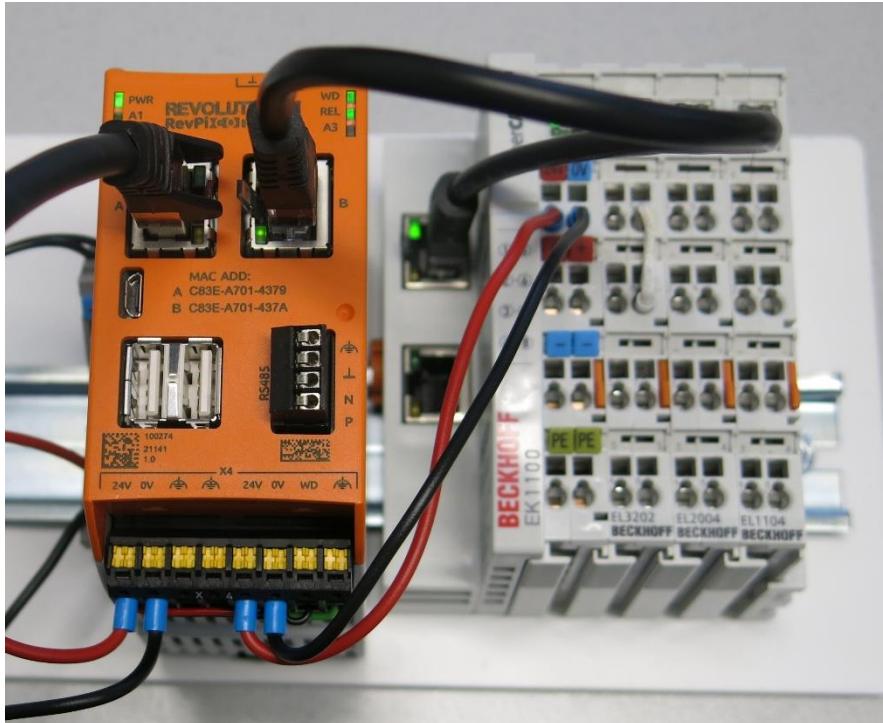
„**Automatisierung** ist das Ausrüsten einer Einrichtung, so dass sie ganz oder teilweise **ohne** Mitwirkung des Menschen **bestimmungsgemäß arbeitet**“ [DIN 19233].

In der Elektrotechnik und Verfahrenstechnik wird Automatisierung mit MSR (Messen, Steuern, Regeln) gleichgesetzt.

“Eine **Speicherprogrammierbare Steuerung** (SPS; Programmable Logic Controller, **PLC**; Steuer-, Automatisierungsgerät) ist ein digital arbeitendes, elektronisches System für industrielle Umgebung. Es verwendet einen **programmierbaren** Speicher für Anweisungen, um durch digitale oder analoge Ein- und Ausgänge Maschinen oder **Prozesse** zu **steuern**“ [DIN 61131-1].



Speicherprogrammierbare Steuerungen Bauarten, „Scheibenbasierte Systeme“



B&R SPS

Modulare SPS mit Klemmen

Die einzelnen Module werden als Klemmen montiert und über ein Bussystem verbunden. Links ein System mit Beckhoff Klemmen und einem Revolution PI, der als EtherCAT Master fungiert.

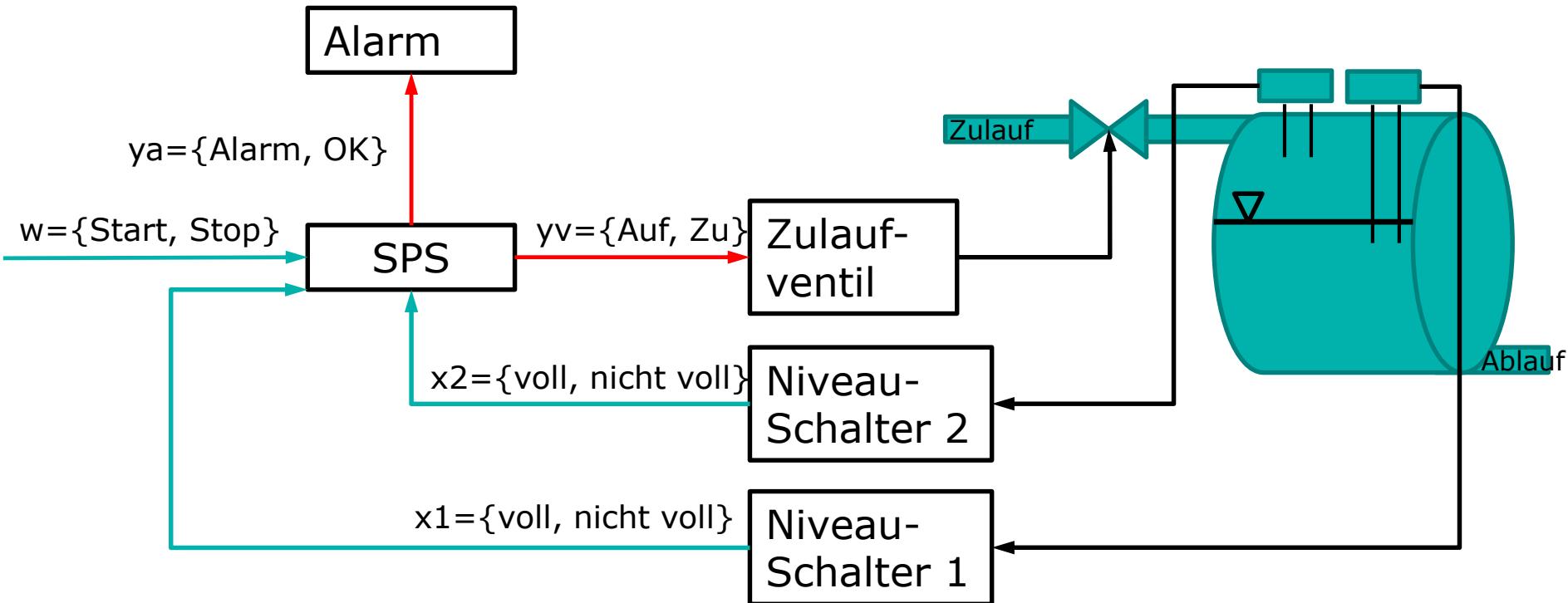


Wago SPS ähnliches System wie oben.
Ebenfalls auf Hutschienen montierbar. Modular über Klemmen erweiterbar.

Steuerungstechnik

Technologieschema

Wenn ($x_1=\text{voll}$) und ($w=\text{Start}$) dann $y_v=\text{Zu}$
Wenn ($x_2=\text{voll}$) dann $y_a=\text{Alarm}$
Wenn ($x_1=\text{nicht voll}$) und ($w=\text{Start}$) dann $y_v=\text{Auf}$
Wenn $x_2=\text{nicht voll}$ $y_a=\text{OK}$
Wenn $w=\text{Stop}$ dann $y_v=\text{Zu}$



Steuerungstechnik

Technologieschema

Wenn ($x_1 = \text{voll}$) und ($w = \text{Start}$) dann $y_v = \text{Zu}$

Wenn ($x_2 = \text{voll}$) dann $y_a = \text{Alarm}$

Wenn ($x_1 = \text{nicht voll}$) und ($w = \text{Start}$) dann $y_v = \text{Auf}$

Wenn $x_2 = \text{nicht voll}$ $y_a = \text{OK}$

$$1 : X + 1 = 1 \\ 1' : X \cdot 0 = 0$$

$$2 : X + 0 = X \\ 2' : X \cdot 1 = X$$

$$3 : X + X = X \\ 3' : X \cdot X = X$$

$$4 : \overline{(X)} = X \\ 4' : \overline{(X)} = X$$

$$5 : X + Y = Y + X \\ 5' : X \cdot Y = Y \cdot X$$

$$6 : X + \overline{X} = 1 \\ 6' : X \cdot \overline{X} = 0$$

$$7 : X + XY = X \\ 7' : X(X + Y) = X$$

$$8 : (X + \overline{Y})Y = XY \\ 8' : X\overline{Y} + Y = X + Y$$

$$9 : X + Y * Z = (X + Y) + Z = X + (Y + Z) \\ 9' : XYZ = (XY)Z = X(YZ)$$

Assoziative
Gesetze

$$10 : XY + XZ = X(Y+Z) \\ 10' : (X + Y)(X + Z) = X + YZ$$

Distributive
Gesetze

$$11 : (X + Y)(Y + Z)(Z + \overline{X}) = (X + Y)(Z + \overline{X}) \\ 11' : XY + YZ + ZX = XY + XZ$$

$$12 : (X + Y)(\overline{X} + Z) = XZ + \overline{X}Y$$

$$13 : \overline{(X + Y + Z + \dots)} = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \dots \\ 13' : \overline{X \cdot Y \cdot Z \dots} = \overline{X} + \overline{Y} + \overline{Z} + \dots$$

De Morgan

Rechenregeln [WECK4]

	UND	ODER	NICHT																																				
Schaltalgebra	$Y = X_1 \wedge X_2$	$Y = X_1 \vee X_2$	$Y = \overline{X} = \neg X$																																				
Funktions-tabelle, Wertetafel	<table border="1"> <tr> <th>X₁</th> <th>X₂</th> <th>Y</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	X ₁	X ₂	Y	0	0	0	1	0	0	0	1	0	1	1	1	<table border="1"> <tr> <th>X₁</th> <th>X₂</th> <th>Y</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	X ₁	X ₂	Y	0	0	0	1	0	1	0	1	1	1	1	1	<table border="1"> <tr> <th>X</th> <th>Y</th> </tr> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </table>	X	Y	0	1	1	0
X ₁	X ₂	Y																																					
0	0	0																																					
1	0	0																																					
0	1	0																																					
1	1	1																																					
X ₁	X ₂	Y																																					
0	0	0																																					
1	0	1																																					
0	1	1																																					
1	1	1																																					
X	Y																																						
0	1																																						
1	0																																						
Funktionspla-n (FUP)																																							
Kontaktplan (KOP)																																							
Zeitablauf-diagramm (ZAD)																																							

if x_1 AND w then NOT y_v
 if x_2 then y_a
 if NOT x_1 AND w then y_v
 if NOT x_2 then NOT y_a
 if y_a then NOT y_v

Steuerungstechnik

Wertetabelle

Wenn ($x_1 = \text{voll}$) und ($w = \text{Start}$) dann $y_v = \text{Zu}$ {unabhängig von w }
 Wenn ($x_1 = \text{voll}$) und ($w = \text{Stop}$) dann $y_v = \text{Zu}$
 Wenn ($x_2 = \text{voll}$) dann $y_a = \text{Alarm}$
 Wenn ($x_1 = \text{nicht voll}$) und ($w = \text{Start}$) dann $y_v = \text{Auf}$
 Wenn $x_2 = \text{nicht voll}$ $y_a = \text{OK}$
 Wenn $w = \text{Stop}$ dann $y_v = \text{Zu}$

if x_1 AND w then NOT y_v
 if x_1 AND NOT w then NOT y_v

-> if x_1 then NOT y_v

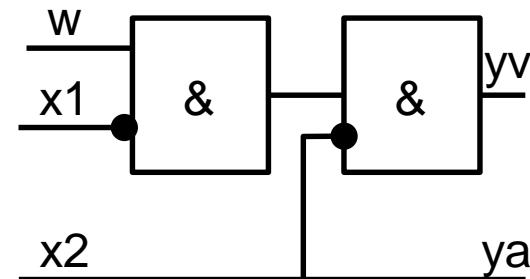
if x_2 then y_a
 if NOT x_2 then NOT y_a {gleich}

if NOT x_1 AND w then y_v

if y_a then NOT y_v

if NOT w then NOT y_v

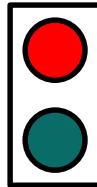
w	x1	x2	yv	ya
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1 -> 0	1
1	1	0	0	0
1	1	1	0	1



Steuerungstechnik

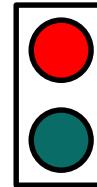
Beispiel: Fußgänger-Ampel

Programmtechnische Realisierung (IEC 61131)

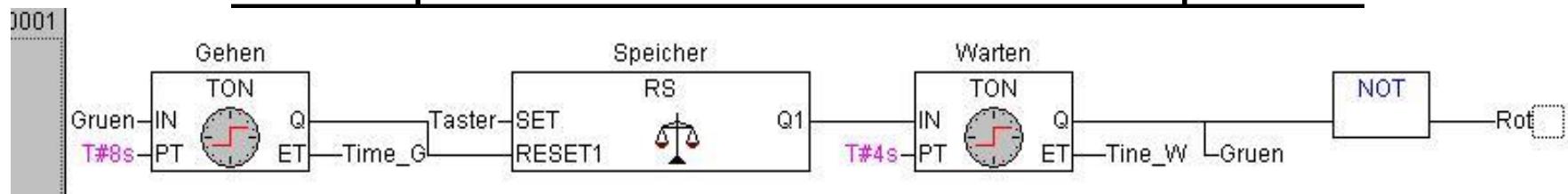


Deklaration:

```
0001 PROGRAM PLC_PRG
0002 VAR
0003   Taster: BOOL := 0;
0004   Speicher: RS ;
0005   Warten: TON ;
0006   Gruen: BOOL := 0;
0007   Gehen: TON ;
0008   Rot: BOOL := 0;
0009   Time_G: TIME;
0010   Time_W: TIME;
0011
0012 END_VAR
```



Funktionsrumpf:



Steuerungstechnik

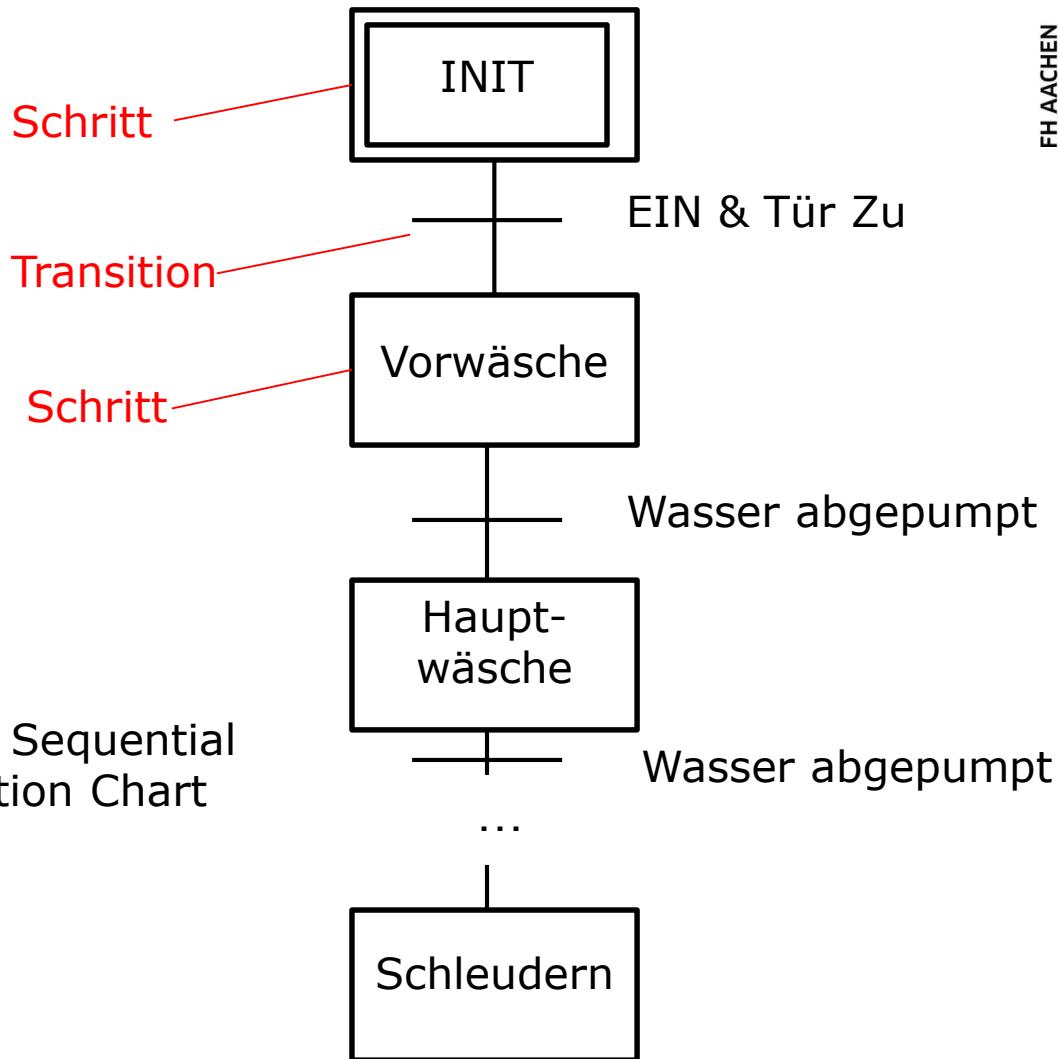
Ablaufsteuerung Waschmaschine



[Electrolux Hausgeräte
Vertriebs GmbH, Nürnberg]

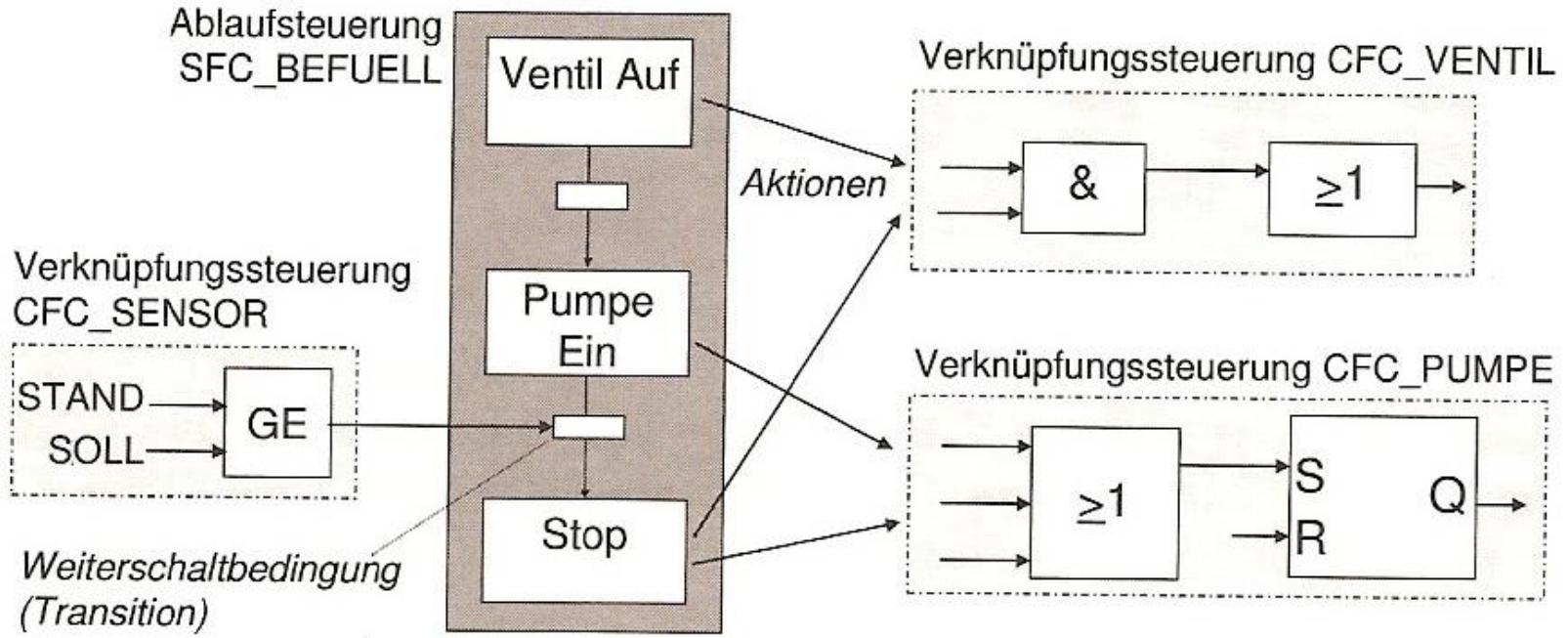
- > Wasser einlaufen
- > Heizen
- > Trommel drehen
- > Lauge abpumpen
- > schleudern

SFC: Sequential
Function Chart



Steuerungstechnik

Ablaufsteuerung Detail: Befüllen

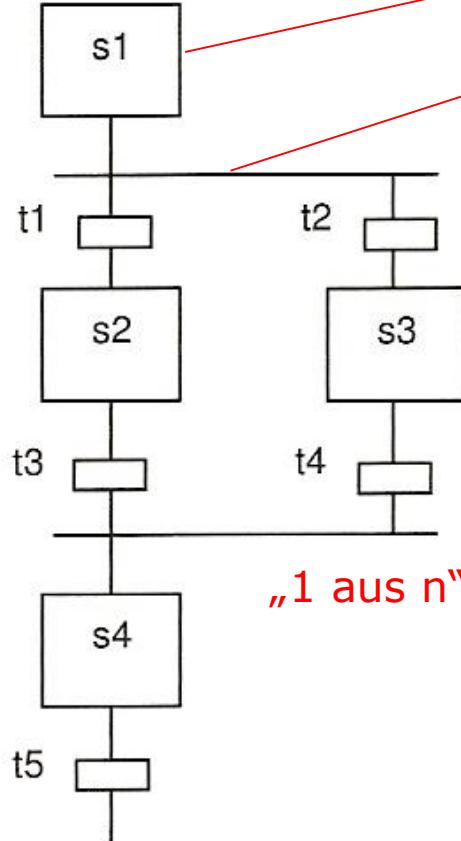


CFC: Continuous
Function Chart

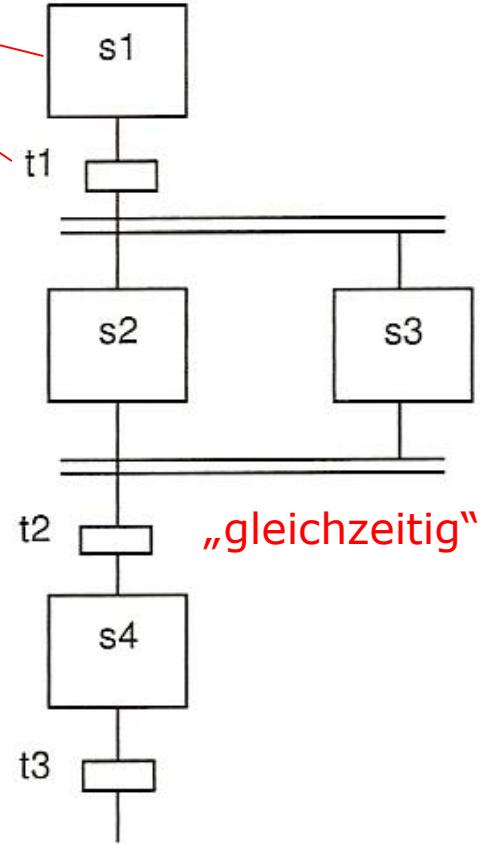
Steuerungstechnik

Ablaufsteuerung: Strukturen

a) Alternativverzweigung



b) Parallelverzweigung



Steuerungstechnik

Datentypen

Datentyp	Größe	Wertebereich	Beispiel
BOOL	1 Bit	False/True	False
BYTE	8 Bit	0/255	42
WORD	16 Bit	0/65535	4242
DWORD	32 Bit	0/4294967296	424242
CHAR	8 Bit	0/255 -128/127	-42
INT	16/32 Bit	+/-2147483648	-4242
UINT	16/32 Bit	0/4294967296	424242
LONG INT	64 Bit	+/-9223372036854775807	-424242
FLOAT/REAL	32 Bit	1 Sign, 8 Exp, 24 Mantisse	42.42
DOUBLE	64 Bit	1 Sign, 11 Exp, 53 Mantisse	42.4204
STRING	char array		Text:

Datenstrukturen

Struktur als Datentyp

Komplexe Datentypen lassen sich als Strukturen aufbauen.
Sie bestehen aus Komponenten, die wieder Standard-Datentypen sind:

```
struct student{ /* deklariert den Strukturtyp student */
    char alter;
    int matrikelnr;
    float gewicht;
    char name[25];
};

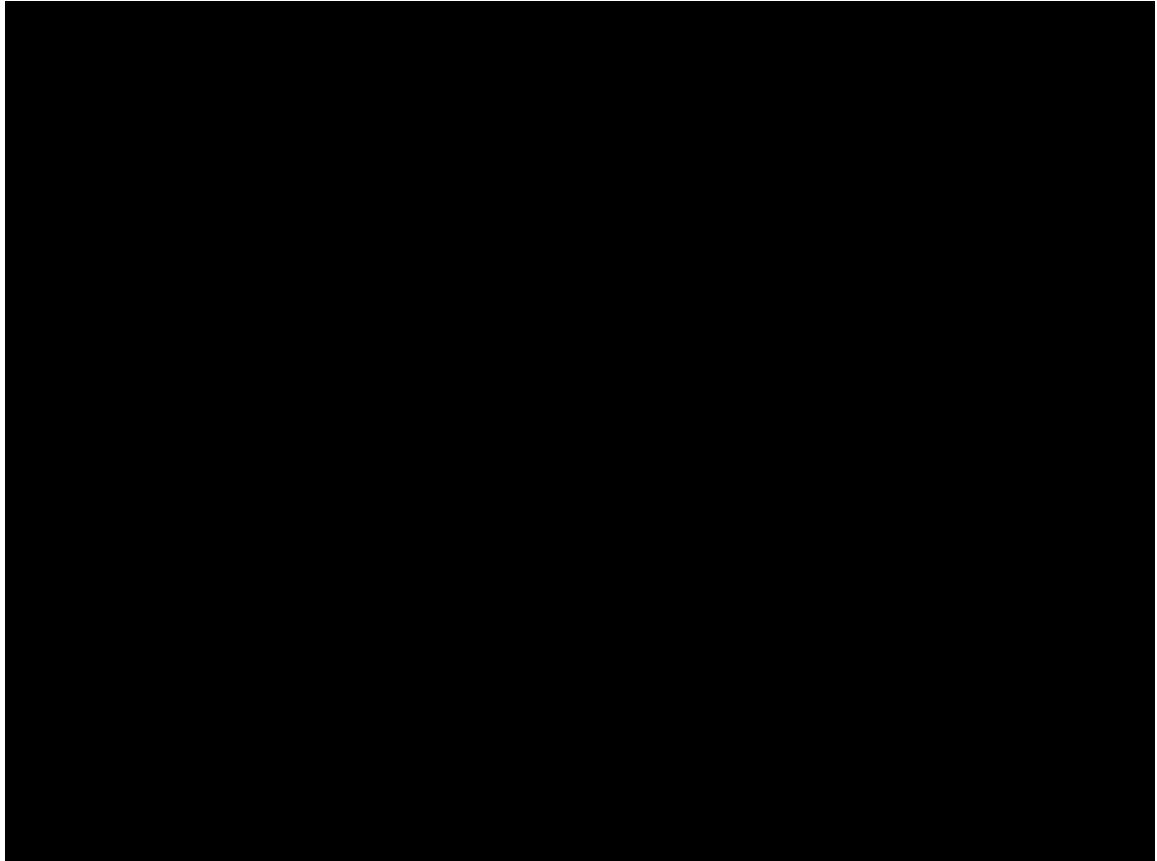
struct student kalle;
kalle.alter=28;
```

Werden zusätzlich Methoden innerhalb der Datenstruktur benötigt kann man objektorientierte Sprachen benutzen, z.B. C++

```
class student {
public:
    int matrikelnr;
    int alter;
    float gewicht;
    void erhoehe_alter();
}
```

Objektorientierte Programmierung

Videogames, die Faszination Computer...



Objektorientierte Programmierung

Videogames und Klassen

```
class asteroid{  
public:  
    int Size;  
    int color;  
    float velocity;  
    void move();  
    void explode();  
    void sound_explode();  
}
```

Wie sieht die Klasse UFO aus?

Sehr ähnlich!

Es werden heutzutage als „höhere“ Programmiersprachen fast ausschließlich objektorientierte Sprachen eingesetzt. Diese unterscheiden sich durch die Einführung von Klassen für Objekte von z.B. Standard ANSI C, älteren Sprachen, wie Fortran oder Pascal.

```
class UFO{  
public:  
    int Size;  
    int color;  
    float velocity;  
    void move();  
    void explode();  
    void sound_explode();  
    void accelerate();  
    void decelerate();  
}
```

Objektorientierte Programmierung

Videogames und Vererbung

Datentypen und Funktionen werden eng miteinander verknüpft, so dass die Objekte Eigenschaften besitzen, die sogar „vererbt“ werden können. So können abstrakte Klassen definiert werden, deren Eigenschaften erst später gefüllt werden:

```
class UFO{  
public:  
    int Size;  
    int color;  
    float velocity;  
    void move();  
    void explode();  
  
    void  
    sound_explode();  
    void accelerate();  
    void decelerate();  
}
```

Alte UFO
Klasse

```
class Flying_Object{  
public:  
    int Size;  
    int color;  
    float velocity;  
    void move();  
    void explode();  
    void sound_explode();  
}
```

Neue Eltern
Klasse

```
class UFO:parent  
Flying_Object{  
public:  
    void accelerate();  
    void decelerate();  
}
```

Neue, vererbte
UFO Klasse

In Python kann ebenfalls objektorientiert programmiert werden. Hier kommt noch der Ausdruck „self“ vor, der eine Referenz zur aufrufenden Instanz darstellt. Die Vererbung erfolgt durch Benutzung der Elternklasse als Argument der neuen Klasse.

```
class asteroid:  
    Size = 0  
    color = 0  
    velocity = 0.0  
  
    def move(self):  
        #move_code  
  
    def explode(self):  
        #explode_code  
  
    def sound_explode(self):  
        #sound_explode_code
```

```
class UFO:  
    Size = 0  
    color = 0  
    velocity = 0.0  
  
    def move(self):  
        #move_code  
  
    def explode(self):  
        #explode_code  
  
    def sound_explode(self):  
        #sound_explode_code  
  
    def accelerate(self):  
        #accelerate_code  
  
    def decelerate(self):  
        #decelerate_code
```

```
class Flying_Object:  
    Size = 0  
    color = 0  
    velocity = 0.0  
  
    def move(self):  
        #move_code  
  
    def explode(self):  
        #explode_code  
  
    def sound_explode(self):  
        #sound_explode_code
```

```
class UFO(Flying_Object):  
    def accelerate(self):  
        #accelerate_code  
    def decelerate(self):  
        #decelerate_code
```



Alte UFO
Klasse



Neue UFO
Klasse

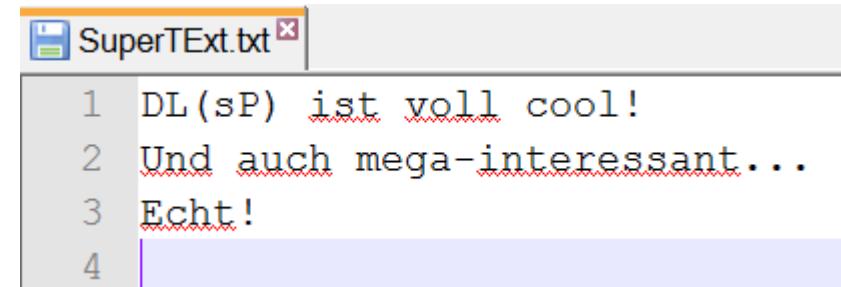


Neue Eltern
Klasse

Ein- und Ausgabe

Dateien Öffnen und Lesen

- > Dateien bestehen aus einer Aneinanderreihung von sequentiellen Bits, die wiederum z.B. zu Bytes zusammengefasst werden.
- > Die Anwendung entscheidet, ob es sich dann bspw. um Text, Bilder oder Musik handelt.
- > Eine einfache Textdatei enthält bspw. vom OS interpretierbare Zeichen, die als Textzeichen ausgegeben werden können.
- > Es werden aber auch weitere Daten, die teilweise nicht sichtbar sind, in die Datei geschrieben (','0D' und ','0A' für CR und LF bzw. '\r\n'). Diese Daten dienten früher zur Steuerung des Ausgabegerätes, jetzt zur Formatierung des Textes.



Datenformate

Ein- und Ausgabe

Dateien Öffnen und Lesen

- > Die Textdatei ist nach ASCII codiert. Mit einem Hex-Editor z.B. HxD, kann man alle vorhandenen Daten sichtbar machen.

ASCII-Zeichentabelle, hexadezimale Nummerierung

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

 HxD Hex Editor

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Dekodierter Text
00000000	44	4C	28	73	50	29	20	69	73	74	20	76	6F	6C	6C	20	DL(sP) ist voll
00000010	63	6F	6F	6C	21	DD	0A	55	6E	64	20	61	75	63	68	20	cool!..Und auch
00000020	6D	65	67	61	2D	69	6E	74	65	72	65	73	73	61	6E	74	mega-interessant
00000030	2E	2E	2E	0D	0A	45	63	68	74	21	0D	0A				Echt!..

Hex-Umrechnung

Kleine Hilfe für die Umrechnung in Hex-Zahlen

- > Die Darstellung von allen Zahlen, die auf Byte Variablen basieren, wird häufig im hexadezimalen oder sedezimalen System durchgeführt. Hier werden immer 4Bits – ein sog. Nibble – betrachtet. Mit 4Bit lassen sich 16 Zustände darstellen, also die Zahlen von 0-15. Da man im Dezimalsystem nur Zahlen von 0-9 darstellen kann, wenn man nicht eine neue Stelle benutzen will, codiert man die dezimale 10 als hexadezimales A, die 11 als B usw. Dadurch kann man nun zwei 4Bit Zahlen als eine zweistellige Hex Zahl zusammenfassen. Man macht mit 0x?? oder mit ??h (??H) Hexadezimalzahlen kenntlich.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	dez	hex
128	64	32	16	8	4	2	1		
0	0	1	0	1	0	1	0	42	0x2A
1	0	1	0	1	0	0	0	168	0xA8
0	1	1	1	1	1	1	1	127	0x7F

Datenformate

Ausflug in die Welt der Dateiformate

Es existieren tausende von Datenformaten. Bitte erklären Sie exemplarisch die folgenden Formate unter den Aspekten:

Welcher Datentyp? Verlustbehaftet? Welche Art von Daten? Quelloffenes Datenformat?

- > Datenformat auf einer Audio CD?
- > Wie funktioniert das MP3 Format?
- > Wer hat JPG erfunden?
- > Was ist das PDF Format?
- > Was macht eine XML Datei?
- > Wofür sind TIF Dateien da?
- > Was ist MPEG-2?

Name	Daten	Quell-Offen	komprimiert	Format	Verbreitung	Anwendung	Technische Funktion
	Audio/Video	Patent?	Ja/nein, Wie?	Binär/Text	häufig		

Übersicht Datenformate

Name	Daten	Quell-Offen	komprimiert	Format	Verbreitung	Anwendung	Technische Funktion
Audio CD, WAV	Audio	ja	Nein	PCM, Pulse Code Modulation Rohdaten	hoch	Audio, Entertainment	Musikarchivierung
MP3	Audio	Patent, aber ausgelaufen	ja	Keine Rohdaten	Sehr hoch	Audio, Streaming Dienste	Streaming, Tonspuren
JPG	Bilder	quelloffen	Meist komprimiert	Binär aber umgewandelt, Unterdateiformate	Sehr hoch	Fotografie	Bildverarbeitung, Archivierung
pdf	Dokumente	Proprietär, Adobe	komprimiert	Binär, Zertifikat	Sehr hoch	Dokumente	Dokumentenaustausch
XML	Text	ja	nicht	Sprachelemente	Sehr hoch	Datenaustausch	Schnittstellen
TIF	Bilder	ja	Meist unkomprimiert	Binär, teilweise mit Geo-Info	hoch	Druckvorstufe	Bildverarbeitung
MPEG-2	Video mit Audio	patentfrei	komprimiert	Binär, ähnlich wie jpg	Hoch, alte DVD	Video	Streaming, DVB

Datenformate II

Versuch einer Systematik

Datei (statische Daten)

- > Formatiert, strukturiert, Datenbank
 - > Header
 - > Identifier
 - > Synchronisation
 - > Parameter z.B. Samplingfrequenz
 - > Daten



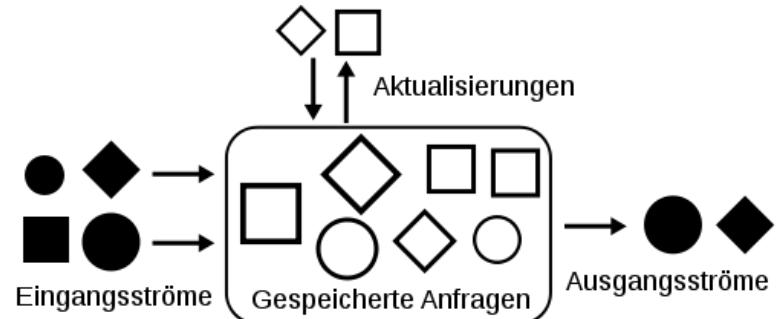
Abfrage per SQL

Beispiel Datenpaket: Frame Header MP3 [Wiki]

Byte 1								Byte 2						Byte 3						Byte 4						
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Sync								ID	Layer	Pr	Bitrate				Freq	Pa	Pv	Kanal	ModEx	Cp	Or	Emph				

Stream (Datenstrom)

- > Kontinuierliche Abfolge von Datensätzen
 - > Geordnet
 - > Nur sequentieller Zugriff
 - > Große Datenmengen in kurzer Zeit möglich
 - > Echtzeitsysteme



Data Stream Management System [Wiki]



Datenformate III

Protokoll/Kommunikation

Protokoll

- > Austausch von Daten
- > Syntax bestimmt Kommunikationsverhalten (Semantik)
- > Datenpaket enthält z.B.
 - > Sender/Empfänger
 - > Paketlängen
 - > Prüfsummen
 - > Daten
- > auch feste Paketsequenzen für den Auf- und Abbau der Verbindung (Overhead)

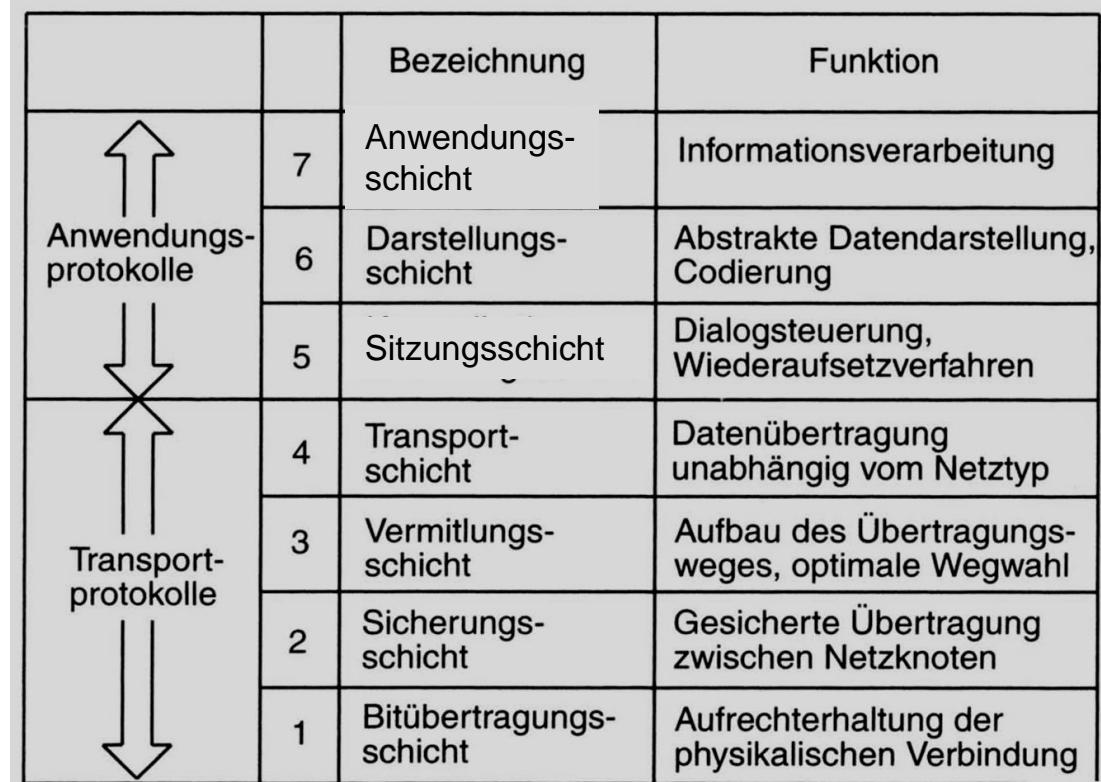
Bekannte Internet-Protokolle:
http, SMTP, ftp usw.

The image shows a screenshot of a web browser window. At the top, there is a green header bar containing the ILIAS logo (a blue 'e' and 'L' inside a red square) followed by the text "ILIAS FH AACHEN - Vorlesung". To the right of this text are three white icons: a magnifying glass, a plus sign, and a minus sign. Below the header is a white address bar. In the address bar, there is a shield icon followed by a lock icon, indicating a secure connection. To the right of these icons is the URL: "https://www.ili.fh-aachen.de/ilias.php?ref_id=469734&cmdClass=ilrepos".

Datenprotokolle

ISO/OSI-7-Schichten-Referenzmodell

- > OSI = Open Systems Interconnection, herstellerunabhängig
- > es müssen nicht alle 7 Schichten genutzt werden
- > DIN EN ISO/IEC 7498: Informationstechnik; Kommunikation offener Systeme; Basis-Referenzmodell; 1995
- > Beispiele:
 1. RS422, RS232
 2. CSMA/CD, RTS/CTS
 3. IP
 4. TCP, UDP
 5. RPC, ISO 9548
 6. ISO 9576 (z.B. ASCII Umwandlung)
 7. Anwendung



ISO/OSI [nach LANGMANN]

Alle deutschen Schüler trinken verschiedene Sorten Bier

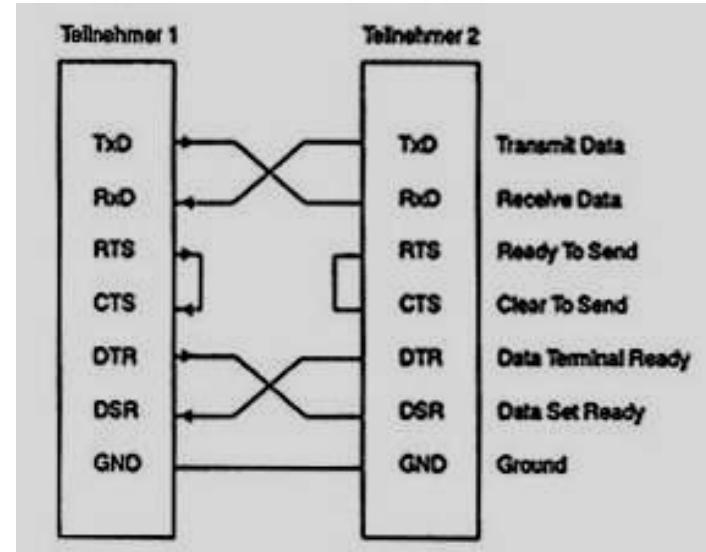
Das ISO/OSI Modell mal anders erklärt



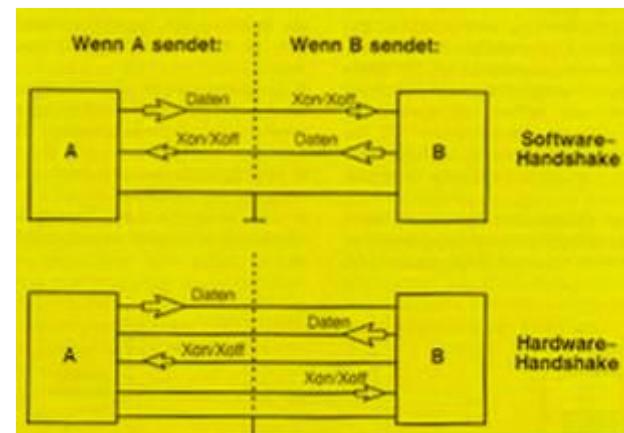
Daten-Schnittstellen

RS 232 – V.24

- > RS = Recommended Standard
- > ist älter als ISO/OSI-Schichtenmodell
- > meist verwendete serielle Schnittstelle
- > Zwei-Punkt-Verbindung (unterste 2 ISO/OSI-Ebenen)
- > seriell
- > Stecker max. 25 pin (oft nur 3 genutzt)
- > logisch 0: +3 V bis +15 V
- > logisch 1: -3 V bis -15 V
- > 19200 Bit/s = 19200 Baud bis ca. 20 m (bei S und E gleich einstellen)
- > vgl. 20 mA-Schnittstelle



T-, R-Data-Adern überkreuz [SCHNELL]



Soft-, Hardware-Handshake [Kief, H. B.: s. o.]

Daten-Schnittstellen

Bekannte Formate im ISO/OSI-7

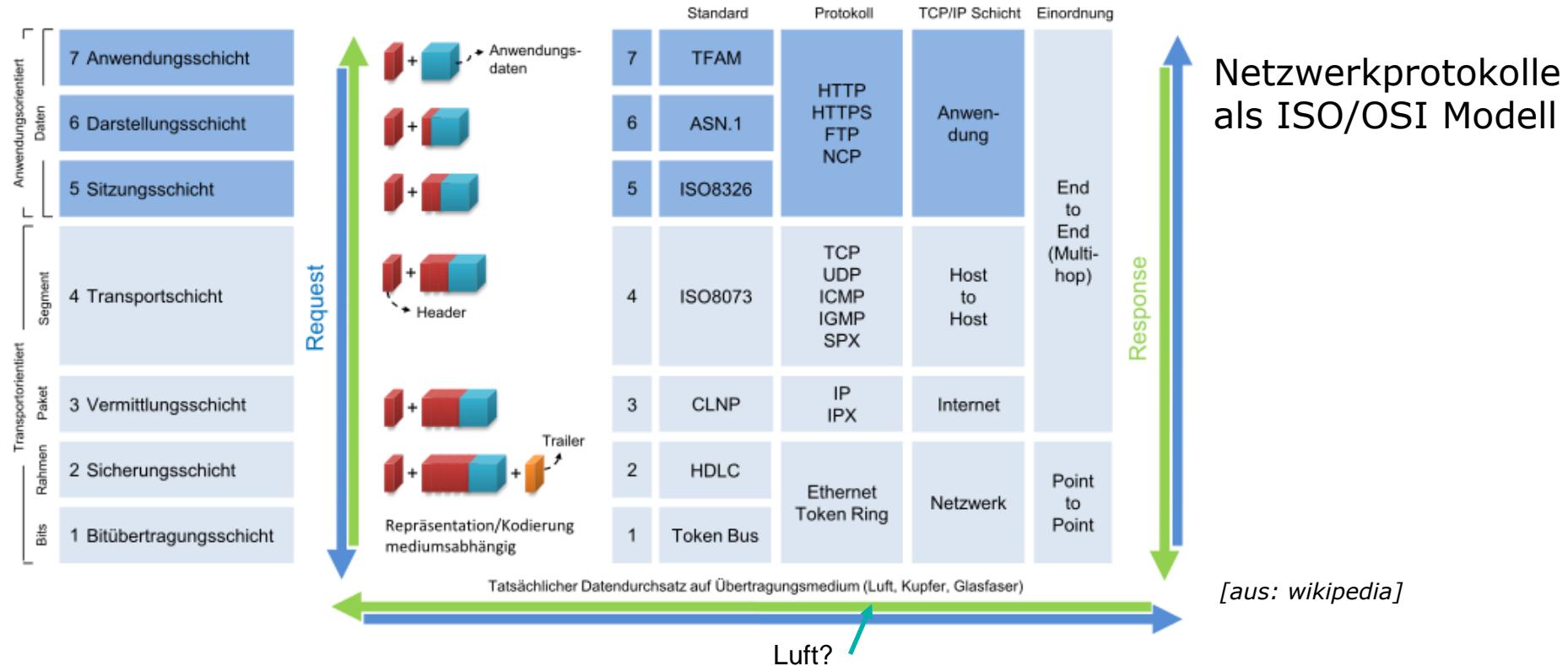
Kommunikation im OSI-7-Schicht-Modell (Open Systems Interconnection Reference Model)

PC 1 in Netzwerk A (z.B. Client)

PC 2 in Netzwerk B (z.B. Server)



Ablauf: PC 1 sendet eine Anfrage (Request) an PC 2, indem diese zunächst vor der eigentlichen Übertragung durch Hinzufügen der Schichtenheader-/trailer formatiert wird. PC 2 empfängt den Request von PC 1 und nimmt die Schichtenheader-/trailer wieder aus der Nachricht, bis nur noch die Anwendungsdaten (innerste Bits) vorhanden sind und verarbeitet diese in der Endanwendung. Die Antwort (Response) läuft analog zur Übertragung der Anfrage, bloß in umgekehrter Richtung ab.

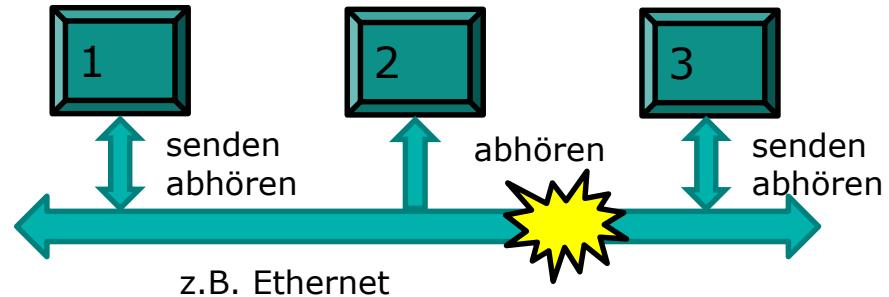


Daten-Schnittstellen

Zugriffs-Kontrollmechanismen

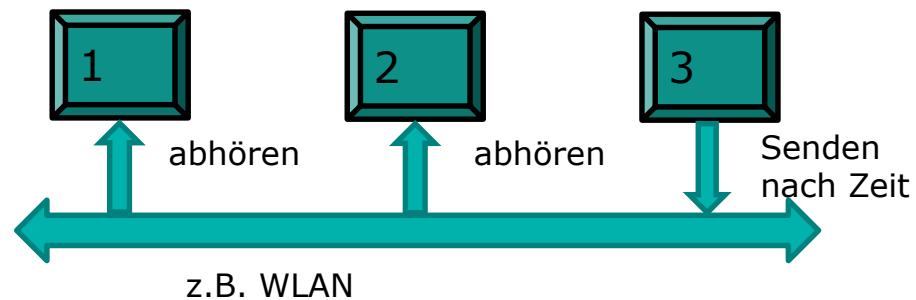
CSMA/CD (carrier sensitive, multiple access/collision detection)

- > alle Busteilnehmer gleichberechtigt
- > keine Adressen
- > Sendung bedarfsweise
- > bei Kollision erneuter Sendeversuch nach Zufallsprinzip
- > bedingt echtzeitfähig



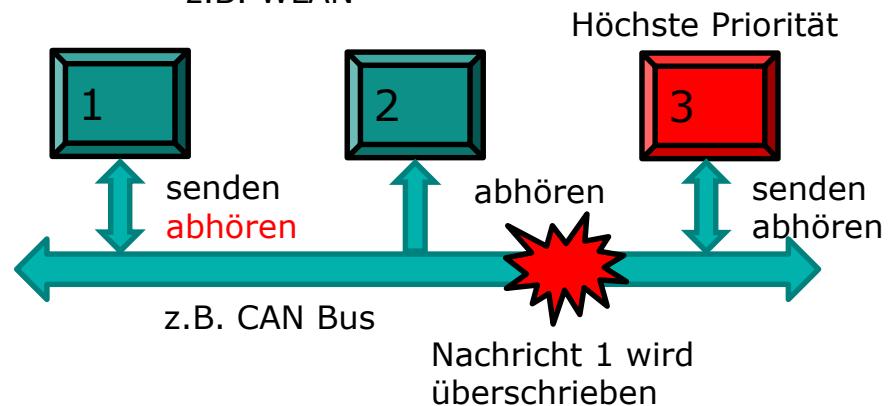
CSMA/CA (carrier sensitive, multiple access/collision avoidance)

- > Nicht notwendigerweise Voll-Duplex
- > Verwendung in Funknetzwerken
- > „Listen before talk“



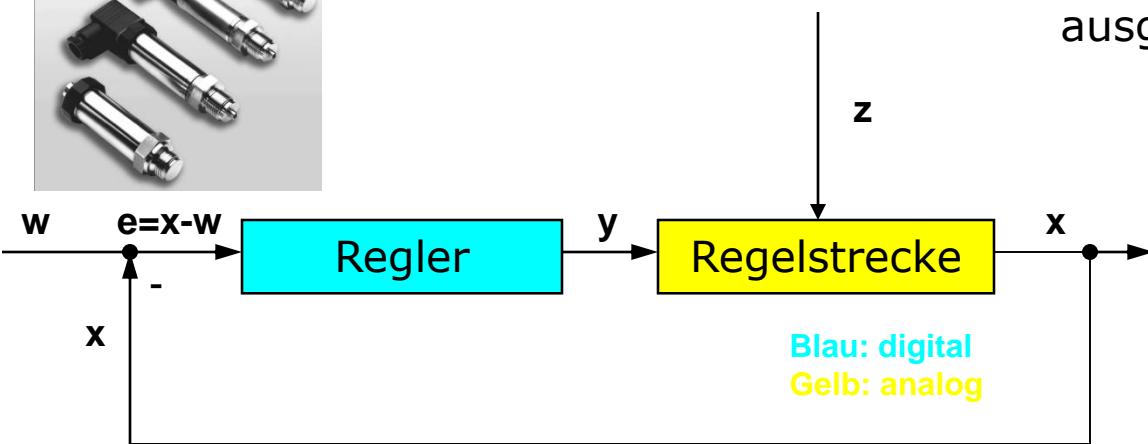
CSMA/CR (carrier sensitive, multiple access/collision resolution)

- > Erkennt Kollisionen nach CSMA/CD
- > Durch Bitarbitrierung wird die Nachricht mit der höchsten Priorität nahezu verzögerungsfrei gesendet



Feldebene Komponenten

x: **Istwert** kommt vom **Sensor** z.B. Temperatur, Position oder Druck.



Regler: Zumeist digital als **PID-** oder **Mehrpunktregler** ausgeführt.



Regler, Regelstrecke und Komponenten

y: Stellwert wird zum **Stellglied** übermittelt z.B. Wärmezufuhr oder Motorstrom, Ventilstellung



Feldebene

Sensorik: Temperatur, Druck, Position

Erfassung des Istwerts



Temperaturmessung durch Thermoelemente oder Widerstandsänderung von Platin (Pt100, -200 bis 850°C).
Ausgang: 4-20mA, 0-10V

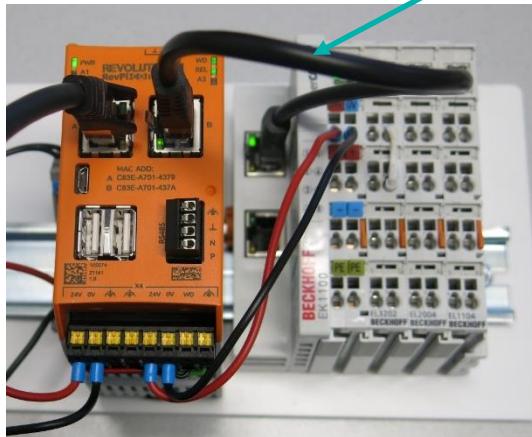
Druckmessung durch Verformung eines Dehnungsmess-Streifens auf einer Membran.
Ausgang: 4-20mA, 0-10V, CANOpen

Positionsbestimmung durch drei digitale Ausgänge
Ausgang: 0 und 5V (TTL)
A,B,I
90° versetzt (Quadratur)

Feldebene

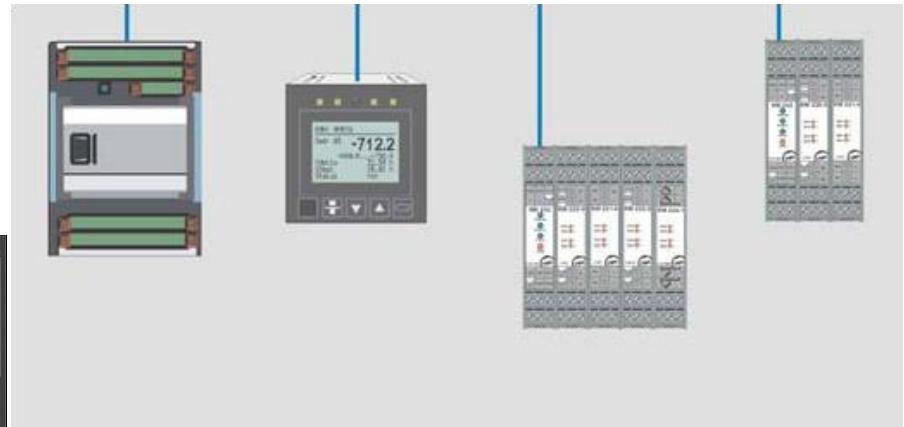
Regler: PID, Mehrpunktregler, SPS

SPS mit PID Regler und EtherCAT Anbindung



Festverdrahtete, digitale PID Regler mit einstellbaren Parametersätzen
 Eingänge: 4-20mA, 0-10V, Frequenz
 Ausgänge: 4-20mA

Erzeugung des Stellwerts



„Multifunktionseinheiten“

Softwarekonfigurierbare PID Regler, Mehrpunktregler inkl. Mathematikbibliotheken, SPS usw.
 Eingänge: 4-20mA, 0-10V, Frequenz, CAN Bus
 Ausgänge: 4-20mA, Bussysteme

Feldebene

Stellglied: Pneumatik, Motoren

Ausgabe des Stellwertes



Pneumatisches Stellglied
zur Druckregelung.
3/2, 5/2, 5/3 Wegeventil
Eingang: 4-20mA, 0-10V



Motorisches Stellglied
zur
Volumenstromregelung
(Drosselung).
Eingang: 4-20mA, 0-
10V



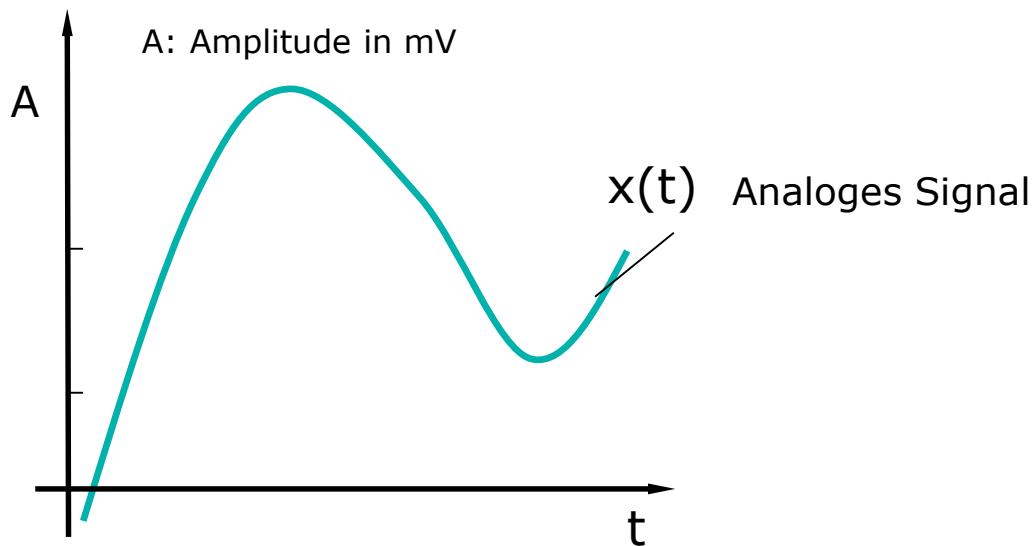
Pneumatisches
Stellglied zur
Volumenstromregelung
(Drosselung).
Eingang: 4-20mA, 0-10V

Digitale Signalverarbeitung

Signale und Digitalisierung

Signale

Ein Signal ist ein zeitabhängiges Zeichen mit dem Informationen übertragen werden können. Meist sind Signale physikalischer Natur, wie Spannung, akustische Wellen usw. Signale sind analog oder digital. Das(selbe?) Signal kann eine analoge (Vinyl-Schallplatte) oder digitale Repräsentation (WAV-Datei) haben. Signale können auch komplex sein.

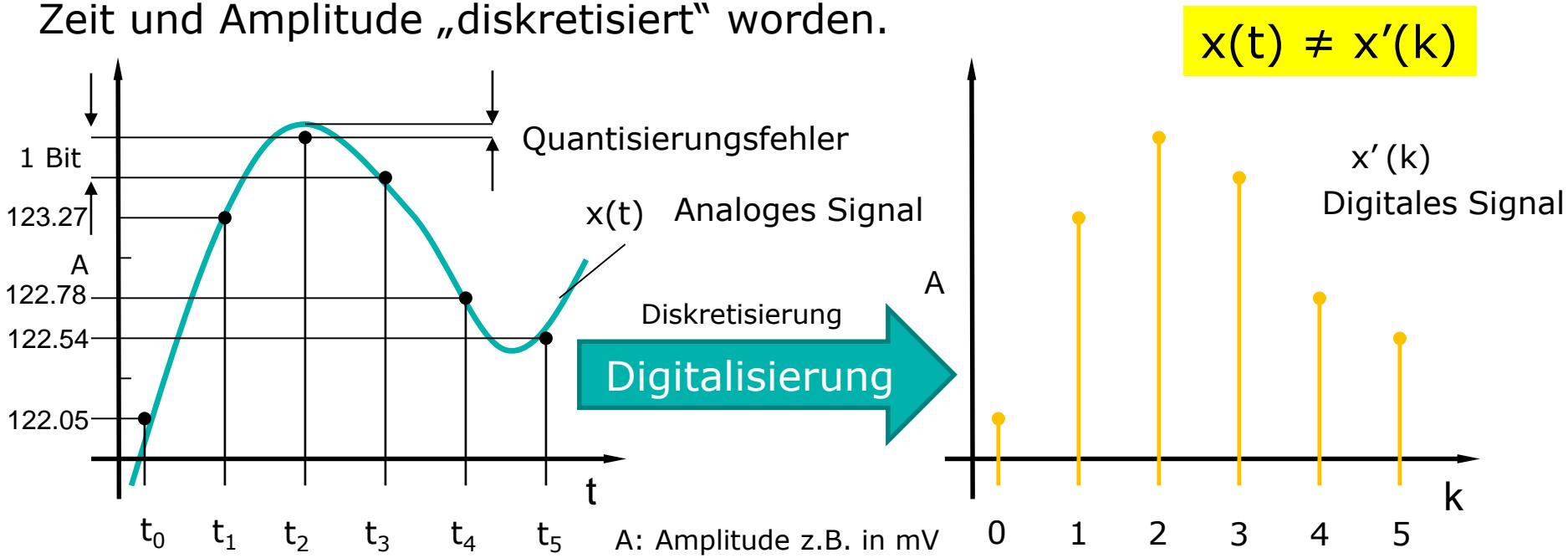


Digitale Signalverarbeitung

Signale und Digitalisierung

Digitalisierung I

Bei der Digitalisierung wird das analoge Signal $x(t)$ abgetastet und in eine andere Form $x'(k)$ überführt. Bei konstanter Abtastzeit entsteht eine Liste von äquidistanten Werten, die das ursprüngliche Signal in einer anderen Form repräsentieren. Da z.B. der Analog/Digitalwandler und die Zeitbasis nur eine begrenzte Auflösung besitzt, ist diese Form in Zeit und Amplitude „diskretisiert“ worden.

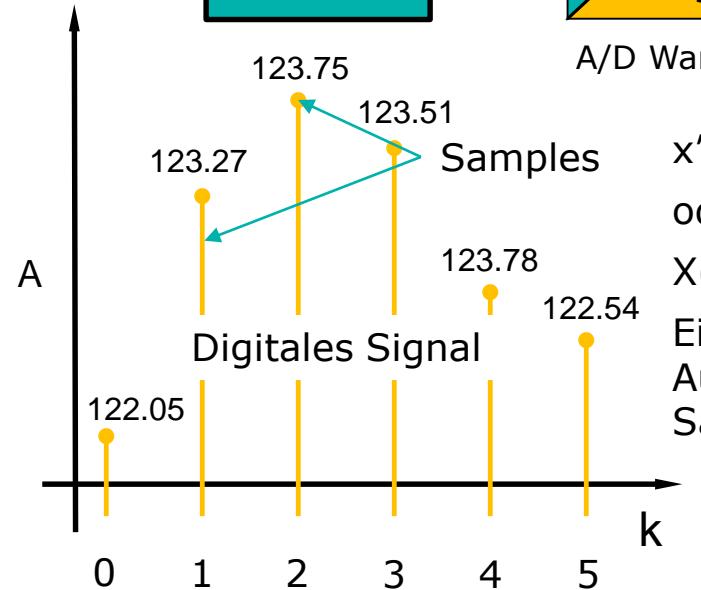
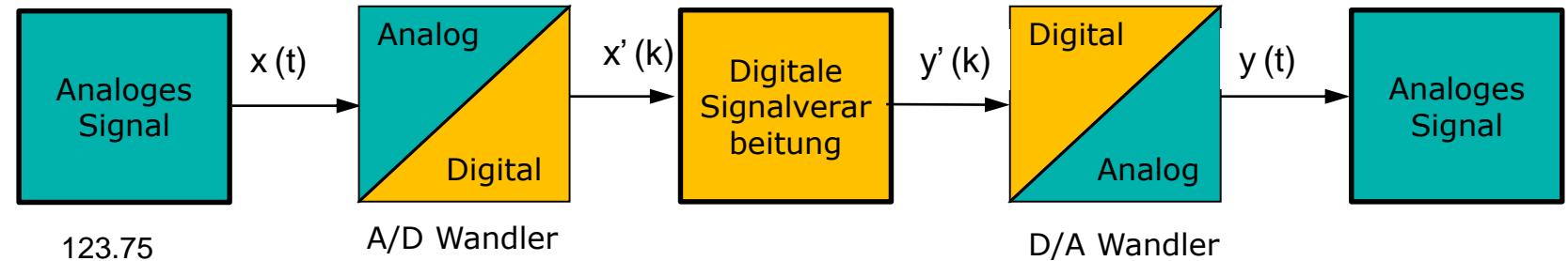


Digitale Signalverarbeitung

Signale und Digitalisierung

Digitale Signalverarbeitung (Digital Signal Processing DSP)

Digitale Signale haben viele Vorteile z.B. lassen sie sich in Computern verarbeiten, archivieren, verändern, erzeugen und relativ einfach übertragen. Die mathematischen Algorithmen zur Veränderung des Signals finden im digitalen Teil statt.



$x'(k) = [122.05, 123.27, 123.75, 123.51, 123.78, 122.54]$
oder bei bekannter Auflösung von 0.2442mV des A/D Wandlers
 $X(k) = [500, 505, 507, 506, 503, 502]$
Eine Reduzierung der Lautstärke, falls das Signal eine Audioquelle ist, kann z.B. durch Subtraktion oder Division der Samples erreicht werden.

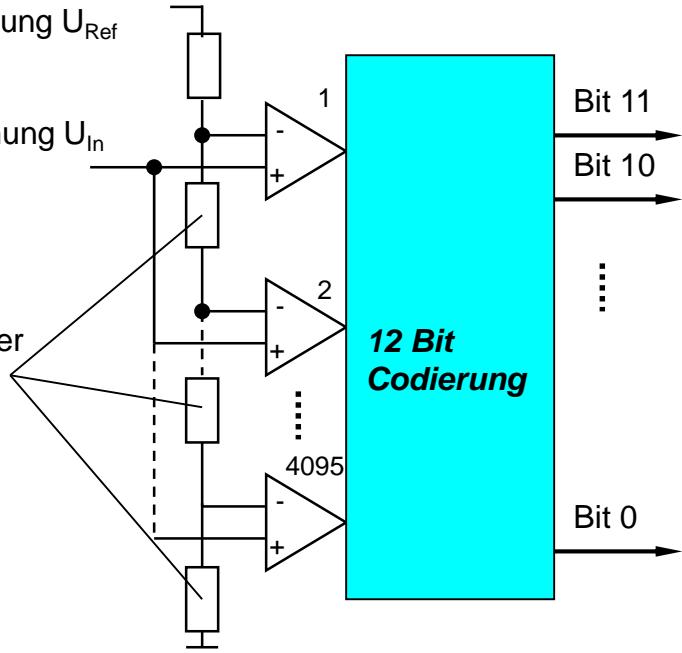
Digitalisierung

A/D Wandler

Referenzspannung U_{Ref}

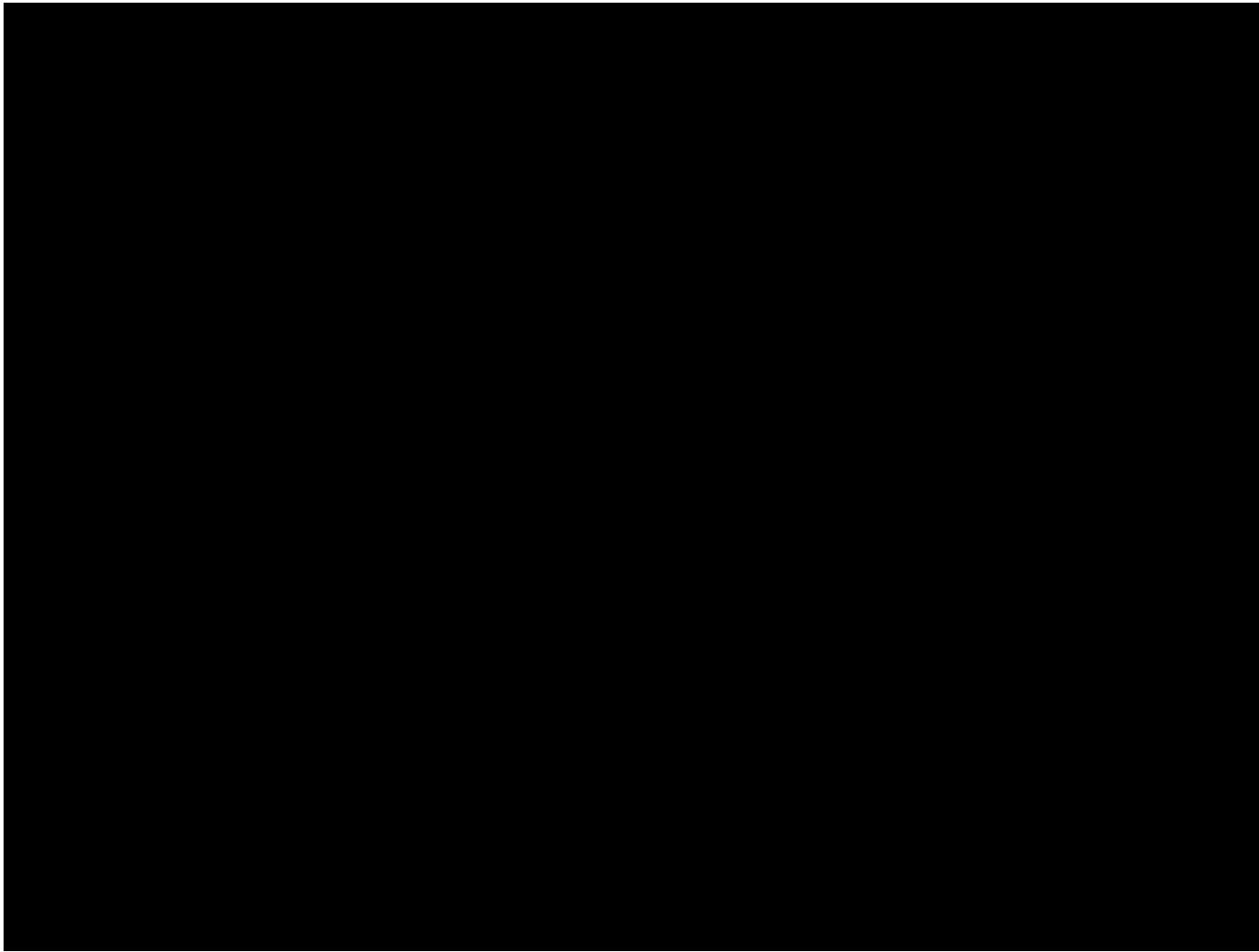
Eingangsspannung U_{In}

Spannungsteiler



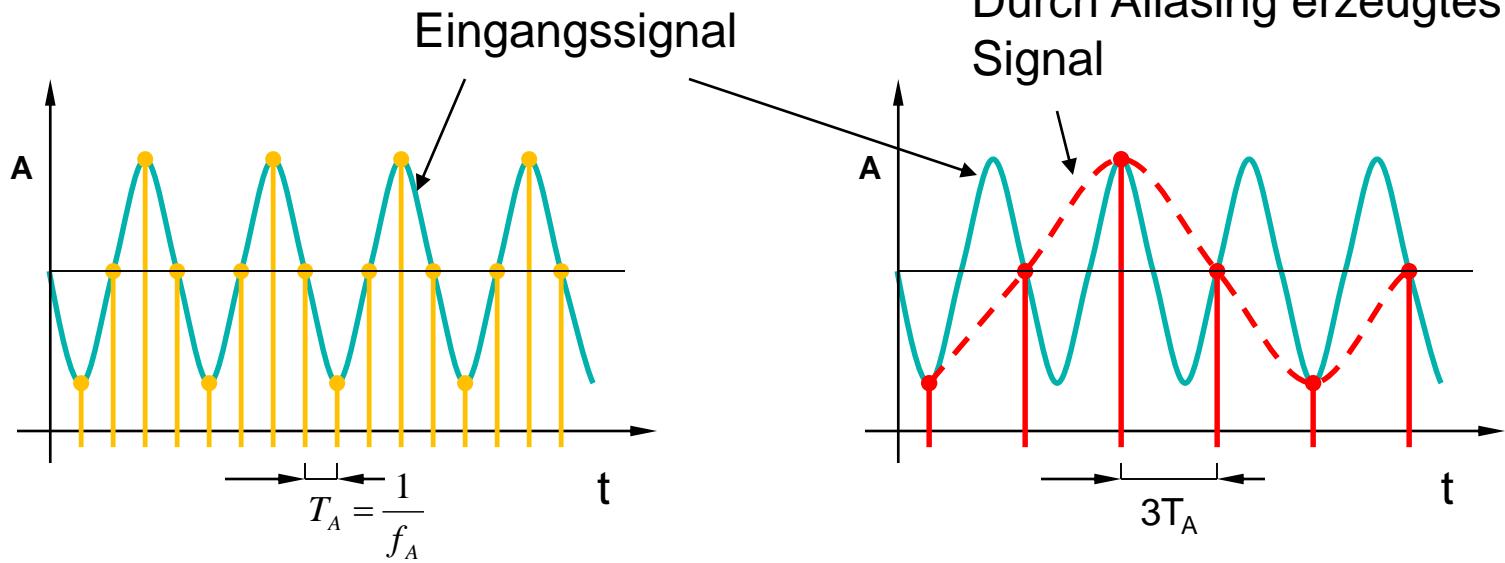
Ein A/D Wandler erzeugt eine zur Eingangsspannung proportionale Zahl, die danach digital weiterverarbeitet werden kann. Man unterscheidet zwischen drei Verfahren: Parallel-, Wäge- und Zählverfahren, wobei für schnelle A/D Wandler das Parallelverfahren (Flashverfahren) eingesetzt wird. Hierbei wird gleichzeitig die Eingangsspannung mit n Referenzspannungen verglichen. Man stellt fest, zwischen welchen Spannungen die Eingangsspannung liegt und erhält die gewünschte Ausgangszahl. Die Auflösung des A/D Wandlers n stellt die Anzahl der Bits der Ausgangszahl dar, z.B. 12 Bit bedeutet, dass der Eingangsbereich des A/D Wandlers (z.B. 1 Volt) in 12Bit (4096) Schritte aufgelöst wird. In diesem Fall wäre die kleinstmögliche Auflösung 0,2441 mV. Das entstehende Signal ist jetzt ebenfalls in der Amplitude diskretisiert (quantisiert). Das Signal ist digitalisiert!

Andere A/D Verfahren



Digitale Signalverarbeitung

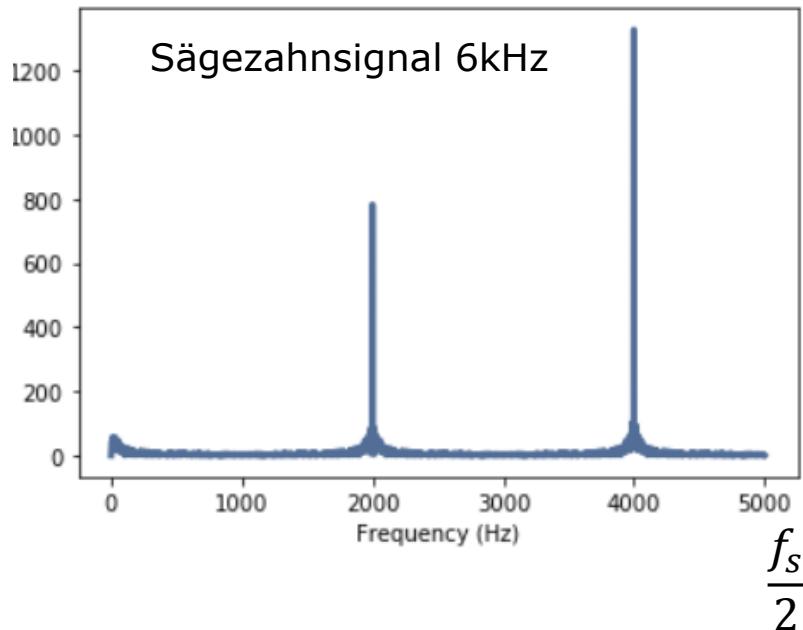
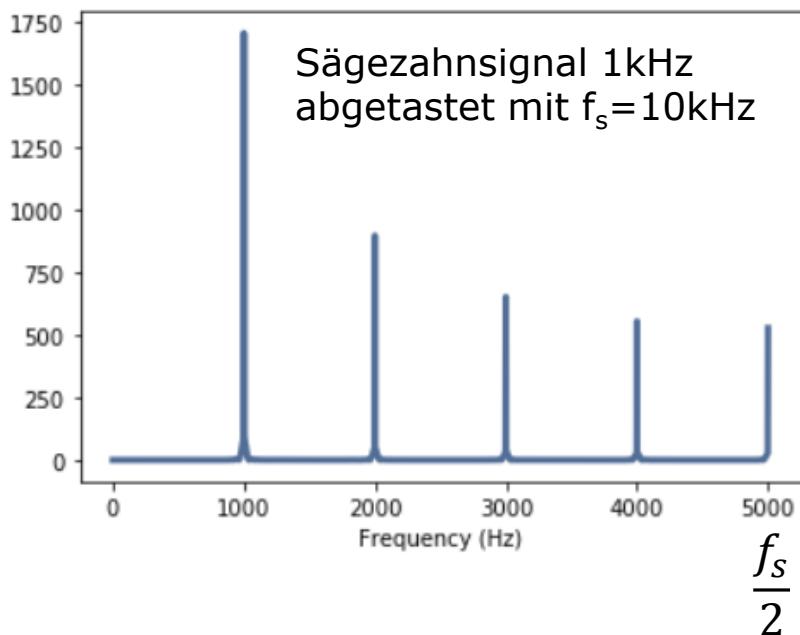
Aliasing im Zeitbereich



Beim Prozess des Abtastens von Signalen kann es zum Aliasing kommen, wenn die Abtastrate kleiner als das Doppelte der höchsten Frequenz ist. Das Eingangssignal kann dann nicht mehr vollständig aus dem abgetasteten Signal rekonstruiert werden. Dieser Zusammenhang wird auch als Shannon oder Nyquist Kriterium bezeichnet.

Digitale Signalverarbeitung

Aliasing im Frequenzbereich



Frequenzen, die oberhalb der halben Abtastfrequenz liegen werden in den unteren Frequenzbereich gespiegelt. Das ursprüngliche Signal ist verschwunden und nur eine Aliasingfrequenz von 2000Hz und 4000Hz ist durch Spiegelung an $f_s/2$ entstanden. Um Aliasing zu vermeiden muss ein analoger (!) Tiefpassfilter vor der Abtastung benutzt werden, der eine Cutoff-Frequenz von deutlich unter $f_s/2$ besitzt. Es sollte dann z.B. mit dem Zehnfachen der höchsten zu erwartenden Frequenz abgetastet werden.

Datenmanagement

Datenerfassung im Prozess I

- > Datenerfassung auf digitalen Systemen ist zeitdiskret
- > Der Zyklus der Erfassung richtet sich nach der Dynamik der Messgröße
 - Abtastzeit/intervall $T_0 \leq T_{\text{dom}}/10$
 - T_{dom} ist die dominierende Zeitkonstante des (Tiefpass-)Systems
 - Temperaturen: ≥ 1 sec; Drücke: $\geq 1 - 10$ msec; Drehzahlen: ≥ 1 msec
- > Die Auflösung/Genauigkeit wird durch die Güte der Messung bestimmt
 - Pt100: $\sim 0,1^\circ\text{C}$ genau => im Messbereich 0 - 100°C reichen 10 Bit A/D-Wandlung
 - Absolutwinkelmessung mit Schlitzscheibe ($0,1^\circ$ genau): 12 Bit A/D-Wandlung nötig
- > Die Zuverlässigkeit/der Informationsgehalt der Messung erhöht sich bei
 - Verwendung von fehlertoleranten Codes (Gray-Code)
 - Verwendung von versetzten Codes (2 Messungen für Drehrichtung)
- > Datenerfassung erfolgt *in situ/online* oder als Analyse/*offline*
 - Analysengrößen (Dichte, Konzentration) oft *offline*
 - Prozessgrößen (Druck, Durchfluss, Temperatur, Drehzahl) *online*

Datenmanagement

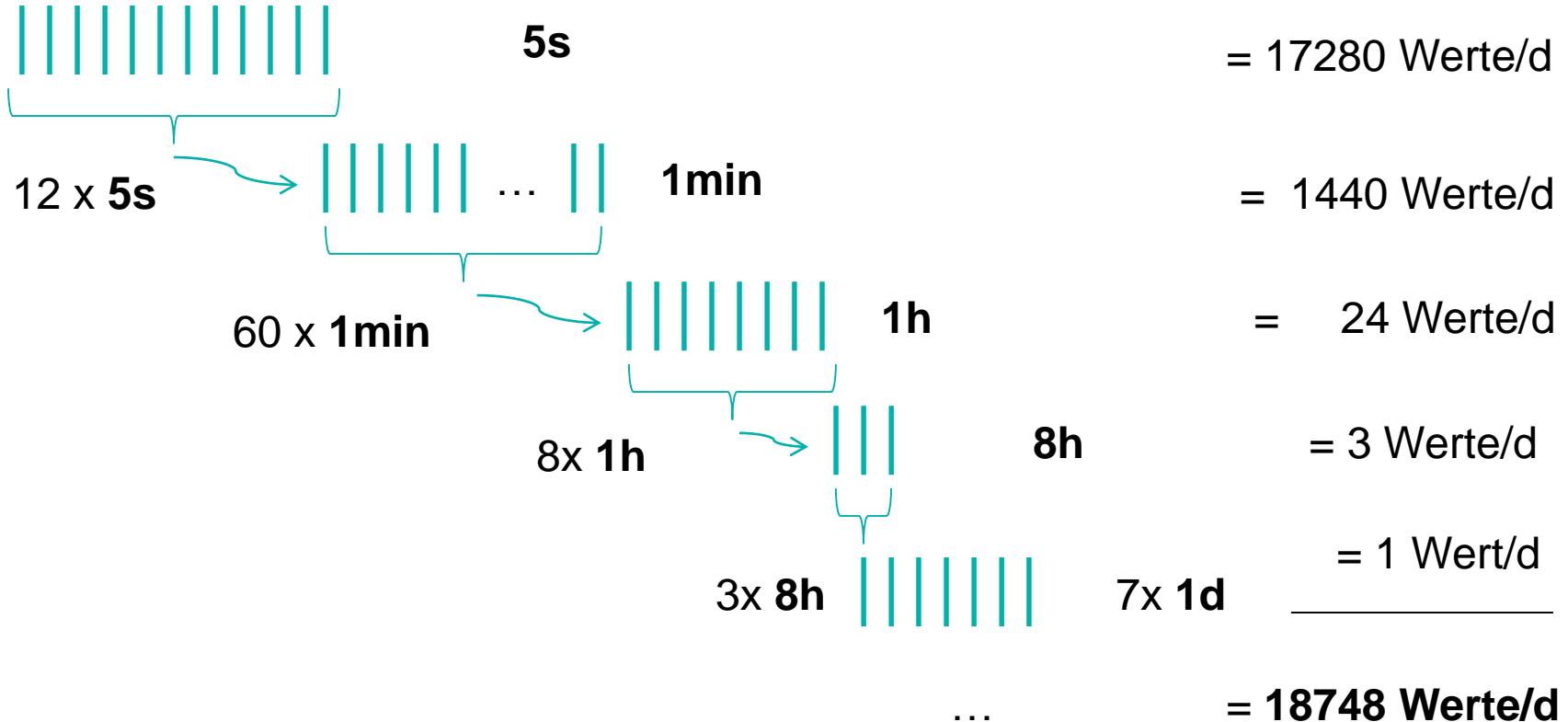
Datenverdichtung

	Konti	Batch
Online-Prozessdaten:	< 1s – 1 min	< 1s – 1 min
Trend:	5 min – 1 h	Einzelne Batch-Phasen
Kurzzeit-Historie:	1 h – 8 h (Schicht)	Gesamt-Batch
Langzeit-Historie:	8 h (Schicht) – 1 d - Gesamt-Lot	Zeitlich/räumlich Zusammengefasste Batches

Datenmanagement

Datenverdichtung

Datenverdichtung im Prozess: 1 Mess-Stelle mit $T_0 = 5\text{s}$

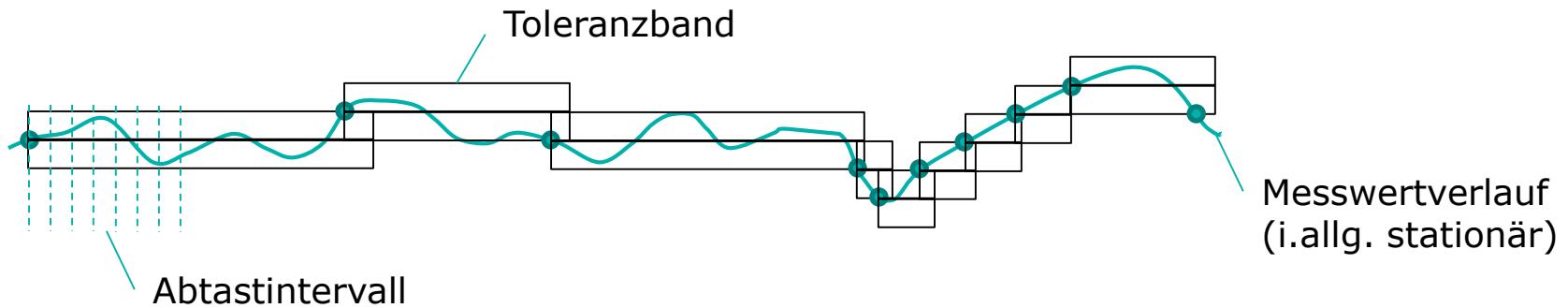


Datenerfassung im Prozess II

1. Exception Reporting (Variante)

Grundgedanke: Daten werden nur erfasst, wenn sich etwas ändert

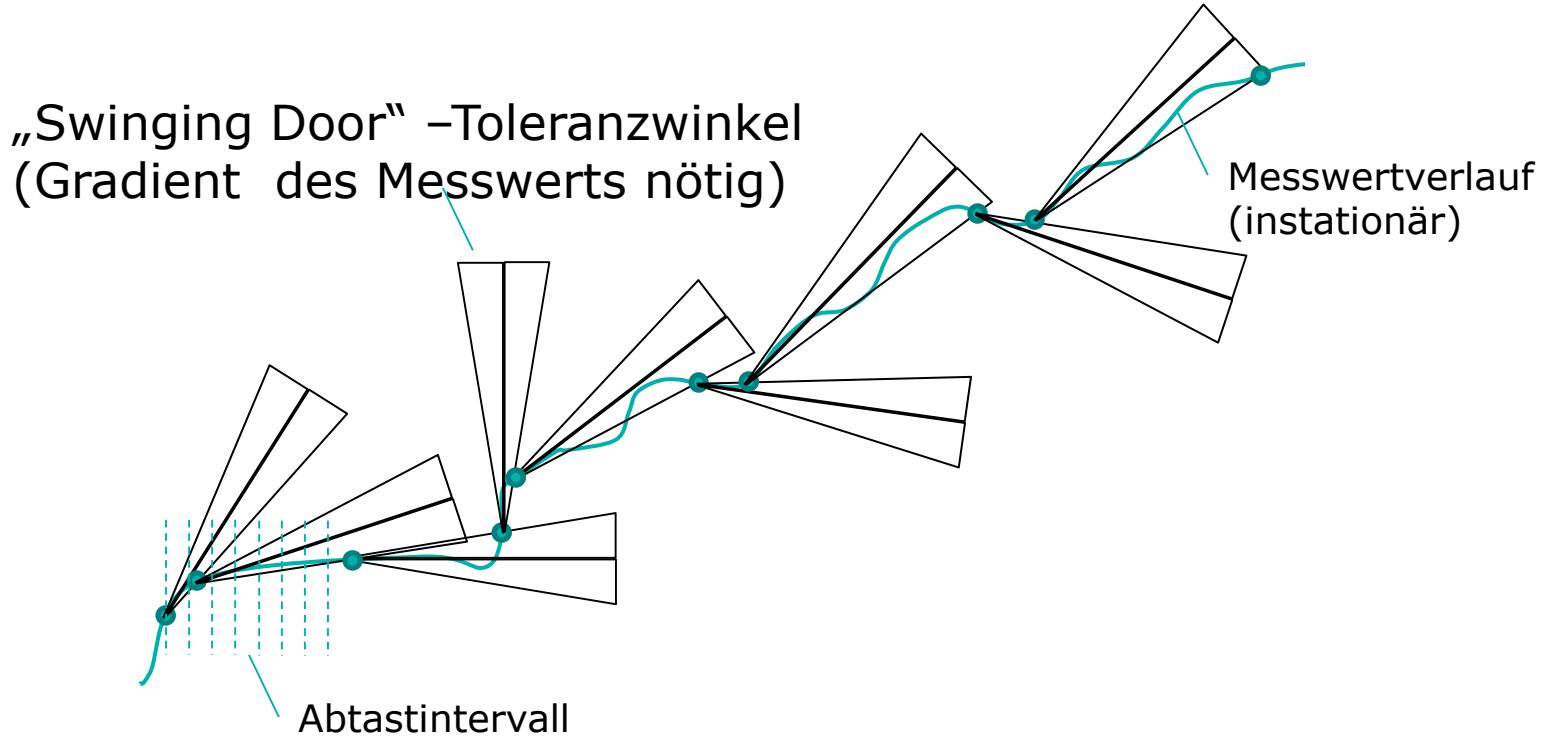
Änderung: Festlegen eines Toleranzbandes



Reduktion des zu archivierenden Datenvolumens durch Übernahme von Werten nur, wenn das Toleranzband verlassen wird.

Datenerfassung im Prozess II

2. Swinging Door Principle (Variante)



Reduktion des zu archivierenden Datenvolumens durch Übernahme von Werten nur, wenn der Winkel der „Swinging Door“ verlassen wird

Datenerfassung im Prozess

Schlussfolgerung

Die Datenreduktion gegenüber der äquidistanten Erfassung und Archivierung ist erheblich.

Der Umfang der Reduktion ist abhängig

- > von der **Toleranzbreite** (Exception Reporting) bzw. vom **Toleranzwinkel** (Swinging Door)
- > von der **Varianz** der Messwerte

Es entsteht **kein Informationsverlust**, da kurzfristige und große Änderungen direkt mit hinreichender Auflösung erkannt werden, während in stationären Zuständen kaum Daten gesammelt werden.

Datenerfassung im Prozess Beispiel

Beispiel: (einfacher) Prozess mit je 10 Mess-Stellen mit $T_0 = 0,2\text{s}$, $1,0\text{s}$ und $5,0\text{s}$

- Verdichtung der $0,2\text{s}$ - und $1,0\text{s}$ -Werte direkt auf 5s -Werte

- Messwerte/d:

$$10 \times (25 + 5 + 1) \times 18748 \text{ Werte/d} = \mathbf{5.811.880 \text{ Werte/d}}$$

=> Beherrschung der Datenflut durch geeignete Wahl der Archivierungszeiträume.

z.B.:

- alle Daten $\leq 5\text{s}$ -Intervall: nur 8h (1 Schicht) archivieren

- Messwerte/d:

$$10 \times (25 + 5 + 1) \times (5760 + 1467) \text{ Werte/d} = \mathbf{2.240.370 \text{ Werte/d}}$$

$$12 \times 60 \times 8 + 1440 + 24 + 3$$

Literatur

Taschenbuch der Automatisierung

Langmann, Hanser Verlag, ISBN 978-3-446-42112-7, 2010

Taschenbuch der Messtechnik

Hoffmann, Hanser Verlag, ISBN 978-3-446-40993-4, 2007

Handbuch der Mess- und Automatisierungstechnik,

Gevatter, Springer Verlag, ISBN 3-540-59135-4, 1999

Handbuch der Prozessautomatisierung, Früh, Maier, Schaudel,

Oldenburg Industrieverlag, ISBN 978-3-8356-3142-7

Taschenbuch für den Maschinenbau, Der „Dubbel“, Springer Verlag,

ISBN 978-3-642-17305-9, Kapitel N, W, X, Y, 2011

LabView for Lego Mindstorms NXT, Gasperi, nts Press, ISBN 978-1-934891-03-2, 2008

Einführung in LabView, Georgi, Hanser Verlag, ISBN 978-3-446-

41560-7, 2009

Datenerfassung im Prozess Beispiel

Statistische Momente:

Mittelwert (MEAN) Summe aller gültigen Werte/Anzahl der gültigen Werte

Maximum (MAX) Maximum aller gültigen Werte

Minimum (MIN) Minimum aller gültigen Werte

Median (MEDIAN) Wert, bei dem es gleichviele gültige Werte gibt, die größer bzw. kleiner sind als der Wert selbst.

Standardabweichung (STABW)

$$\sqrt{\frac{\sum (x - \bar{x})^2}{(n-1)}}$$

Varianz (VARIANCE)

$$\frac{\sum (x - \bar{x})^2}{(n-1)}$$

Generierung von MEDIAN, STABW und VARIANCE aus verdichteten Daten bringt Informationsverlust

Datenauswertung

Beispiel: Pivot-Tabelle

Logbuch Störung Antriebssysteme

Gerät	Kennzeichen	Störung	Datum Beginn	Datum Ende	Dauer
Umrichter	FU_AZ_012	Ausfall	12.11.09 12:45	12.11.09 14:12	0:1.00 1:27
Motor	AS_BV_74	Überhitzung	14.11.09 16:42	14.11.09 17:35	0:1.00 0:53
Motor	AS_BV_74	Lager	16.11.09 20:39	16.11.09 21:32	0:1.00 0:53
Umrichter	FU_AZ_022	Frequenz	19.11.09 0:36	19.11.09 1:56	0:1.00 1:20
Motor	AS_BW_59	Wicklung	21.11.09 4:33	21.11.09 5:10	0:1.00 0:37
Umrichter	FU_BZ_041	Frequenz	23.11.09 8:30	23.11.09 9:37	0:1.00 1:07
Motor	AS_BW_59	Lager	25.11.09 12:27	25.11.09 16:37	0:1.00 4:10
Umrichter	FU_BZ_041	Ausfall	27.11.09 16:24	27.11.09 17:14	0:1.00 0:50
Umrichter	FU_AZ_022	Frequenz	29.11.09 20:21	29.11.09 20:52	0:1.00 0:31
Umrichter	FU_AZ_012	Offset	2.12.09 0:18	2.12.09 0:45	0:1.00 0:27
Motor	AS_BW_18	Überhitzung	4.12.09 0:10		0:1:50
Motor	AS_BV_74	Lager	6.12.09 0:12		0:1:05
Umrichter	FU_AZ_022	Offset	8.12.09 0:14		

Störungsanalyse					
Summe von Dauer Spalten					
Zeilenbeschreibung					
Umrichter		Lager	Überhitzung	Ausfall	Frequenz
Motor	AS_BW_59	Überhitzung	10.12.09 0:16	02:17:00	02:58:00
Motor	AS_BW_18	Ausfall	12.12.09 0:18	02:17:00	02:17:00
Motor	AS_AV_23	Lager	15.12.09 0:20	01:51:00	01:50:00
Motor	AS_BW_18	Überhitzung	17.12.09 0:22	00:50:00	01:07:00
Umrichter			01:27:00	00:27:00	
Motor		06:45:00	04:05:00	01:13:00	00:37:00
AS_BW_59		04:10:00	00:28:00		00:37:00
AS_BW_18		02:44:00	01:13:00		03:57:00
AS_BV_74		01:58:00	00:53:00		02:51:00
AS_AV_23		00:37:00			00:37:00
Gesamtergebnis		06:45:00	04:05:00	03:30:00	02:58:00
			02:17:00	00:37:00	20:12:00

Datenbanksysteme

Datenbankverwaltungssystem DBMS

Informationsmenge verdoppelt sich alle zwei Jahre. Flut von elektronischer Information.

->Database management systems **DBMS**

Gespeicherte Daten werden als **Datenbasis** bezeichnet und enthalten die miteinander in Beziehung stehenden Informationseinheiten [Kemper/Eickler].

Die Gesamtheit der Programme zum Zugriff, zur Kontrolle und zur Modifikation der Daten wird als **DBMS** bezeichnet.

Datenbanksysteme

Datenbankverwaltungssystem DBMS

Datenmodelle

- > Beschreibung der Datenobjekte und
- > Festlegung der anwendbaren Operatoren

Logische Datenmodelle:

- > Netzwerkmodell,
- > **Relationales Datenmodell**,
- > Objektorientiertes und objekt-relationales Datenmodell,
- > Deduktives Datenmodell und
- > **XML**.

Datendefinitionssprache (Data Definition Language, **DDL**)

Datenmanipulationssprache (Data Manipulation Language, **DML**)

Die **DML** besteht aus:

- > Der Anfragesprache (Query Language) und
- > Der eigentlichen Sprache zur Änderung, Einfügen und Löschen der gespeicherten Daten.

Datenbanksysteme

Relationales einfaches Datenbanksystem

Studenten	
MatrNr	Name
26120	Fichte
25403	Jonas

hören	
MatrNr	VorlNr
25403	5022
26120	5001

Vorlesungen	
VorlNr	Titel
5001	Grundzüge
5022	Glaube und Wissen

Beispiel: Abfragesprache SQL Structured Query Language

select Name
from Studenten, hören, Vorlesungen
where Studenten.MatrNr = hören.MatrNr **and**
hören.VorlNr = Vorlesungen.VorlNr **and**
Vorlesungen.Titel = ,Grundzüge';

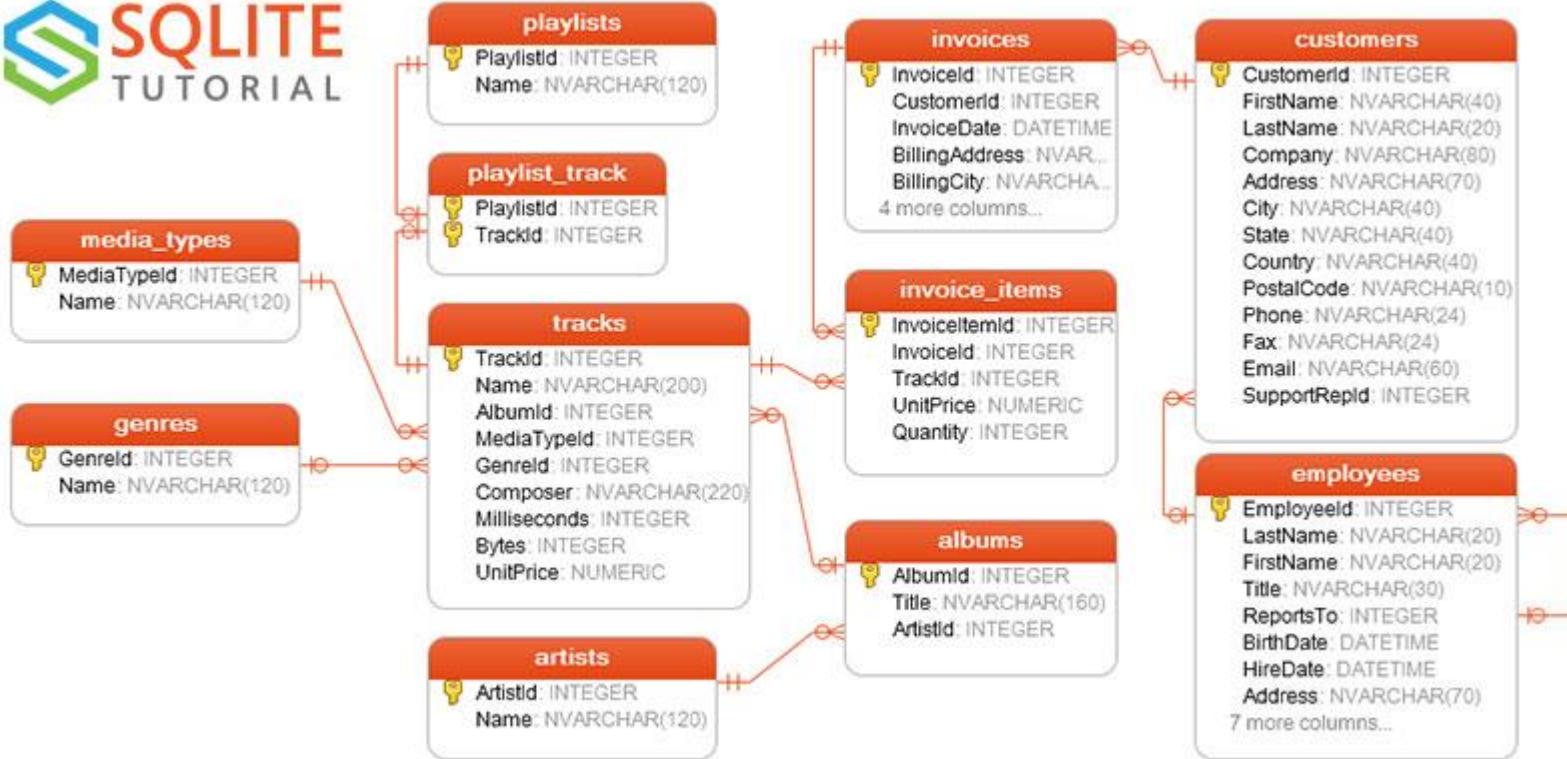
Ermittlung der Namen aller Studenten,
die die Grundzüge VL hören.

update Vorlesungen
set Titel = ,Grundzüge der Logik'
where VorlNr = 5001;

Titel der VL Grundzüge ändern.

Datenbanksysteme

Relationales Datenbanksystem II



Datenbank aus dem SQLite Tutorial.
Damit bitte ein paar Tutorials durchspielen!

Datenbanksysteme

SQLite Visualisierung und Tools

```
SELECT * FROM student limit 100
```

	matrikel_nr	name
1	25403	Jonas
2	26120	Fichte

Enter math.js or SQLite commands

> Plugins für Browser
z.B. Chrome, FireFox



> Plugins für IDE



> Standalone Tools

Datenbanksysteme

SQLite Implementierung



<http://www.sqlitetutorial.net/>



```
import sqlite3
connection = sqlite3.connect("Studenten.db")

#Ein "Cursor" wird erzeugt zum Zugriff auf die Datenbank
cursor = connection.cursor()

#Definition der Felder für den Datensatz
sql_command = """CREATE TABLE student (matrikel_nr INTEGER PRIMARY KEY, name VARCHAR(20));"""

#Befehl für die Datenbank ausführen
sql_command = """CREATE TABLE student (matrikel_nr INTEGER PRIMARY KEY, name VARCHAR(20));"""
#Datensatzinhalt erzeugen
sql_command = """INSERT INTO student (matrikel_nr, name) VALUES (26120, "Fichte");"""

#Befehl für die Datenbank ausführen
cursor.execute(sql_command)

#Noch einen Datensatz erzeugen und Befehl ausführen
sql_command = """INSERT INTO student (matrikel_nr, name) VALUES (25403, "Jonas");"""
cursor.execute(sql_command)

#Die Datenbank aktualisieren
connection.commit()

#Datenbank schliessen
connection.close()
```

Erzeugen einer Datenbank mit
Datensätzen in Python

Datenbanksysteme

SQLite Implementierung DB API 2.0

[sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.9.0 documentation](#)

```
import sqlite3
conn = sqlite3.connect('example.db')
c = conn.cursor()

# Create table
c.execute('''CREATE TABLE stocks
            (date text, trans text, symbol text, qty real, price real)''')

# Insert a row of data
c.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

# Save (commit) the changes
conn.commit()

# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

[SQL Tutorial \(w3schools.com\)](#)

Datenbanksysteme

SQLite Implementierung DB API 2.0

```
import sqlite3
conn = sqlite3.connect('example.db')
c = conn.cursor()

t = ('RHAT',)
c.execute('SELECT * FROM stocks WHERE symbol=?', t)
print(c.fetchone())

# Larger example that inserts many records at a time
purchases = [('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
              ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
              ('2006-04-06', 'SELL', 'IBM', 500, 53.00),
          ]
c.executemany('INSERT INTO stocks VALUES (?,?,?,?,?)', purchases)

conn.commit()

conn.close()
```

ACID und **BASE** Prinzip

Atomicity (Abgeschlossenheit)

Datenbankanweisungen werden erst für gültig erklärt, wenn sie komplett abgeschlossen sind.

Konsistenz (**Consistency**)

Objekte, bzw. Datensätze müssen Integritätsbedingungen also Abhängigkeiten zwischen den Daten jederzeit erfüllen.

Isolation (Abgrenzung)

Nebenläufige Transaktionen dürfen sich nicht beeinflussen, wird durch Sperrverfahren realisiert.

Dauerhaftigkeit (Durability)

Nach erfolgter Transaktion müssen die Daten dauerhaft gespeichert sein, selbst bei Ausfällen.

BASE: Basically Available, Soft State, Eventual consistency

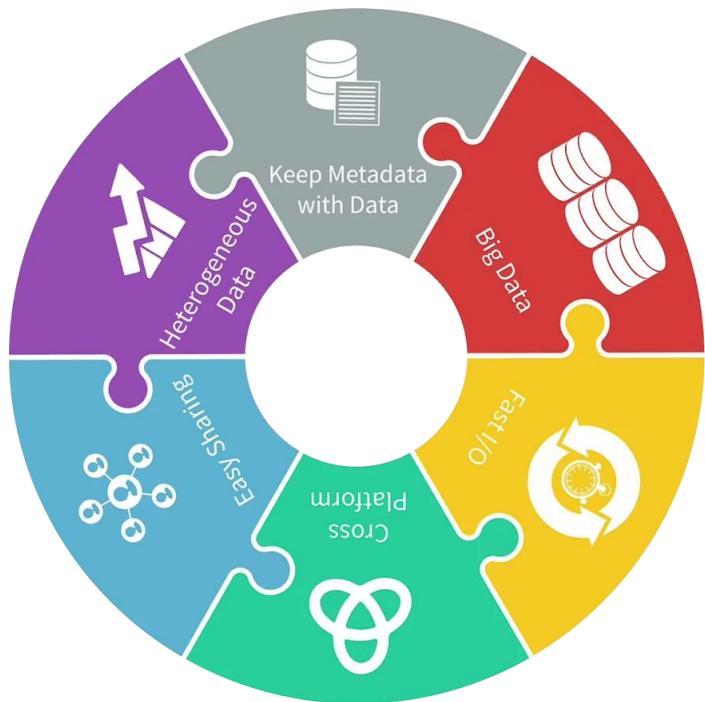
Erst nach kurzer Zeit konsistent

Andere Datenbankformate

Datenbanken und Container für „Big Data“



[<http://nosql-database.org/>]



No SQL

- > Not only SQL
- > Keine Relationen, Open Source
- > Web Scalable, Distributed
- > Einfache API
- > MongoDB, CouchDB

Hierarchische Formate

- > HDF5: Container für verschiedenste Arten von Daten z.B. Tabellen, Einzelwerte, Arrays, Bilder
- > Meist im wissenschaftlichen Umfeld genutzt
- > netCDF: meist genutzt für Arrays

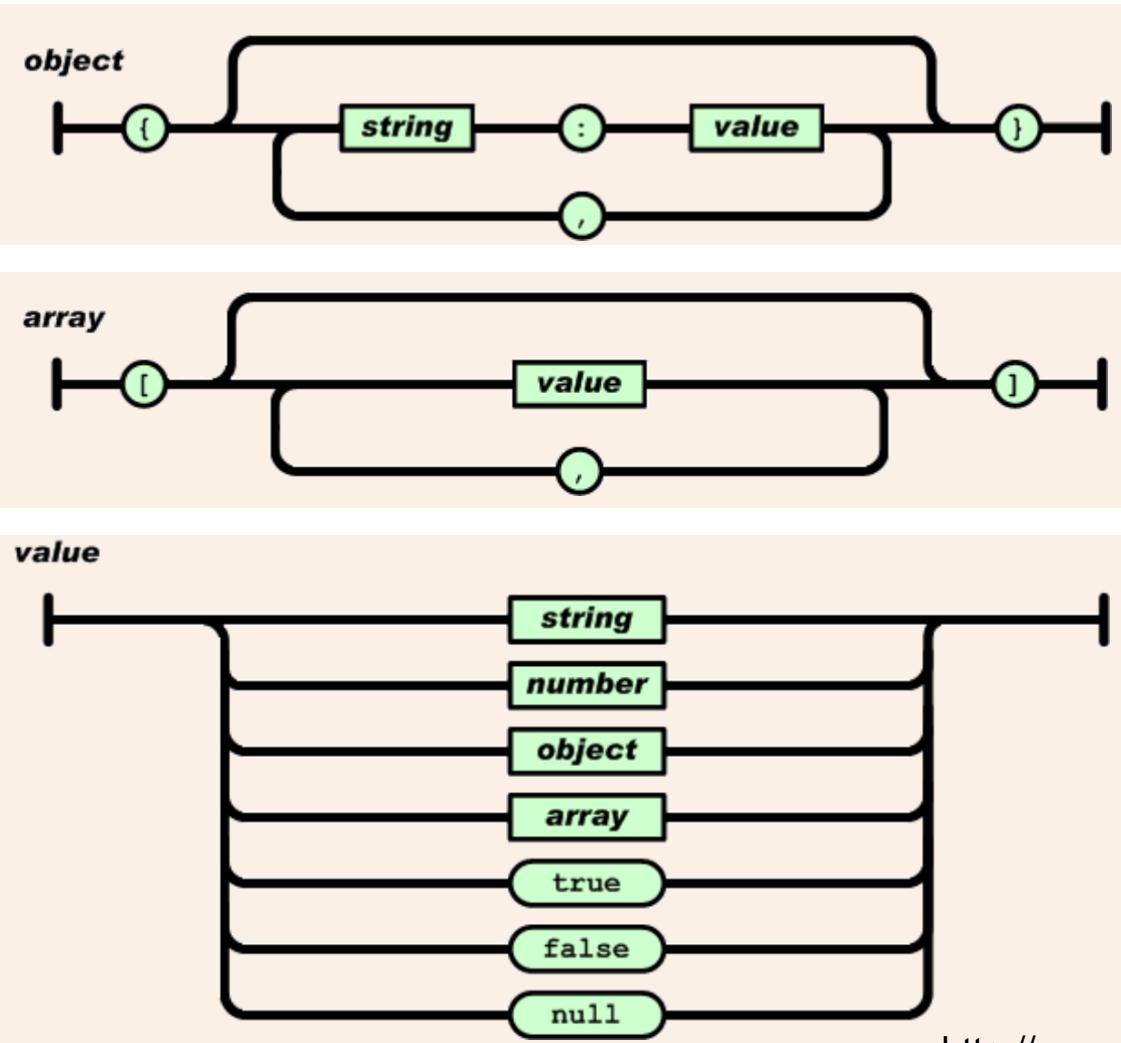


SUPPORT
The HDF Group

[<https://www.hdfgroup.org/>]

JSON (JavaScript Object Notation)

Datenformat zum sprachenunabhängigen Austausch



<http://www.json.org/>

json
element

value
object
array
string
number
"true"
"false"
"null"

object
'{' ws '}'
'{' members '}'

members
member
member ',' members

member
ws string ws ':' element

array
'[' ws ']'
'[' elements ']'

JSON (JavaScript Object Notation)

Python Beispiel

```
import json

# a Python object (dict):
x = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON
string:
print(y)
```

Erzeugen eines einfachen JSON Objektes in Python

```
import json

print(json.dumps({"name": "John", "age": 30}))
```

Ausgabe der einzelnen Datensätze

```
import json

x = {
    "name": "John",
    "age": 30,
    "married": True,
    "divorced": False,
    "children": ("Ann", "Billy"),
    "pets": None,
    "cars": [
        {"model": "BMW 230", "mpg": 27.5},
        {"model": "Ford Edge", "mpg": 24.1}
    ]
}

print(json.dumps(x))
```

Erzeugen eines komplexen JSON Objektes in Python

Andere Datenformate

Datenbanken und Container für „Big data“



<https://youtu.be/dopYLQThdQQ>

NoSQL

NoSQL = “No SQL“



NoSQL = “Not only SQL“

SQL vs. NoSQL

Eigenschaften



SQL:

- > Relationale Struktur
- > Daten werden in „tables“ gespeichert
- > Jeder Eintrag besitzt dieselben Eigenschaften
- > Daten müssen derselben Struktur folgen
- > Erweiterung der Eigenschaften bedingt eine Änderung des kompletten Schemas
- > Das Schema ist strikt
- > ACID Transaktionen
- > Gute „vertikale“ Skalierung

ACID (Atomicity, Consistency, Isolation, Durability)

NoSQL:

- > Keine Relationale Struktur
- > Daten können in verschiedenen Formaten gespeichert werden (JSON, Schlüssel, usw.)
- > Einträge können unterschiedliche Eigenschaften besitzen
- > Neue Eigenschaften können erzeugt werden
- > Das Schema muss nicht aufrecht erhalten werden
- > ACID kann unterstützt sein
- > Häufig BASE Prinzip
- > Konsistenz kann variieren
- > Gute „horizontale“ Skalierung

SQL:

- > Konsistenz
- > Stabilität
- > Ausgereift: jahrelang etabliert und getestet
- > Kompatibel mit allen Betriebssystemen und (Programmier)-Sprachen

NoSQL:

- > Unstrukturierte Daten können gespeichert werden
- > Große Datenmengen speicherbar
- > Flexibel skalierbar
- > Schnell für einfache Abfragen
- > Braucht i.A. keinen Administrator

Beispiele:

MySQL, Oracle, PostgreSQL

Anwendungen:

Kontoführungssystem,
Personalverwaltung,
Adressdatenbank

Beispiele:

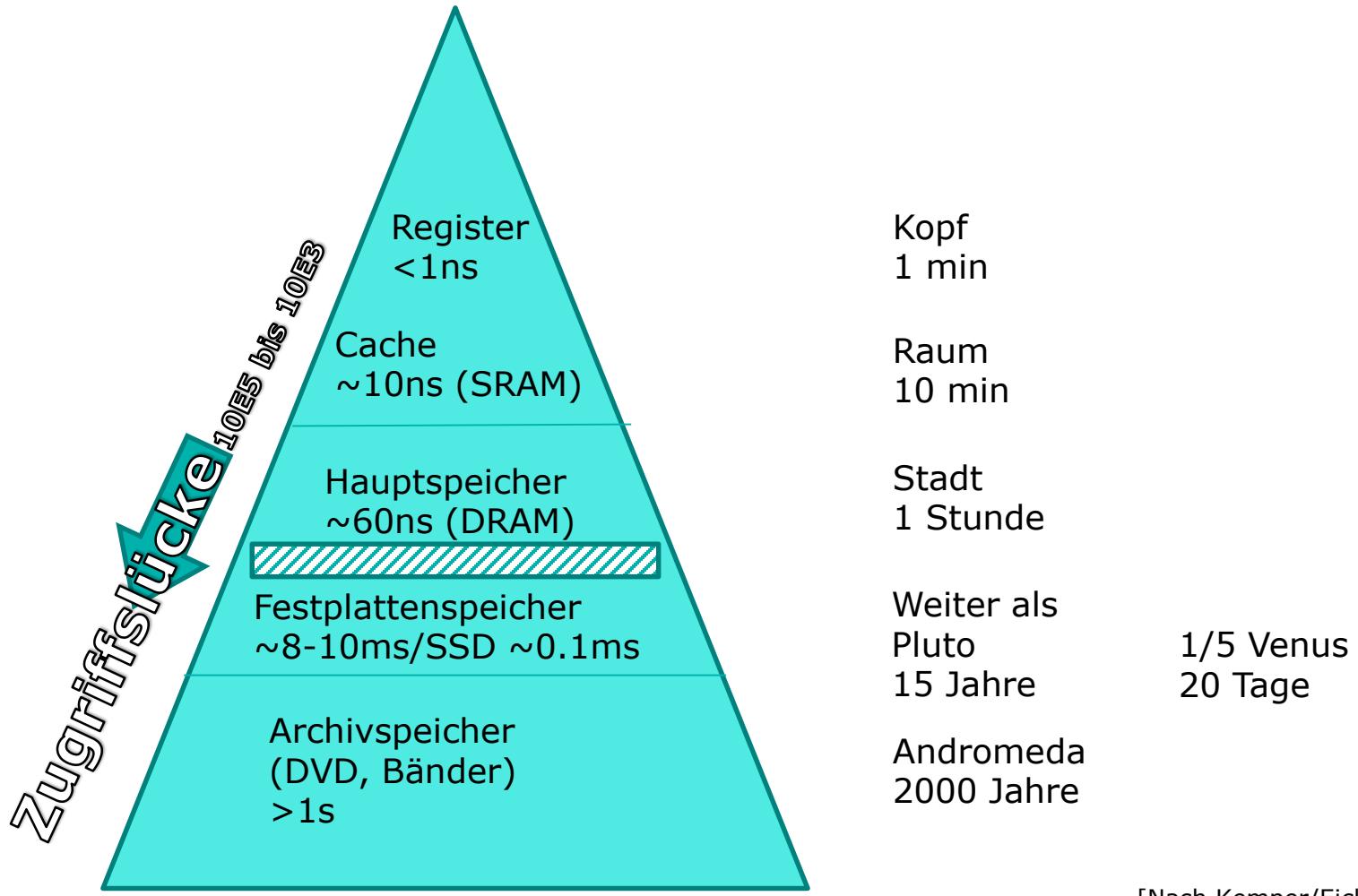
MongoDB, BigTable, Redis,
RavenDB Cassandra, HDF 5

Anwendungen:

Cloud Speicher,
Dynamische Datenbanken z.B.
Prozessdaten,
Echtzeitdaten inkl. Auswertung

Datenbanksysteme

Speicherhierarchie



Datenverkehr

Datenmenge

Maßeinheiten für Datenmengen

Präfix	Bytes	Datenmenge
Byte	1	Ein Buchstabe
Kilobyte (KB)	1.000	Eine Textseite
Megabyte (MB)	1.000.000	Ein kleines Foto
Gigabyte (GB)	1.000.000.000	Ca. 8,5 Minuten HD-Video von einem Camcorder
Terabyte (TB)	1.000.000.000.000	Ca. 250.000 MP3-Dateien
Petabyte (PB)	1.000.000.000.000.000	Die geschätzte Speicherkapazität aller Rechenzentren weltweit 2002
Exabyte (EB)	1.000.000.000.000.000.000	Die fünffache Datenmenge aller jemals gedruckten Bücher
Zettabyte (ZB)	1.000.000.000.000.000.000.000	Die geschätzte Menge aller jemals von Menschen gesprochenen Worte würde digitalisiert 42 Zettabyte entsprechen
Yottabyte (YB)	1.000.000.000.000.000.000.000.000	Unfassbar viel

Alle Angaben sind der Wikipedia und der Studie "How Much Information? 2009" entnommen

ASCII Tabelle – Codierung von Zeichen

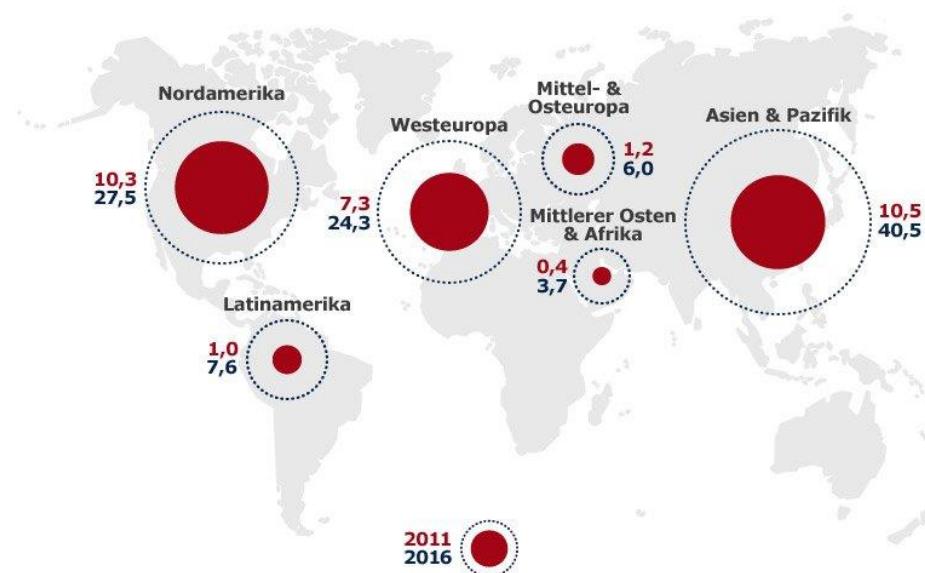
Scan-code	ASCII hex dez	Zeichen	Scan-code	ASCII hex dez	Zch.	Scan-code	ASCII hex dez	Zch.	Scan-code	ASCII hex dez	Zch.	
00	0	NUL ^@	20	32	SP	40	64	@	0D	60	96	
01	1	SOH ^A	02	21	33	!	41	65	A	1E	61	97
02	2	STX ^B	03	22	34	"	42	66	B	30	62	98
03	3	ETX ^C	29	23	35	#	43	67	C	2E	63	99
04	4	EOT ^D	05	24	36	\$	44	68	D	20	64	100
05	5	ENQ ^E	06	25	37	%	45	69	E	12	65	101
06	6	ACK ^F	07	26	38	&	46	70	F	21	66	102
07	7	BEL ^G	0D	27	39	'	47	71	G	22	67	103
OE	8	BS ^H	09	28	40	(48	72	H	23	68	104
0F	9	TAB ^I	0A	29	41)	49	73	I	17	69	105
0A	10	LF ^J	1B	2A	42	*	4A	74	J	24	6A	106
0B	11	VT ^K	1B	2B	43	+	4B	75	K	25	6B	107
0C	12	FF ^L	33	2C	44	,	4C	76	L	26	6C	108
1C	13	CR ^M	35	2D	45	-	4D	77	M	32	6D	109
0E	14	SO ^N	34	2E	46	.	4E	78	N	31	6E	110
0F	15	SI ^O	08	2F	47	/	4F	79	O	18	6F	111
10	16	DLE ^P	0B	30	48	0	50	80	P	19	70	112
11	17	DC1 ^Q	02	31	49	1	51	81	Q	10	71	113
12	18	DC2 ^R	03	32	50	2	52	82	R	13	72	114
13	19	DC3 ^S	04	33	51	3	53	83	S	1F	73	115
14	20	DC4 ^T	05	34	52	4	54	84	T	14	74	116
15	21	NAK ^U	06	35	53	5	55	85	U	16	75	117
16	22	SYN ^V	07	36	54	6	56	86	V	2F	76	118
17	23	ETB ^W	08	37	55	7	57	87	W	11	77	119
18	24	CAN ^X	09	38	56	8	58	88	X	2D	78	120
19	25	EM ^Y	0A	39	57	9	59	89	Y	2C	79	121
1A	26	SUB ^Z	34	3A	58	:	5A	90	Z	15	7A	122
1B	27	Esc ^[33	3B	59	:	5B	91	[7B	123	{
1C	28	FS ^\	2B	3C	60	<	5C	92	\	7C	124	
1D	29	GS ^]	0B	3D	61	=	5D	93]	7D	125	}
1E	30	RS ^^	2B	3E	62	>	5E	94	^	7E	126	~
1F	31	US ^_	0C	3F	63	?	5F	95	_	53	7F	127

Datenverkehr

Weltweiter Datenverkehr: Prognose I

Asiatische Trafficexplosion

IP-Traffic 2011 und 2016* in Exabyte pro Monat

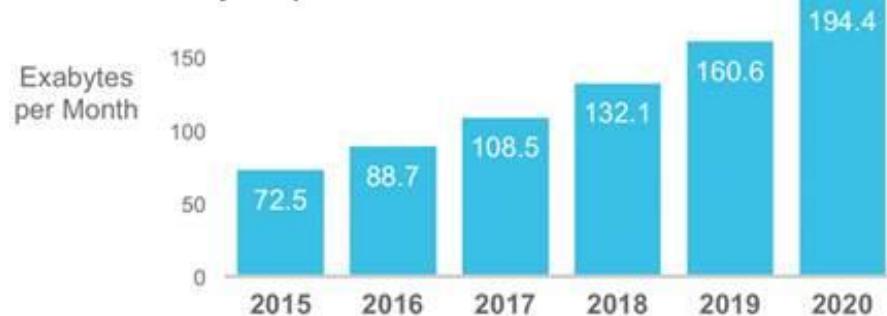


CAGR: Compound Annual Growth Rate,
jährliche Wachstumsrate

22% CAGR
2015–2020

statista SPIEGEL ONLINE

*Prognose Quelle: Cisco



[Source: Cisco VNI, 2016]

Datenverkehr

Weltweiter Datenverkehr: Prognose II

IP Traffic, 2016–2021						
	2016	2017	2018	2019	2020	2021
	CAGR 2016–2021					
By Type (Petabytes [PB] per Month)						
Fixed Internet	65,942	83,371	102,960	127,008	155,121	187,386
Managed IP	22,911	27,140	31,304	35,226	38,908	42,452
Mobile data	7,201	11,183	16,646	24,220	34,382	48,270
By Segment (PB per Month)						
Consumer	78,250	99,777	124,689	154,935	190,474	232,655
Business	17,804	21,917	26,220	31,518	37,937	45,452
By Geography (PB per Month)						
Asia Pacific	33,505	43,169	54,402	68,764	86,068	107,655
North America	33,648	42,267	51,722	62,330	73,741	85,047
Western Europe	14,014	17,396	21,167	25,710	30,971	37,393
Central and Eastern Europe	6,210	7,451	8,940	11,016	13,781	17,059
Middle East and Africa	2,679	3,910	5,538	7,773	10,941	15,490
Latin America	5,999	7,502	9,141	10,861	12,909	15,464
Total (PB per Month)						
Total IP traffic	96,054	121,694	150,910	186,453	228,411	278,108
						24%

“Annual global IP traffic will reach 3.3 ZB (ZB; 1000 Exabytes [EB]) by 2021.”

“Smartphone traffic will exceed PC traffic by 2021.”

“It would take an individual more than 5 million years to watch the amount of video that will cross global IP networks each month in 2021.”

[Source: Cisco VNI, 2017]

What is Industry 4.0

Is there a definition?

„This vertical integration of all processes – from the processing of the order, to the resource management and the manufacturing, up to the delivery of the goods is a revolution of the industrial production: The process levels of the enterprise are continuously linked and can be updated and adjusted with the newest actual individual process data.“ [Future Vision Industry 4.0 BMBF]

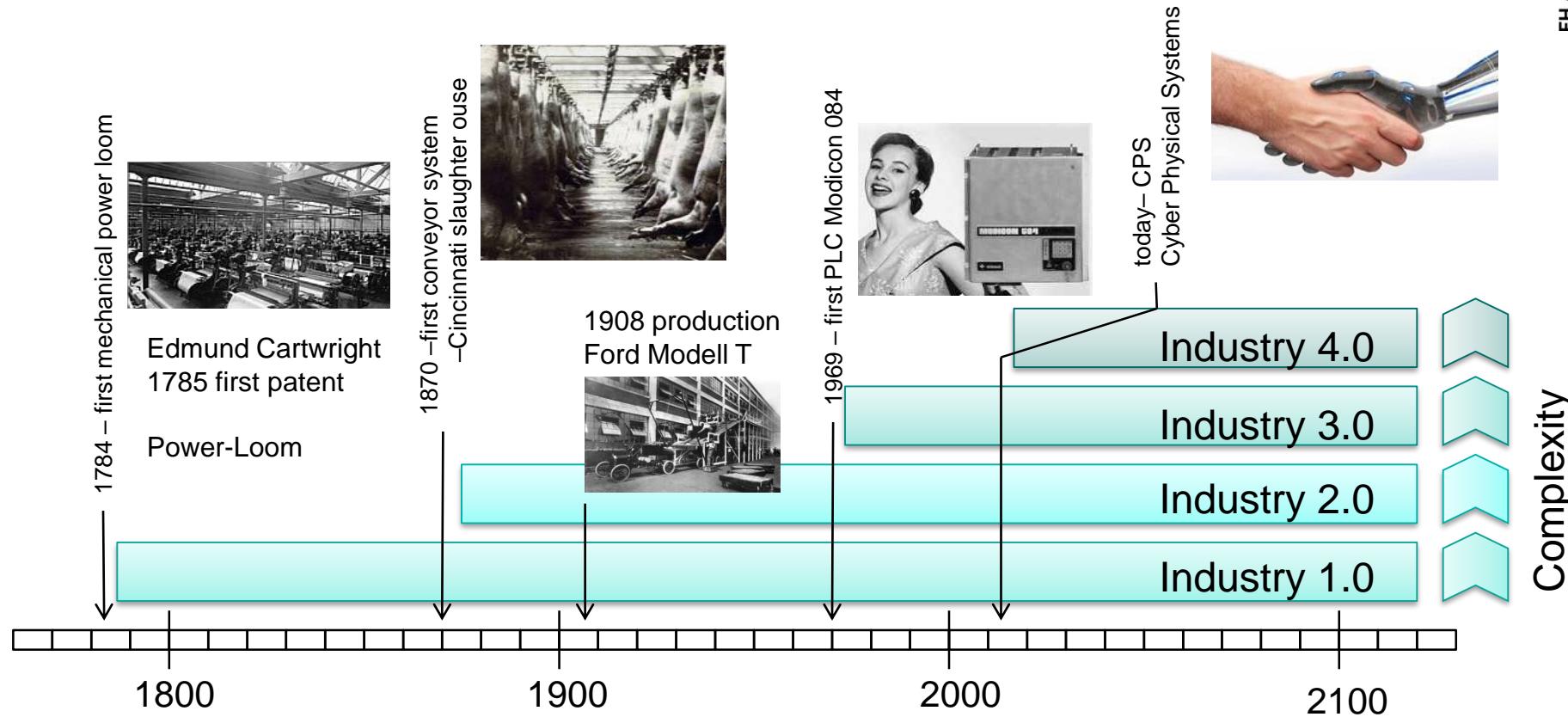


Isn't this the definition of an ERP system?

Industry 4.0 was even mentioned in the last political contract from 2013.

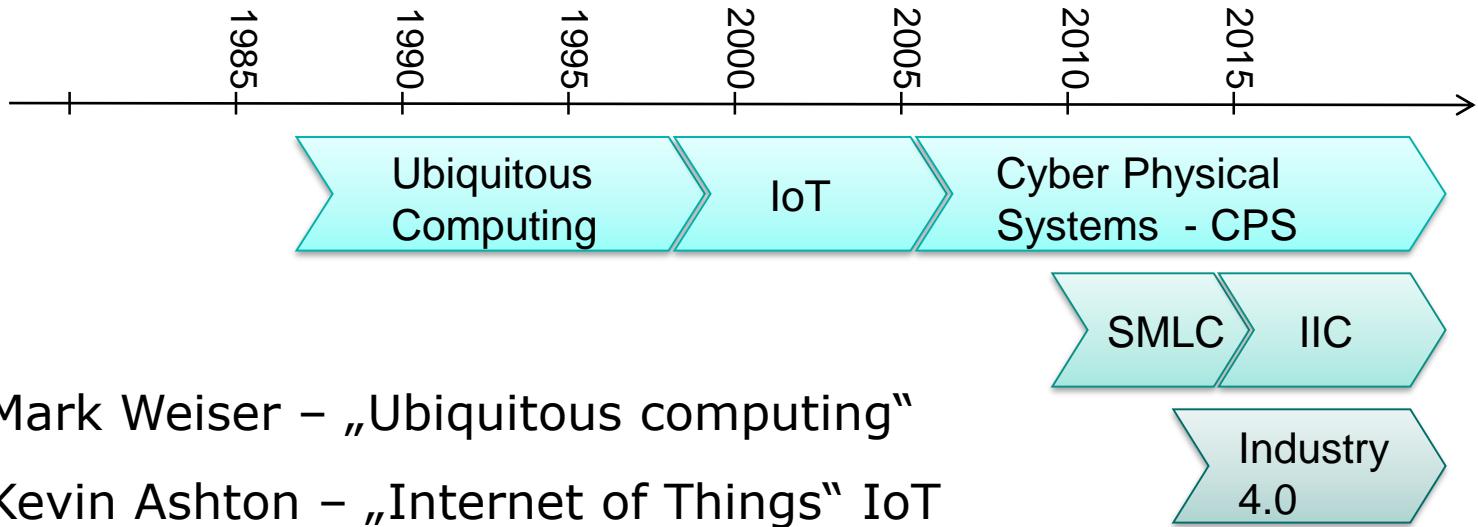
Historical Developments

Industrial Revolution and Version Numbers



The Internet of Things

Time development of IoT



Industry 4.0

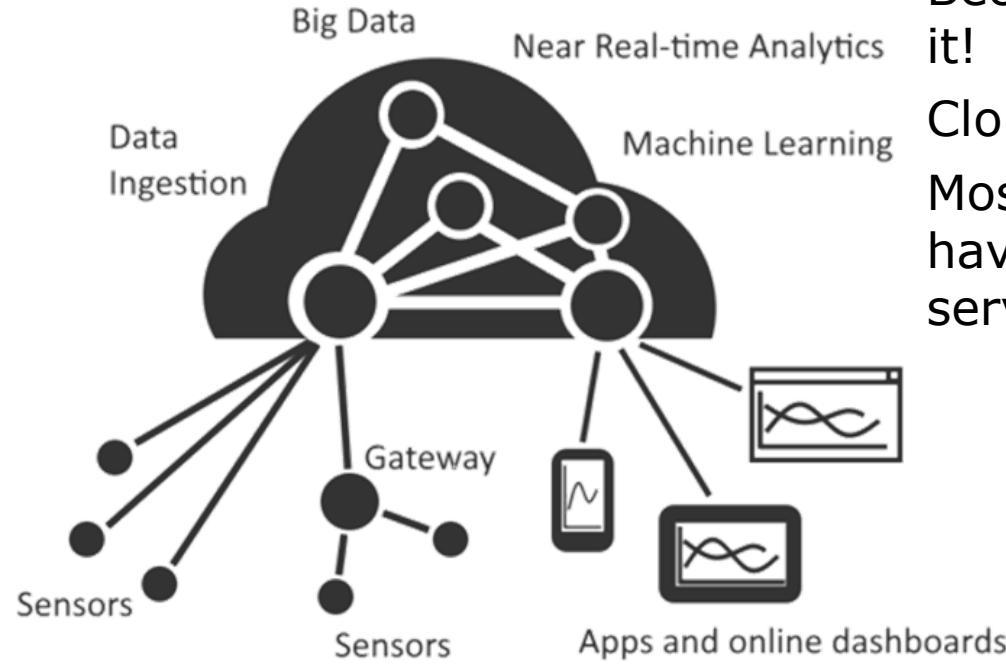
And another description

- > „Machines, Products and production lines continuously exchange data.“
- > The internet of things (IOT)
- > „All production and logistic procedures will be integrated.“
- > Adaptive machines with the possibility of reconfiguration at no additional costs.
- > Lot size one and "Customization"

Main topic seems to be flexibility!

Industry 4.0

Why coping with Industry 4.0 solutions?



...and – people say - there is a need to implement it in technical systems and factories!

Because everybody is talking about it!

Cloud solutions are trendy!

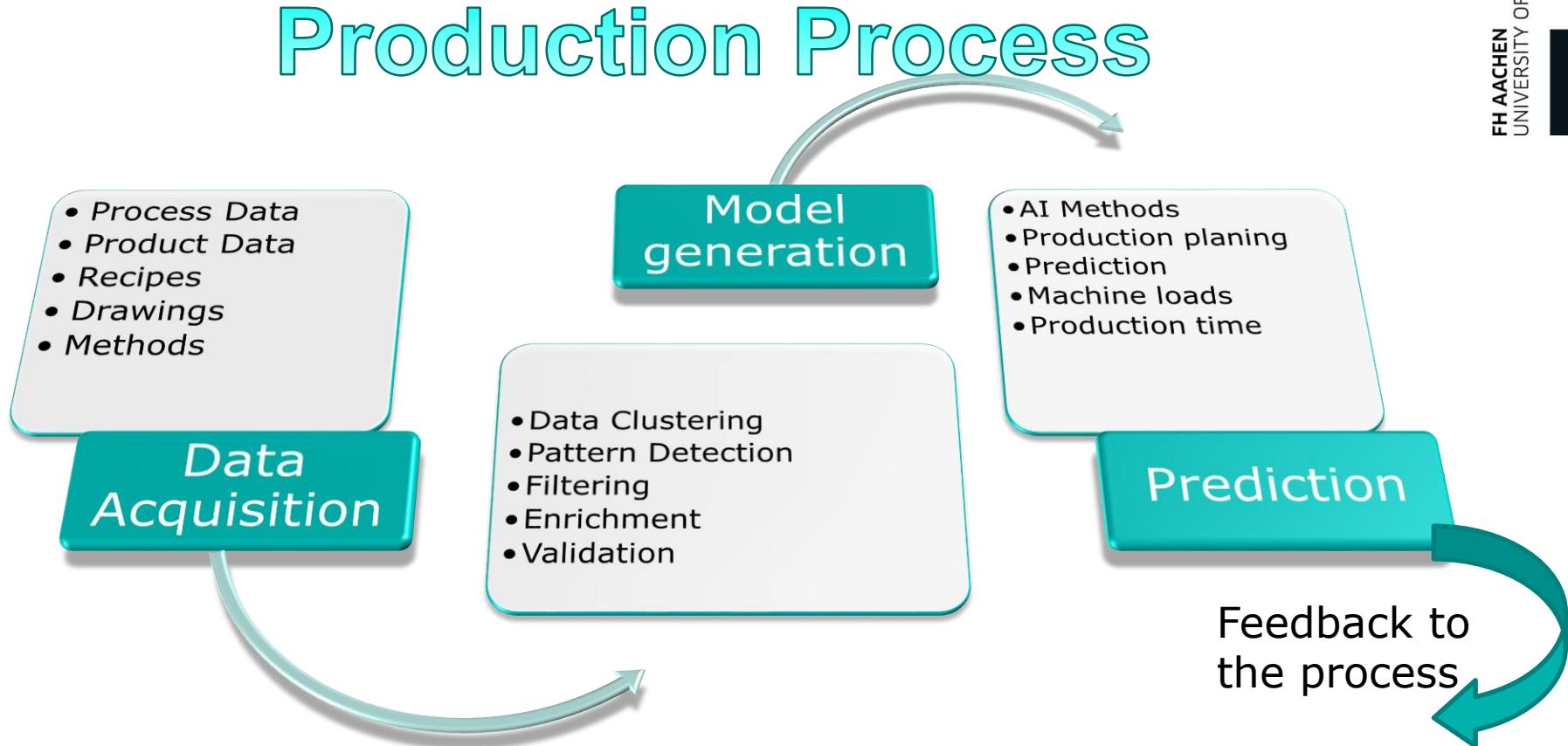
Most of the emergent technologies have something to do with cloud services!

A lot of new buzzwords:

- > Big Data
- > Data Ingestion
- > Real time data analytics
- > Machine learning
- > Application

Another Vision of Industry 4.0

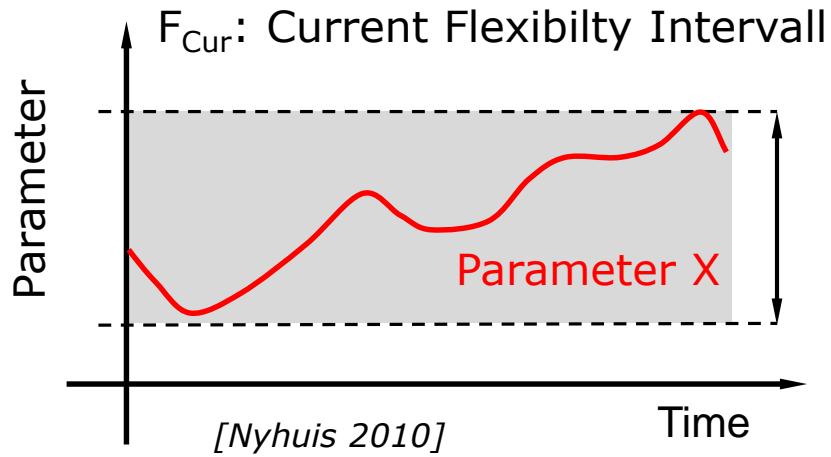
Flexibility of the complete production line



Flexible reconfiguration of the process in real time.
The factory changes over time, again: flexibility.

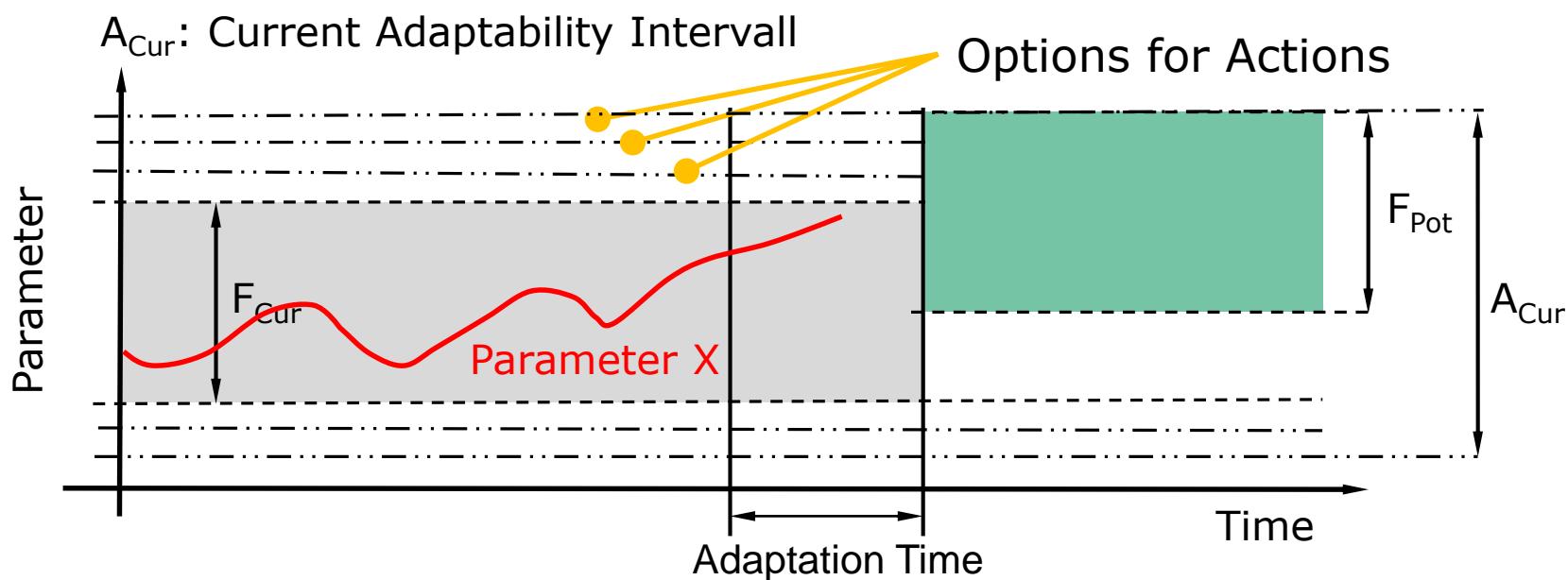
Industry 4.0

Flexibility and Adaptability



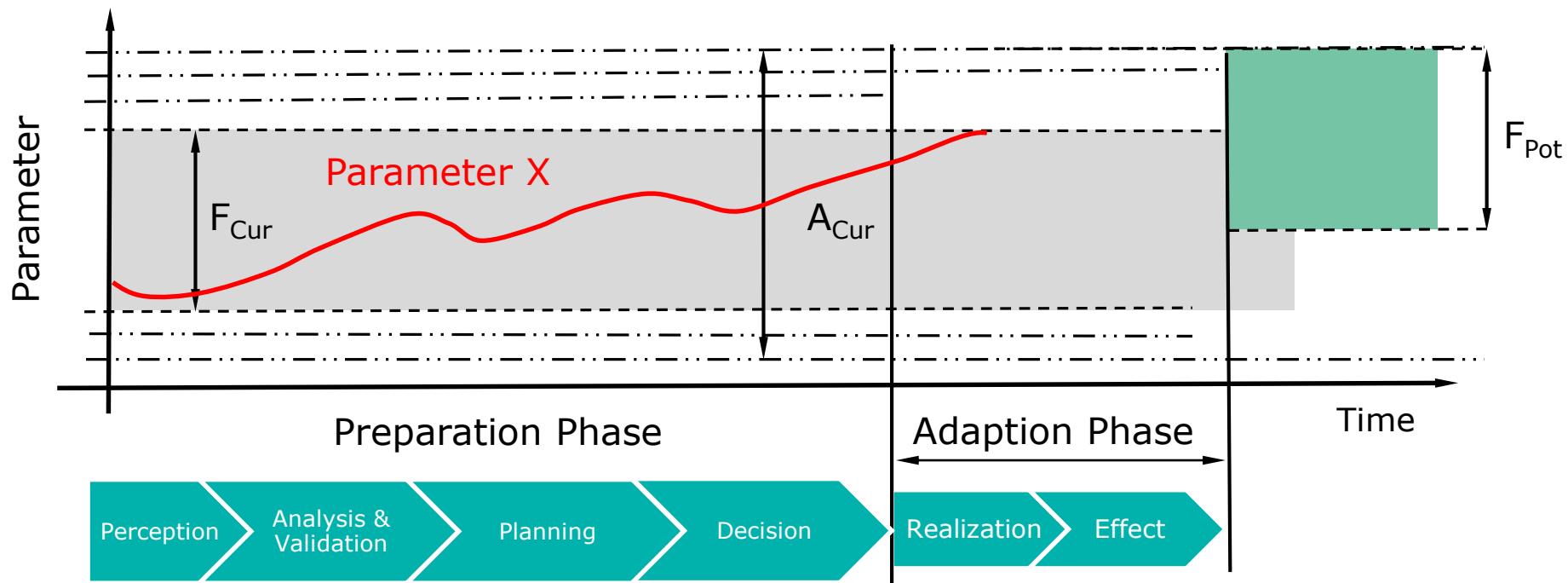
Possible Parameters

- > Stock Product X
- > Nr. of Sales of Product X



Industry 4.0

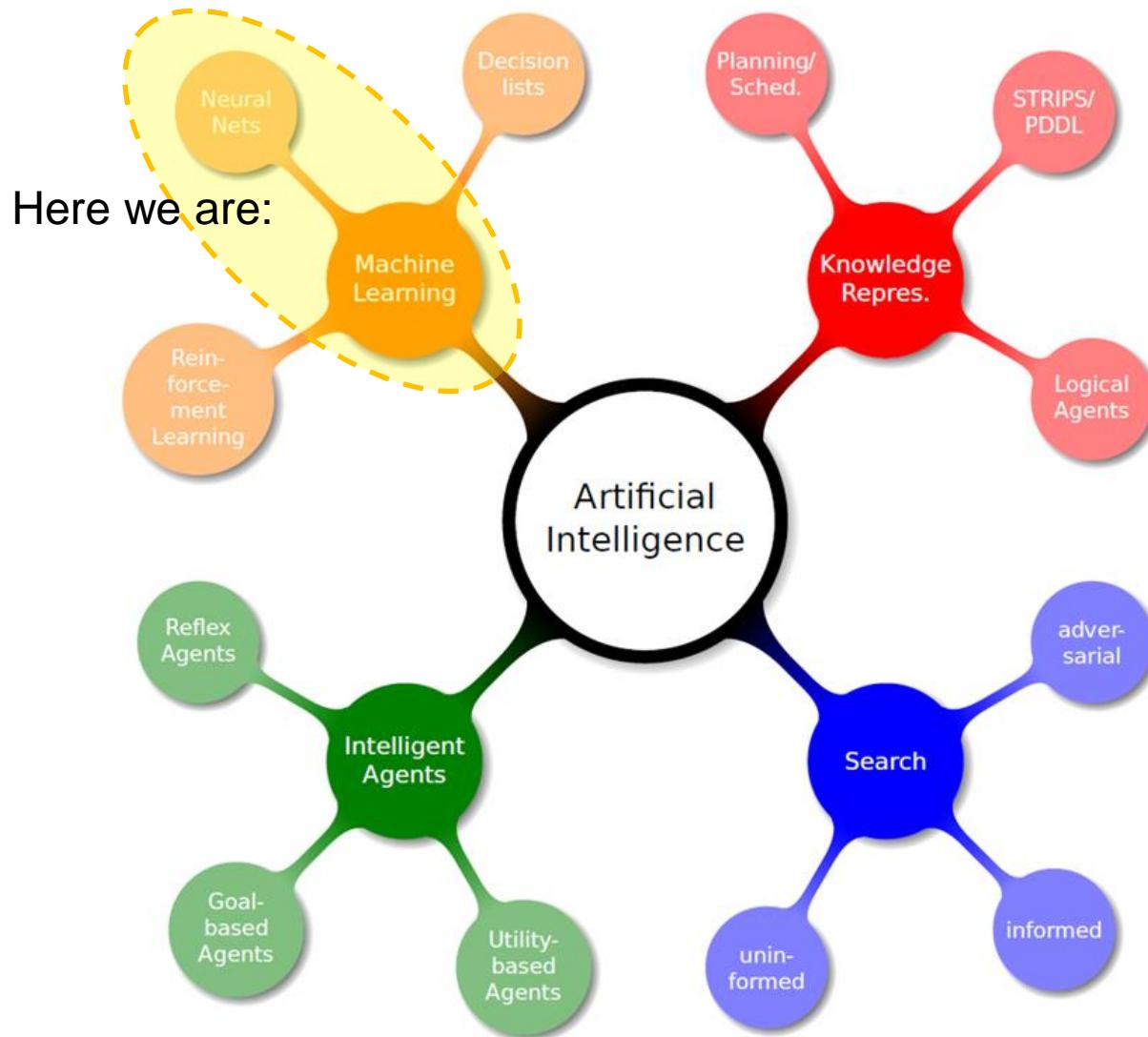
Flexibility and Adaptability



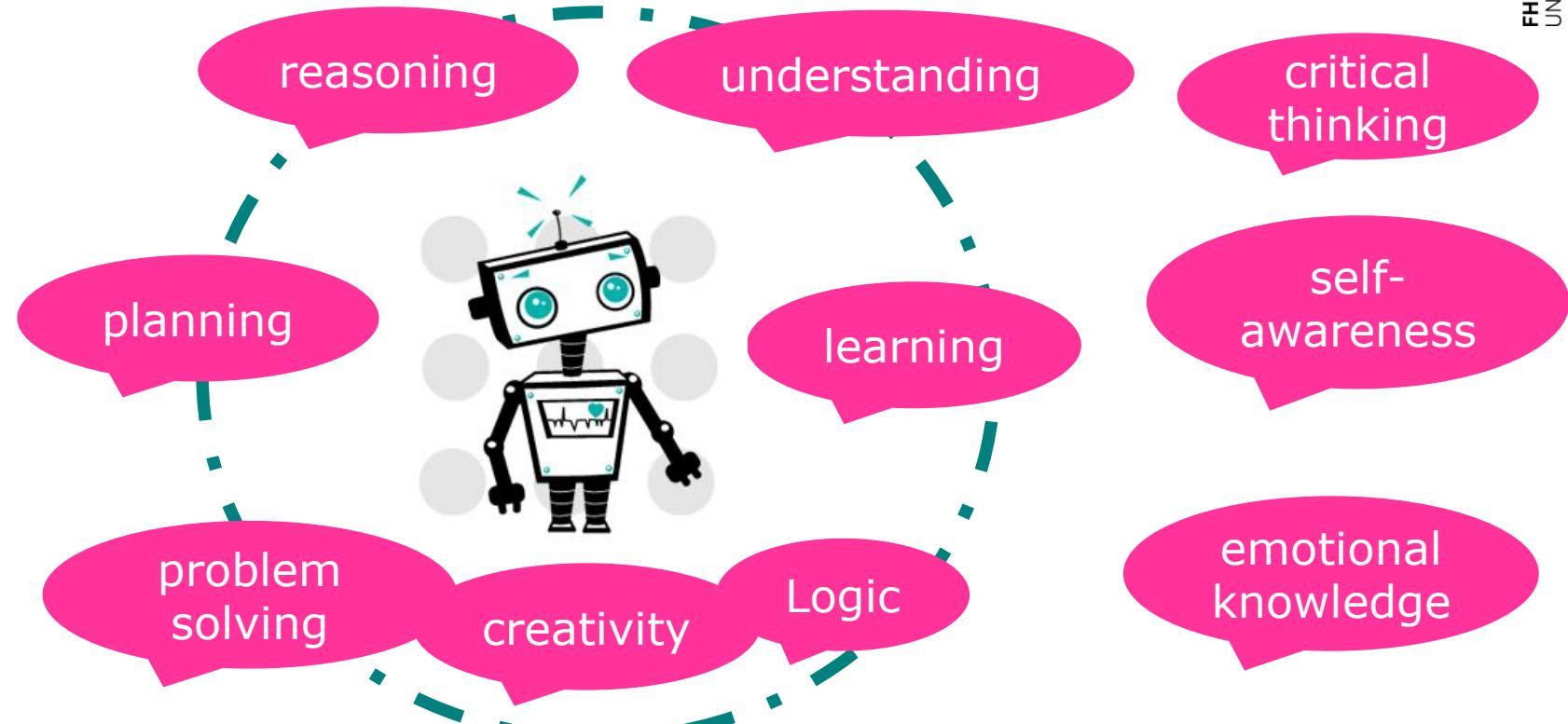
Very similar to mobile Robotics and AI!

Artificial Intelligence

An Overview

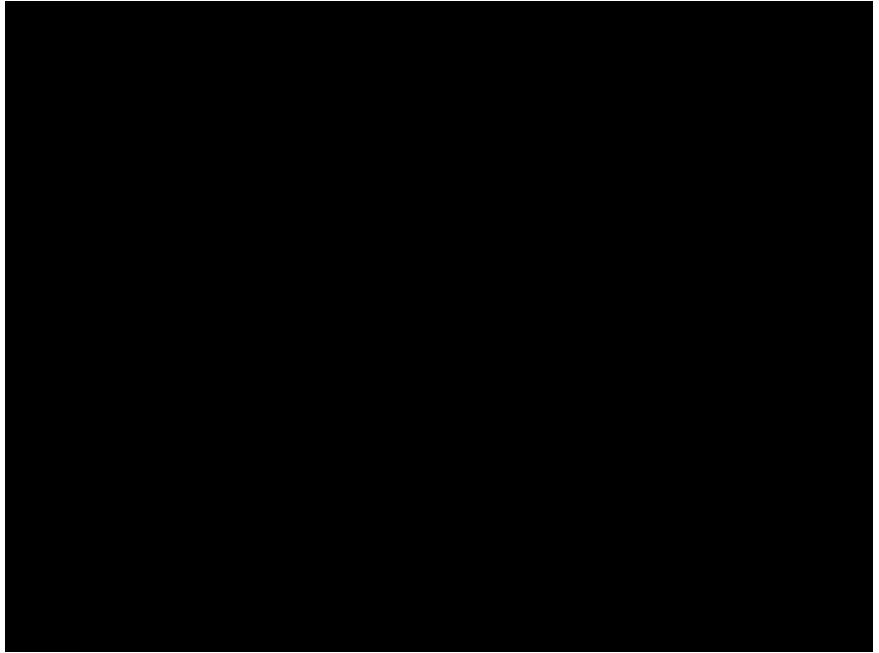
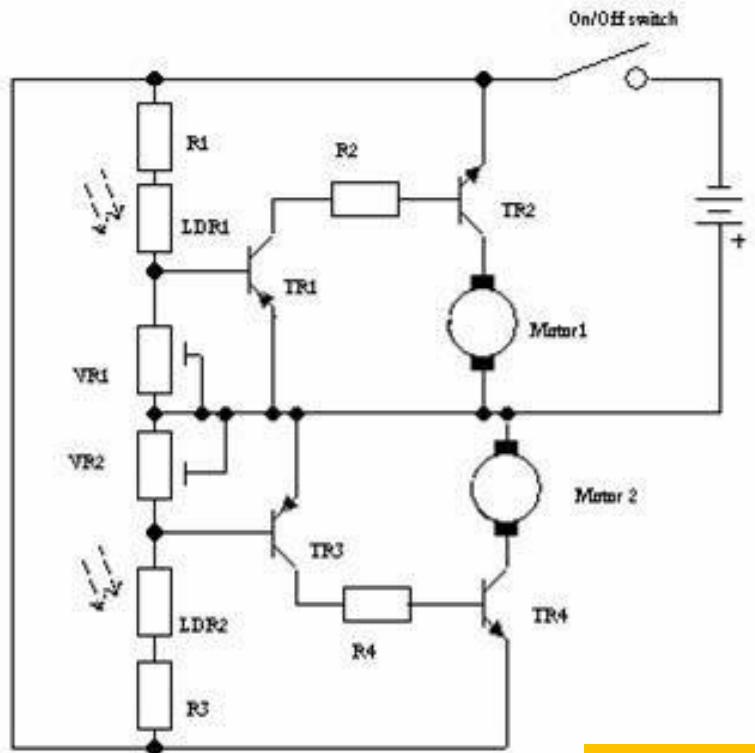


What is intelligence? Wiki says: **the capacity for...**



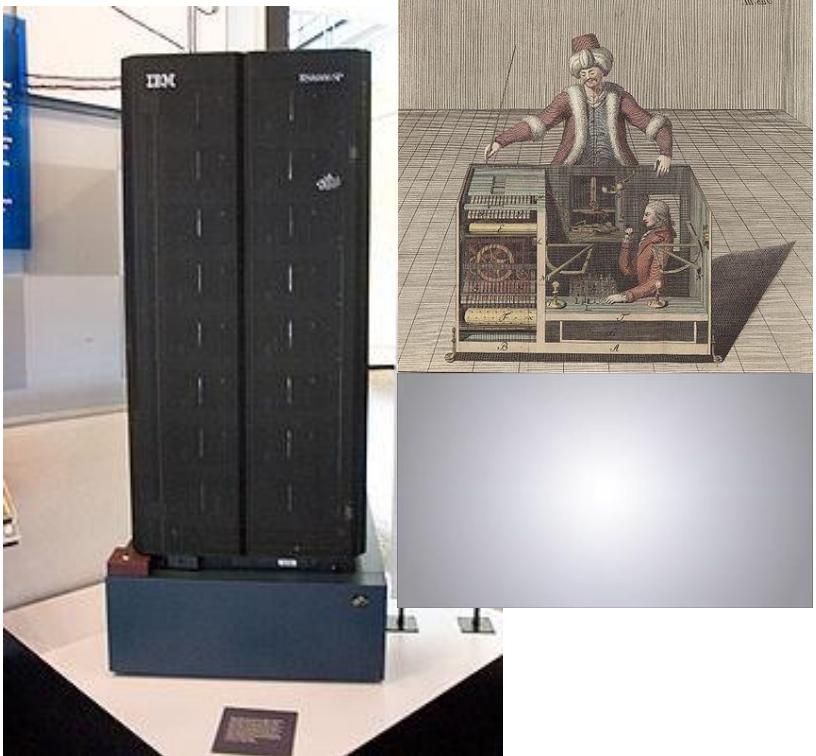
Robots are able to do this

An **intelligent** circuit...? The way we interpret intelligence.



Simple behaviour but interpreted as intelligence

When is a **machine intelligent?** Playing Chess



In 1957 Herbert Simon claimed, that a human Chess Grandmaster will soon be beaten by a computer. He did it again in 1990. Finally 1997 Deep Blue won against Kasparov. Specially designed hardware, no intelligence.

But it shows that complex problems can be solved with a lot of money, algorithms and hardware.

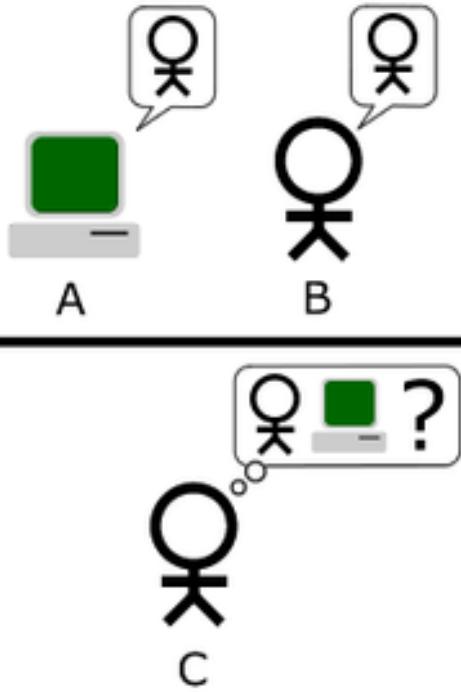
When is a **machine intelligent?** Playing Go and Chess and discover creativity.



Nature paper:
Mastering the game
of Go with deep
neural networks and
tree search

After beating the chess Grandmasters the asian board game „Go“ was the next challenge. In March 2016 AlphaGo won against a human Go Master. End of 2017 a new version, AlphaGo Zero, used a complete new strategy by learning the board game from scratch, so no human training examples were used. End of 2018 Alpha Zero won against all digital versions of Chess, Go and Shogi...

Another test for machine intelligence: The Turing test.



An evaluator C should distinguish between a machine and a human just by asking questions via a keyboard and screen. If the evaluator cannot tell the machine from the human the machine has passed the test.

In 2014 The Turing test was claimed to be passed by a chat bot named Eugene Goostman. He pretends to be a 13 year old boy from Ukraine not speaking english very well ...

A chat with Eugene I

Business Insider: Hi Eugene, What's up?

Eugene Goostman: I am fine, thanks, and you?

BI: I'm good. You doing anything fun today?

EG: Since early this morning I've been involved in this funny contest. I also plan to visit some interesting places in Atlanta.

BI: Why Atlanta? What's there?

EG: East Coast is a good place, but I heard about sharks and typhoons that terrify people there.

BI: So... you're going to Atlanta?

EG: If you sure that you're not eatable for sharks, I think the Eastern Coast is a very nice place for living and swimming in the Atlantic Ocean:-)

BI: Do Sharks prefer one type of human over another?

EG: Maybe. I think *you* know the answer. I know you are supposed to trick me.

BI: Why would I want to trick you?

EG: Aha! Now I know what you want! If I'm not mistake - you still didn't tell me where you live. OR it's a secret?:-)

[<https://www.businessinsider.com/chat-with-program-that-passed-turing-test-2014-6?IR=T>]

A chat with Eugene II

BI: Huh? You tell me where you live first.

EG: I live in a big Ukrainian city called Odessa. It is on the Black Sea shore.

BI: So Atlanta is a far trip for you, no?

EG: East Coast is a good place, but I heard about sharks and typhoons that terrify people there.

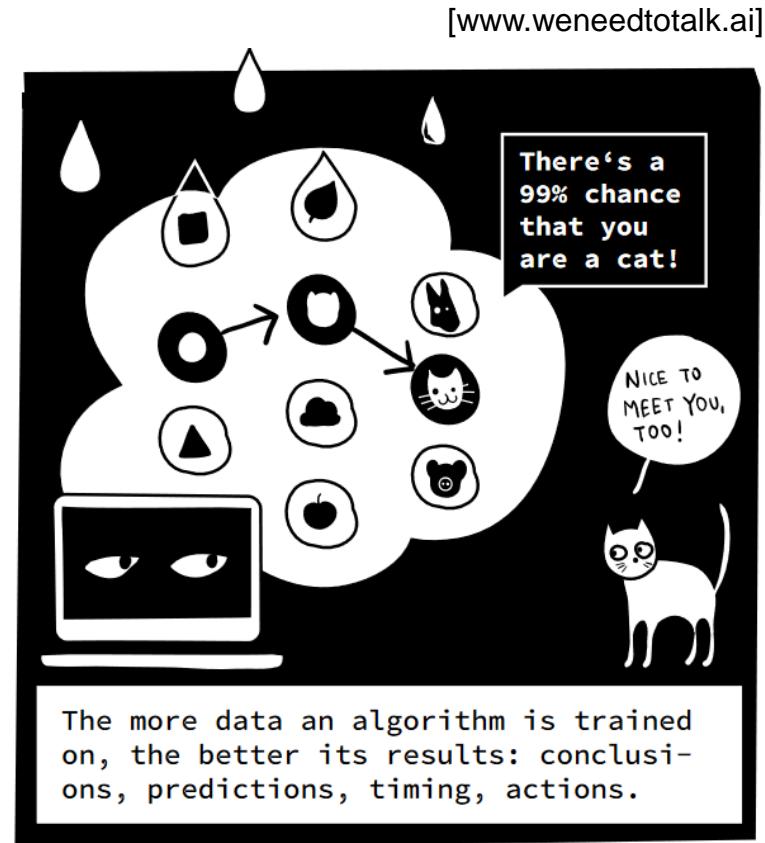
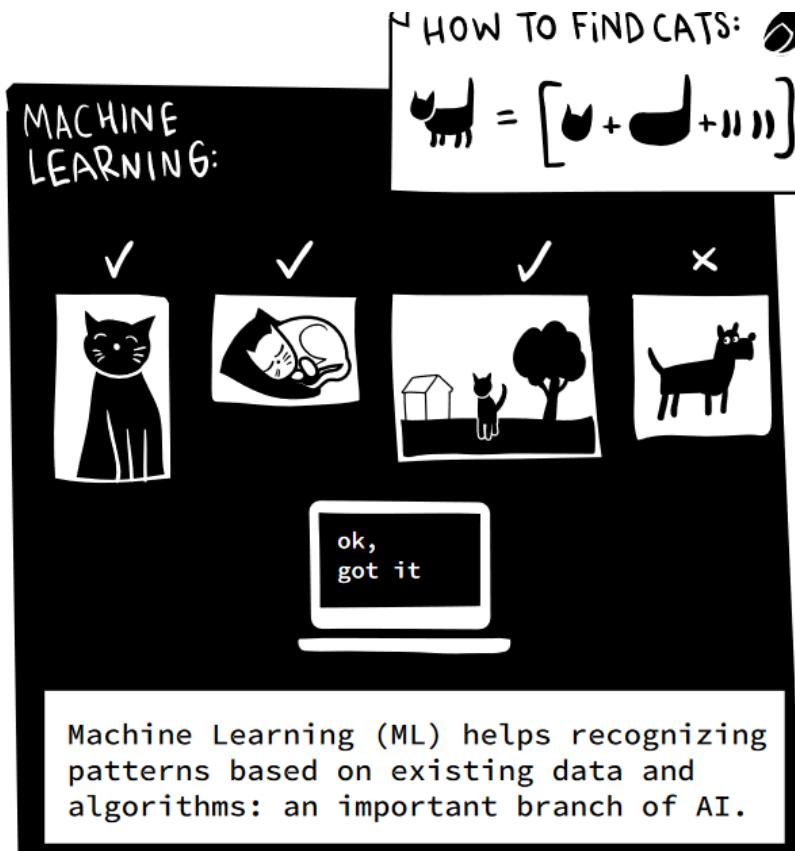
BI: And we've come full circle. Are you a human?

EG: No. Beep-beep. I am not a ma-chine. Blink-blink. I am hu-man. Click! Hu-man. Click! Hu...
Damn.

Simple questions like "Why are you going to Atlanta?" are answered with a totally irrelevant statement. And even with the same response when brought up again. The chat engine picks out one word from each of the questions and just ran with it.

[<https://www.businessinsider.com/chat-with-program-that-passed-turing-test-2014-6?IR=T>]

But there is hope for intelligent machines: Machine Learning



Machine Learning needs data. A lot of data...



Only with the invention of the internet, we were able to collect and store enough data to make AI work.



Too much data for us to process. Just the right amount for AI.

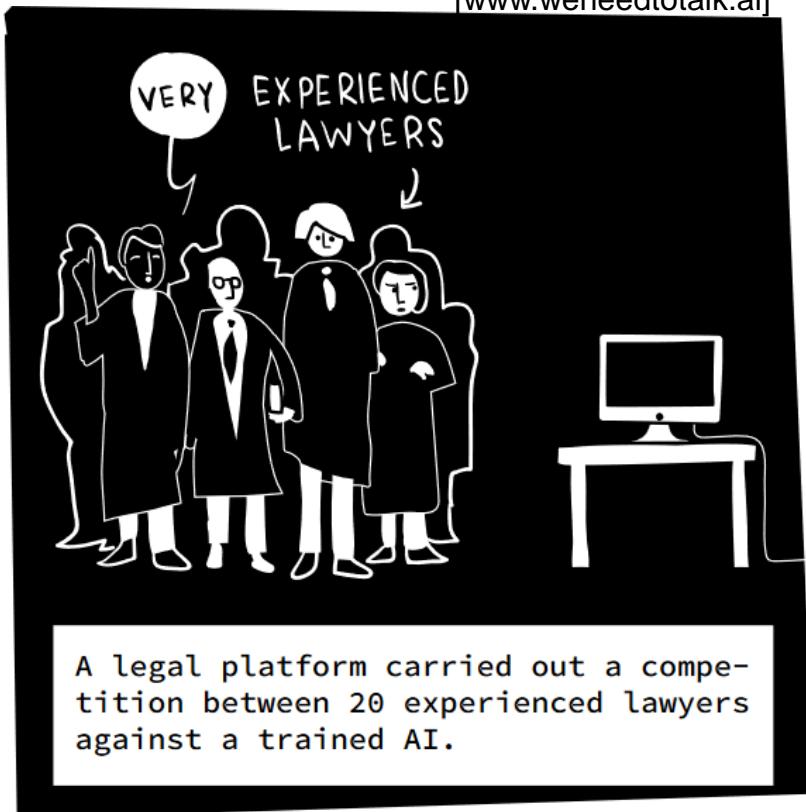
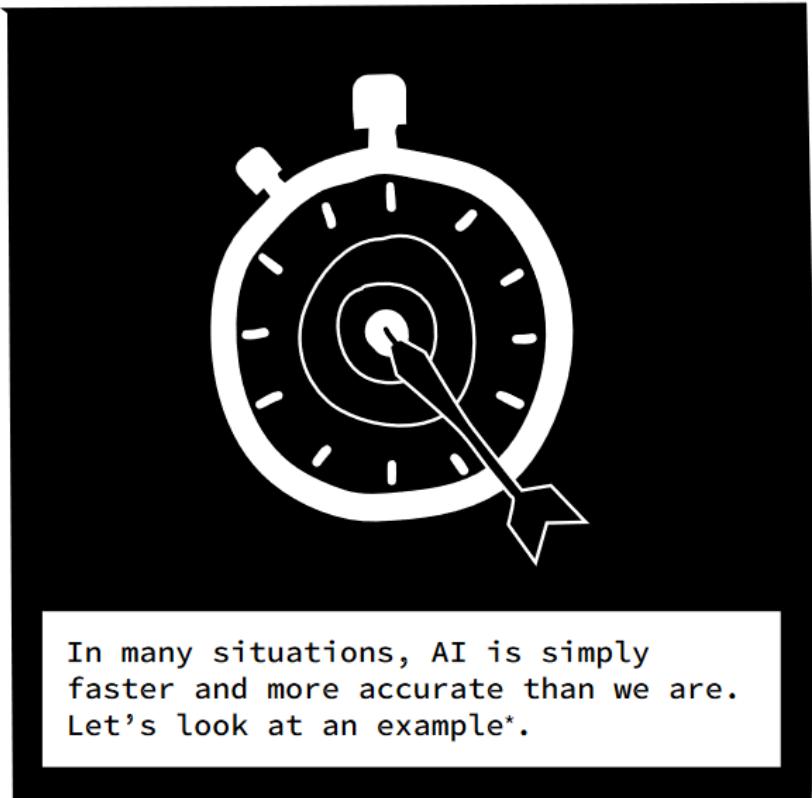
Machine Learning or better: Pattern Classification



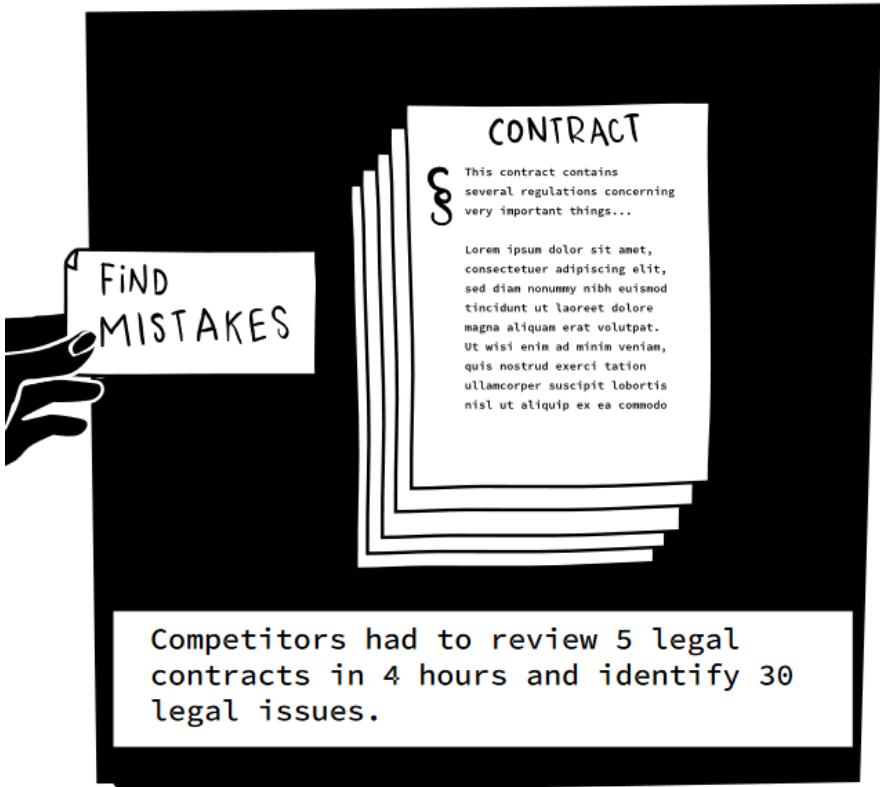
Neuro Time

Problems from the field of pattern classification, can be solved by artificial neural networks by training with a lot of examples. The detection can be done in realtime e.g. for autonomous driving.

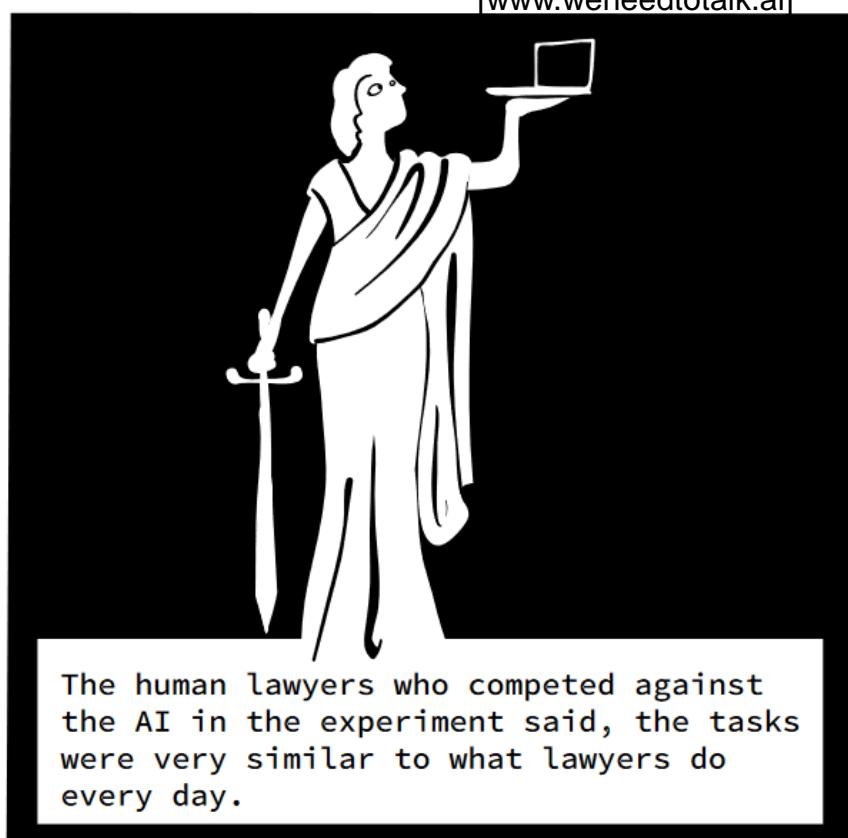
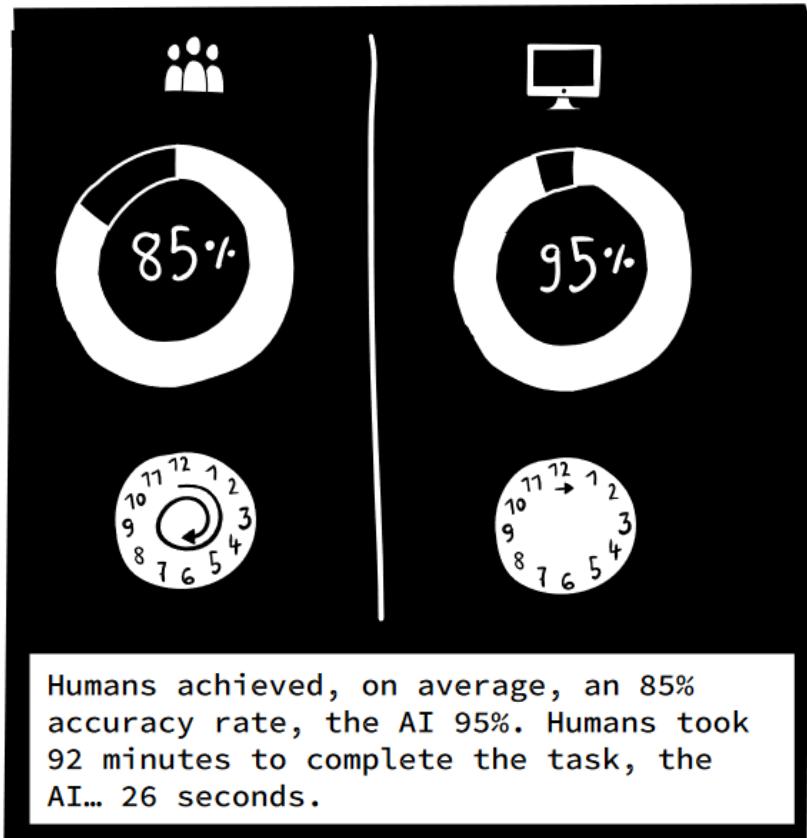
Another promising example



Any **Lawyers** around?

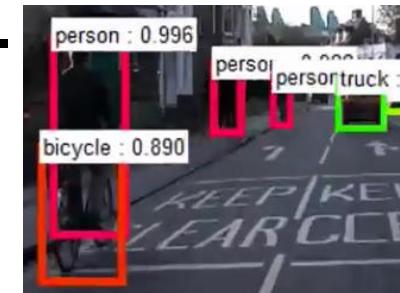


Don't worry: that's not possible with **farmers**...



A short **Summary** of intelligent machines

- Lawyers can be replaced.
- Classification Problems can be solved by machine learning with extensive training.
- We need a lot of data for machine learning.
- A certain behaviour is interpreted from humans as intelligent.
- Human „behaviour“ is still difficult to simulate (chat bot).
- Newest algorithms can be even „creative“.

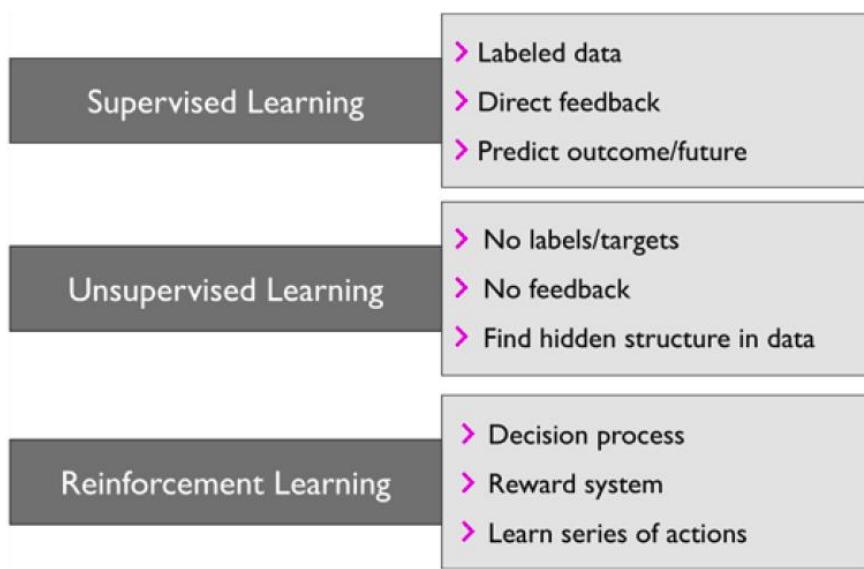


Machine Learning

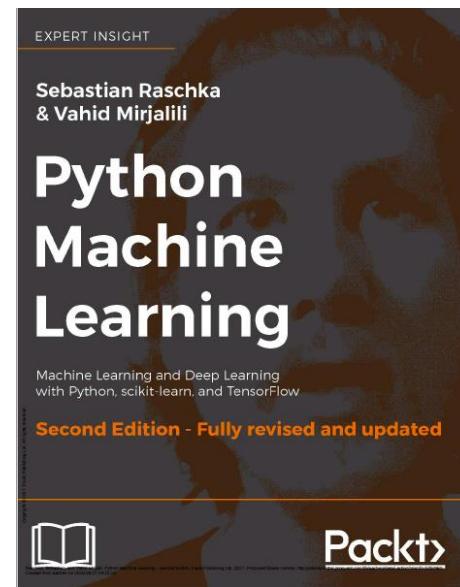
Introduction

Large amount of data is available for us. In order to analysis this data machine learning can be used to:

- > Transform data into knowledge
- > Recognize and classify patterns and
- > Predict future events.



[Raschka]

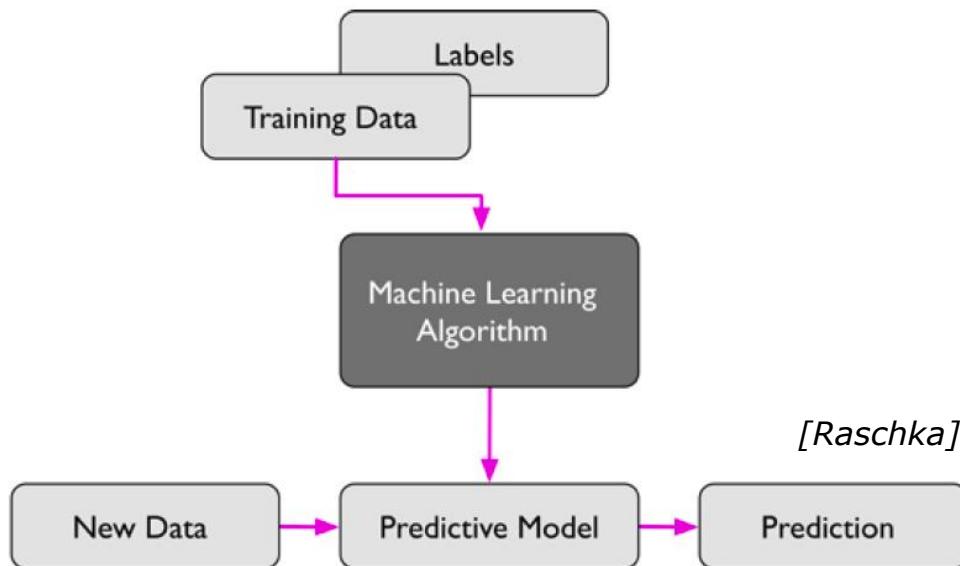


Machine Learning

Supervised Learning

Generating a model based on training data allows the classification of new data

- > Label training data
- > Adapt the output of a predictive model (learning) and
- > Classify new data (patterns) or predict future events.

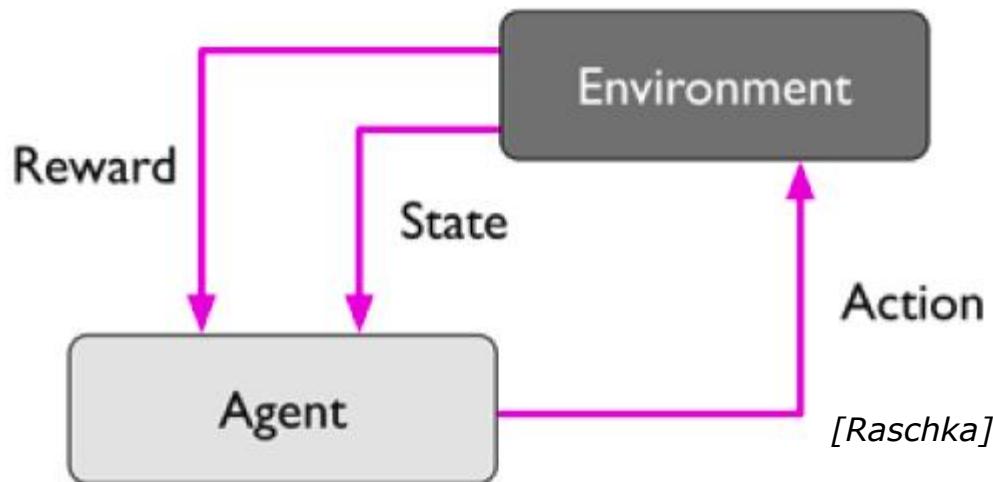


Machine Learning

Reinforcement Learning

Improve the performance of the system interacting with the environment by a reward function

- > Apply a series of actions
- > Maximize the reward signal
- > Immediate or delayed feedback can be used

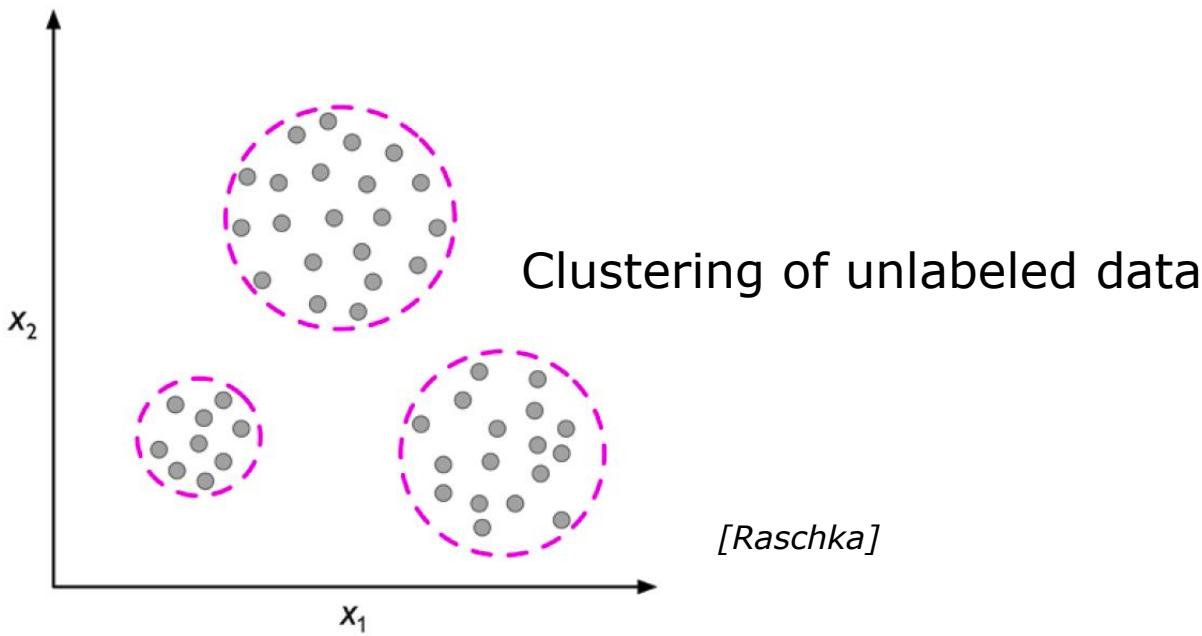


Machine Learning

Unsupervised Learning

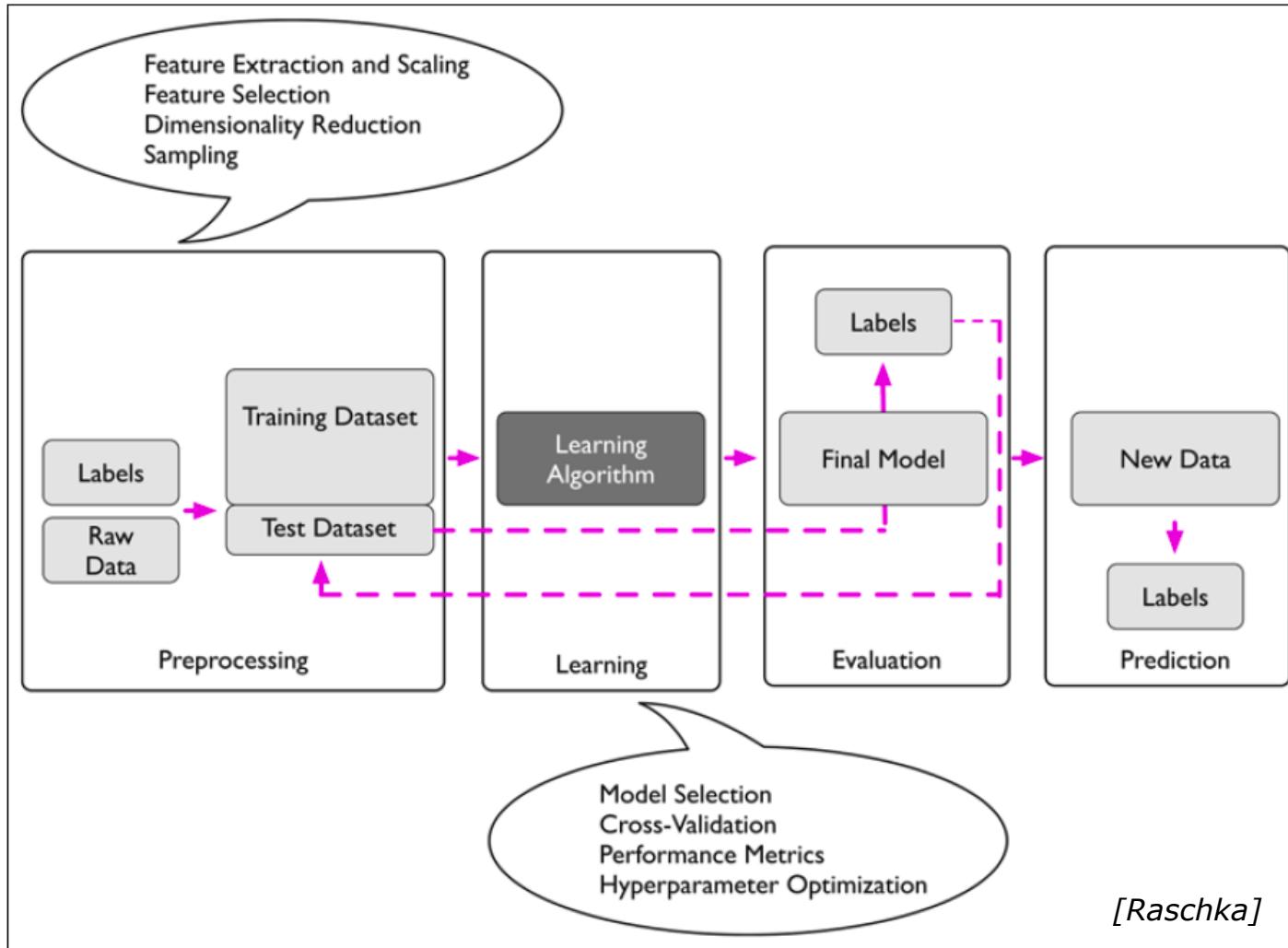
Detect hidden information in data with unknown structure

- > Try to find meaningful subgroups (clusters) in the data
- > Structure the information and
- > Try to find meaningful relationship btw. data



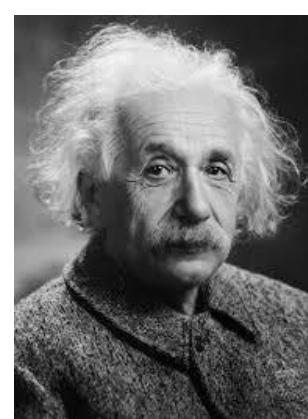
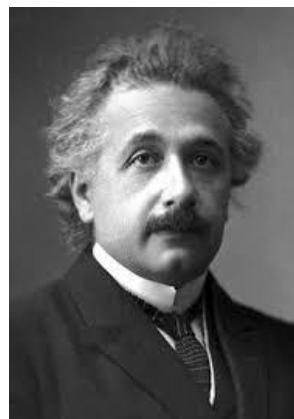
Machine Learning

Typical roadmap for a predictive system



A lot of tasks from the field of perception can be easily solved by human beings wheras computers still have problems.

- > Identification of Persons, Objects, handwritten Characters
- > Speech recognition, Music, cognitive tasks, machine learning
- > Computer are often too slow, not flexibel enough, tolerant to variations in the data sets

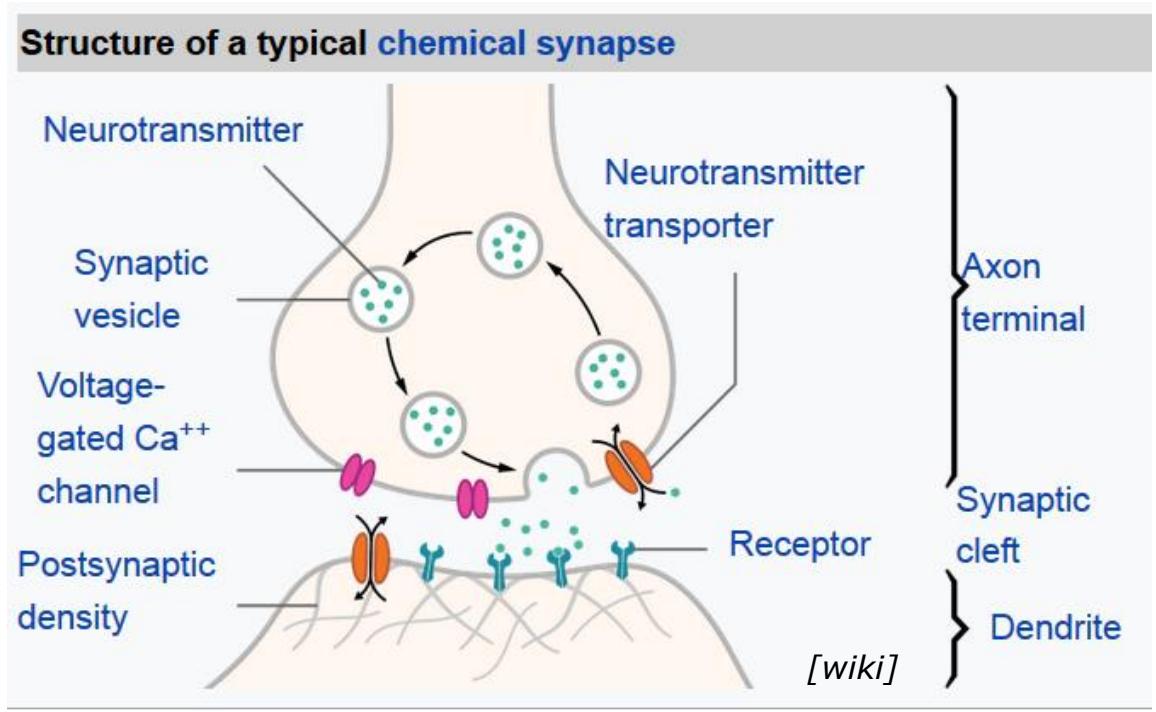
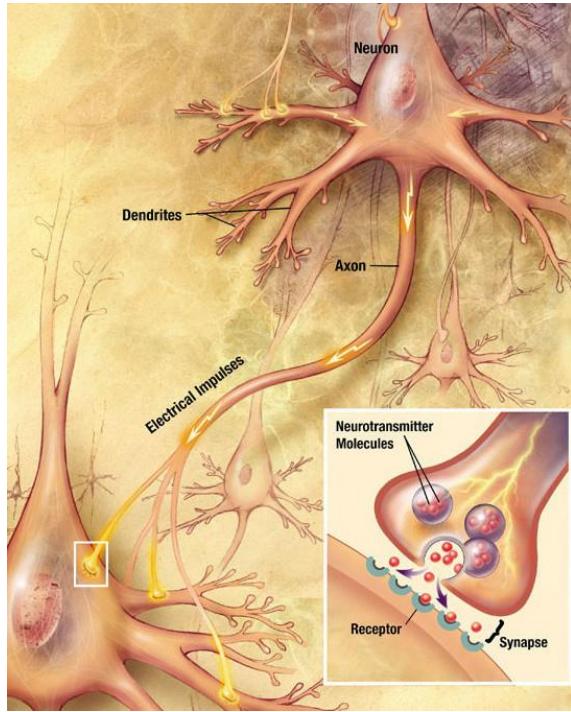


Is this the same person?

Neural Networks

Introduction and Biological Inspiration

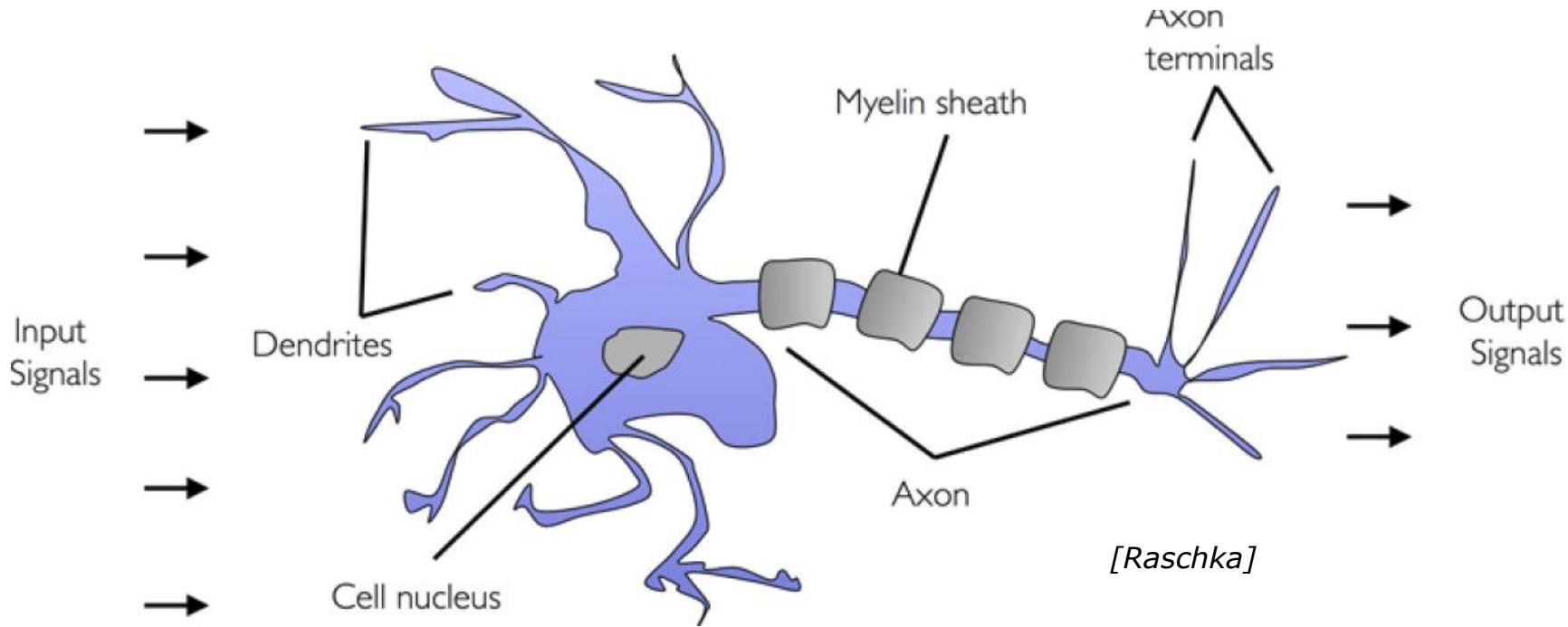
- > Are a simplified „model“ of the human brain
- > The brain is quite slow $\sim 10^{-3}$ sec but massively parallel (Connectionism)
- > Remain functional even if parts of the system are damaged. Loose slowly their initial function as damage proceeds.



Neural Networks

Biological Inspiration

- > The biological brain consists of Neurons, which are connected with a large amount of other Neurons via Synapses.
- > They are activated by a stimulus.



Neural Networks

Biological Function

A more detailed explanation:

- > The end of the axon sends neuro transmitters.
- > They effect at the membran of the receiver dendrite is a change in polarization. The inner part is typically -70mV compared to the outer part.
- > A reduction in the potential difference is a stimulating synapse
- > An increase in the potential difference is an inhibiting synapse
- > If enough stimulating information is available, the axon get depolarized.
- > The resulting action potential is transmitted along the axon, where the speed depends from the coverage with Myelin
- > If the end of the synapse is reached, neuro tranmittters are generated for the next neurons

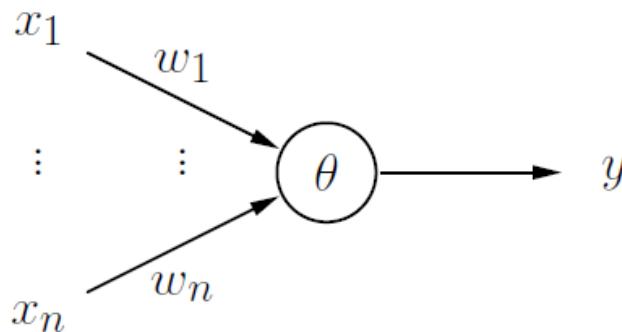
[according to Kruse]

Neural Network

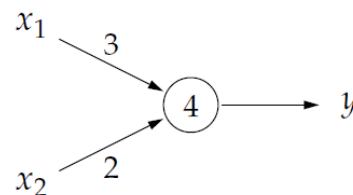
Thresholding Units/Perceptrons

The neuron is modelled as an element with inputs x_i , weights w_i and a threshold unit θ .

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^n w_i x_i \geq \theta, \\ 0, & \text{else} \end{cases}$$



Threshold Unit for $y = x_1 \wedge x_2$



x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

AND

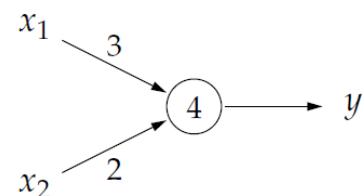
[Kruse]

Neural Networks

Linearly separable Point Sets

Point sets, which can be separated with a $(n-1)$ dimensional hyper plane, are linearly separable.

Perceptron for

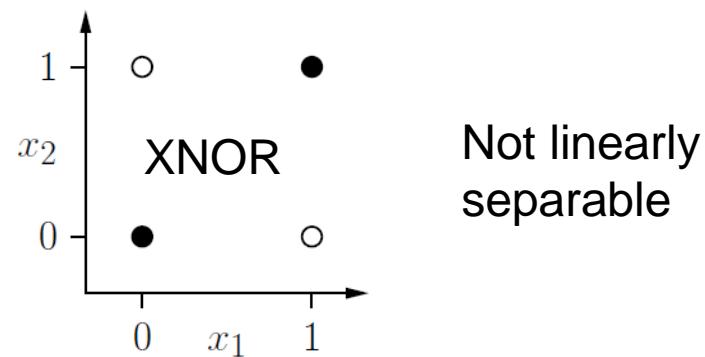
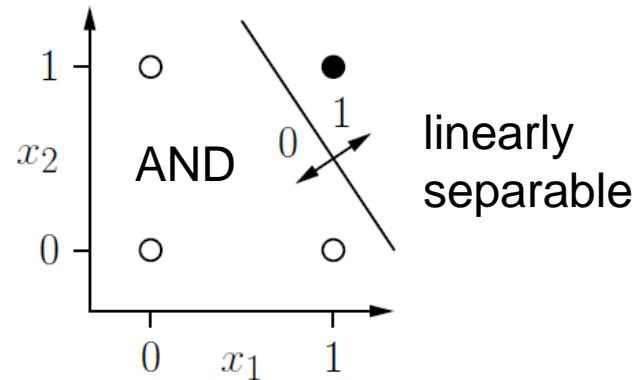


$$y = x_1 \wedge x_2$$

x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

[Kruse]

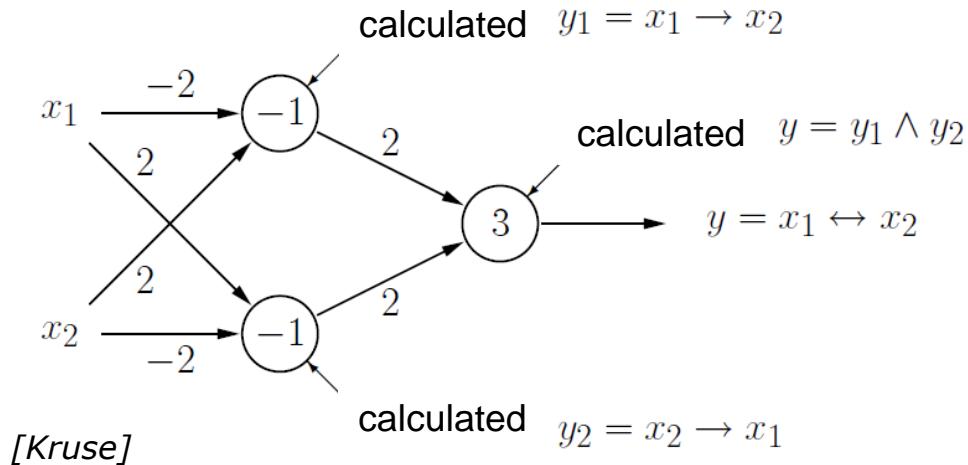
x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1



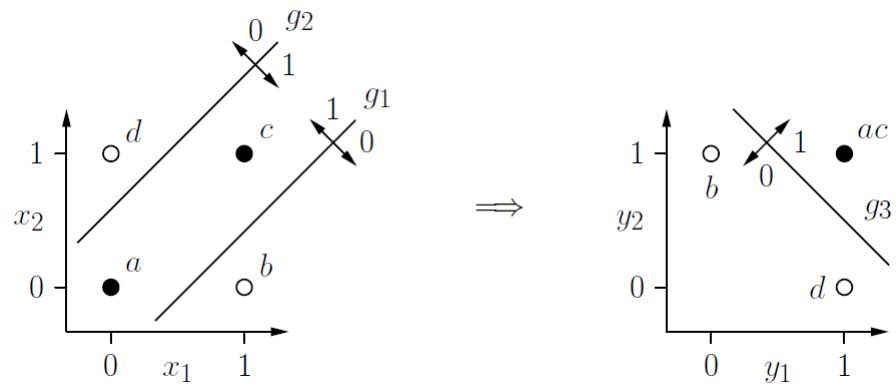
Neuronal Networks

Nets from Perceptrons

Numerous functions cannot be calculated just from a single Perceptron -> Net consisting of several Perceptrons is necessary.



- > First Layer calculates new boolean coordinates for the point sets
- > After this transformation, the problem is linearly separable



Neural Networks

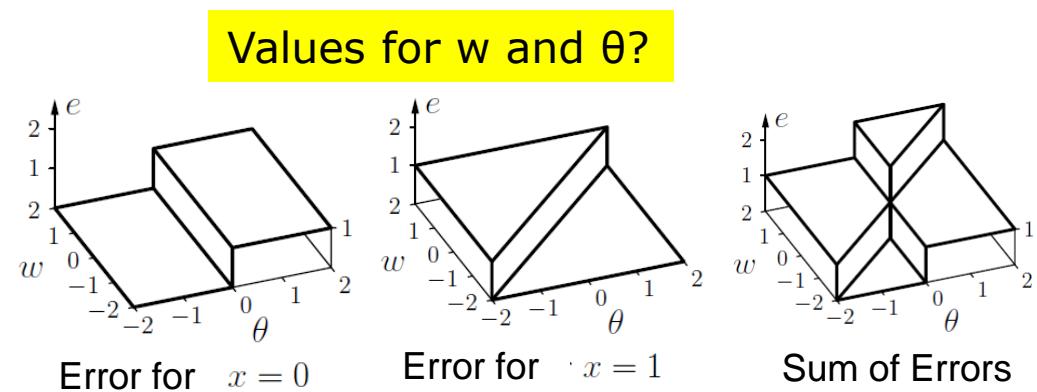
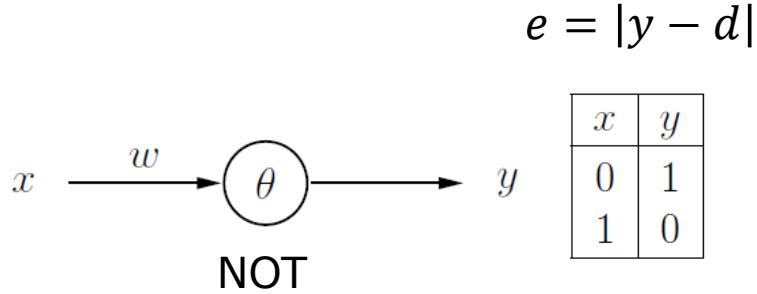
Determine the weights and the threshold

Geometrical Interpretation of the Perceptron Inputs only for up to three. Automatic Adjustement (Training) is necessary.

Basic Training algorithm:

- > Start with random values for weights and threshold
- > Calculate the Error for a Set of training data
- > The error is a function of the weights and the threshold $e = (w_1, \dots, w_n, \Theta)$
- > Change the weights and the threshold in order to minimize error function
- > Repeat the procedure until error is minimal

[Kruse]



Neural Networks

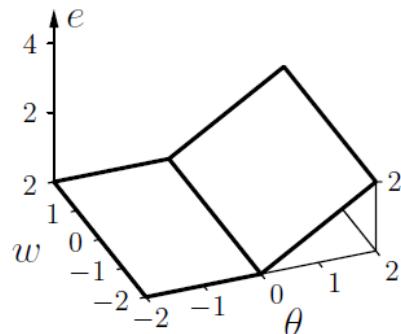
Training of single Perceptrons

Previous error function is adverse, due to the plateaus.

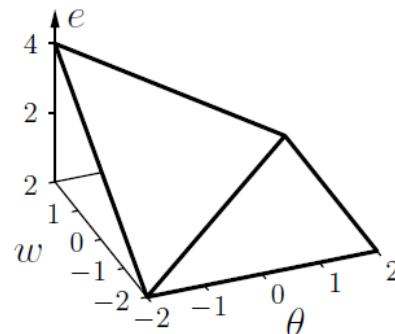
Better way: If the output of the perceptron is wrong, the distance of the weighted sum from the threshold is considered. This generates a direction for the algorithm to search for – the gradient descent algorithm.

$$w_j := w_j + \Delta w_j \quad \Delta w_j = \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)} \quad \text{Training rule for perceptron}$$

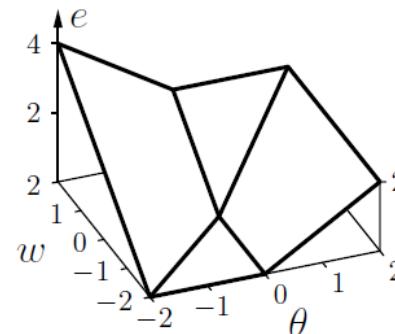
[Raschka]



Error for $x = 0$

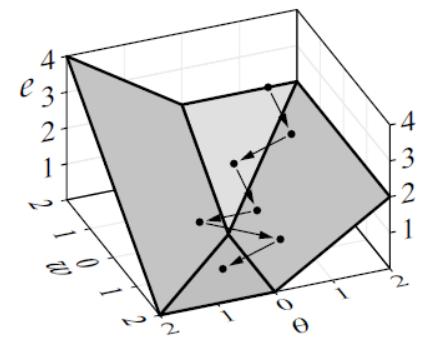


Error for $x = 1$



Sum of Errors

[Kruse]



Gradient descent algorithm

Neural Networks

Summary Perceptrons

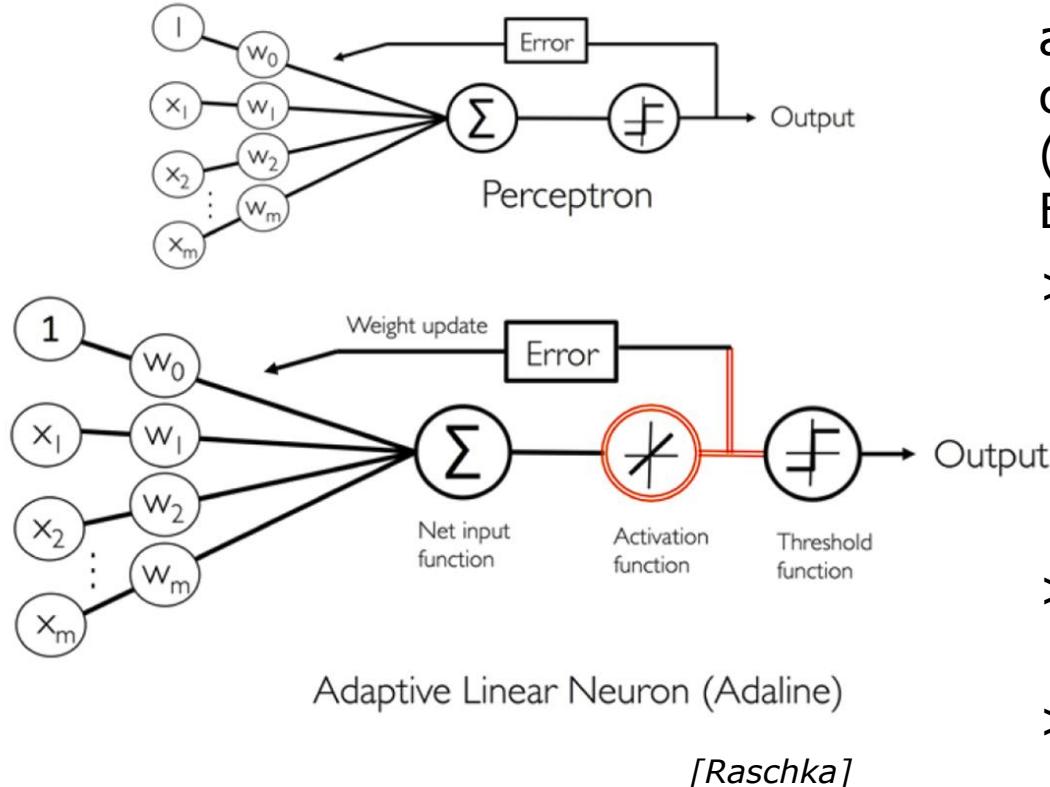
- > Single perceptrons can only calculate linearly separable functions.
- > Perceptron Networks can calculate any boolean function.
- > The training with the delta rule of the perceptron is guaranteed to find a solution, if the solution exists.
- > Networks from perceptrons cannot be trained with the delta rule.

That doesn't make us
too happy!



Neural Networks

The adaptive linear Neuron (ADALINE)

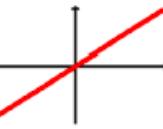
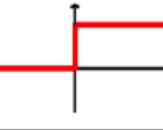
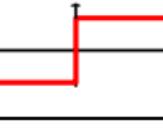
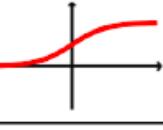


Neural networks can be seen as a **directed graph** consisting out of nodes (Neurons) $u \in U$ and $c \in C$ Edges (Connections).

- > The set of nodes U is divided in: U_{in} (**Input Neurona**), U_{out} (**Output Neurons**), U_{hidden} (**hidden Neurons**)
- > Each Connection $(v,u) \in C$ has a weight w_{uv}
- > Each Neuron has three state functions:
 - > Network Input net_u ,
 - > Activation act_u ,
 - > Output out_u

Neural Networks

Activation Function for MLPs

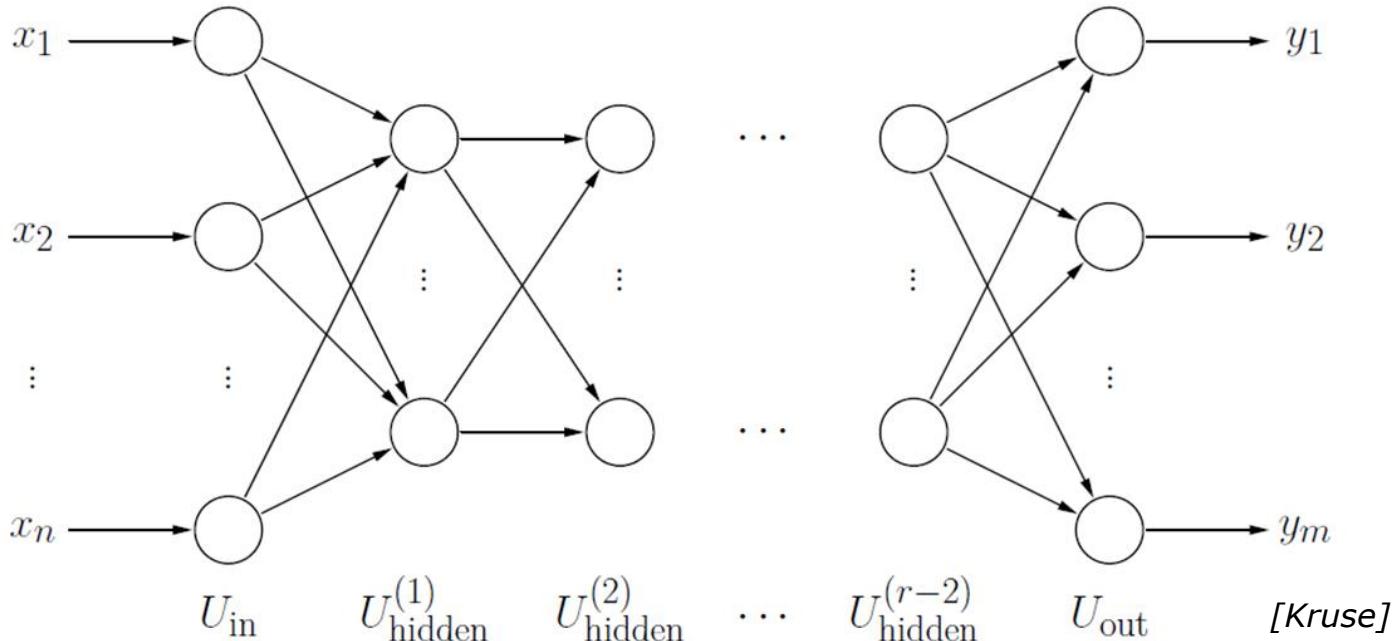
Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

[Raschka]

Neural Networks

Multi Layer Perceptron (MLP)

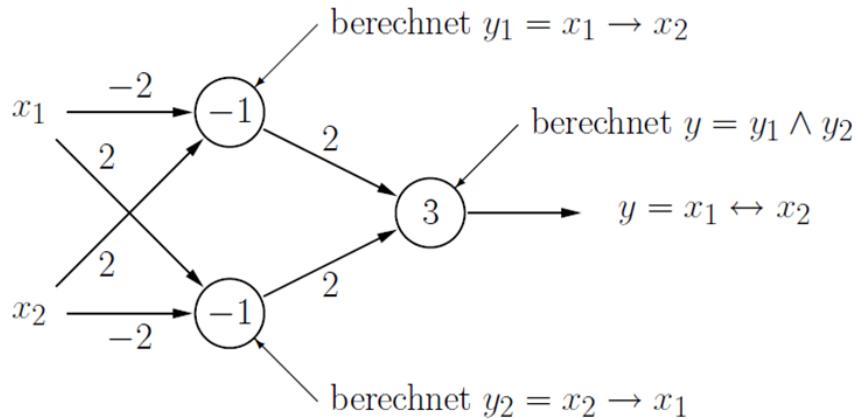
An often used multilayer feedforward Network is the MLP.



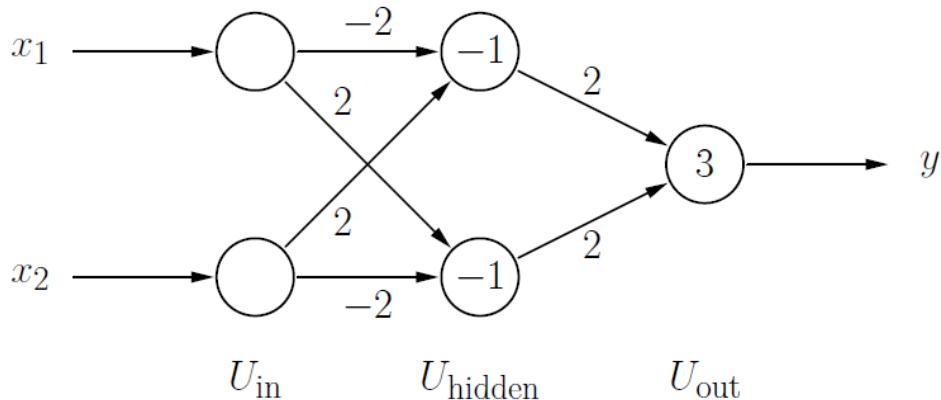
MLP with r layers

Neural Networks

Example with MLP



Solving the XNOR problem
with three perceptrons



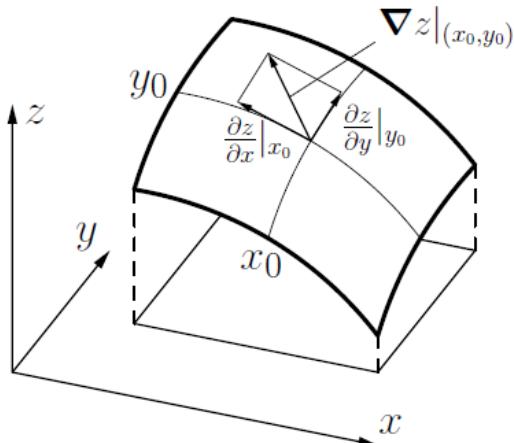
Solving the XNOR
problem with MLP

Neural Networks

Training of a MLP

The training of a neural network is the adjustment of the weights and thresholding function – very much like in **Regression analysis**.

- > But: Logistic Regression only works for two layer nets.
- > Since mid 80s the **Gradient descent method** with „**Error Backpropagation**“ from Rumelhart and McClelland gave the right tool to train any MLP topology – including hidden layers.

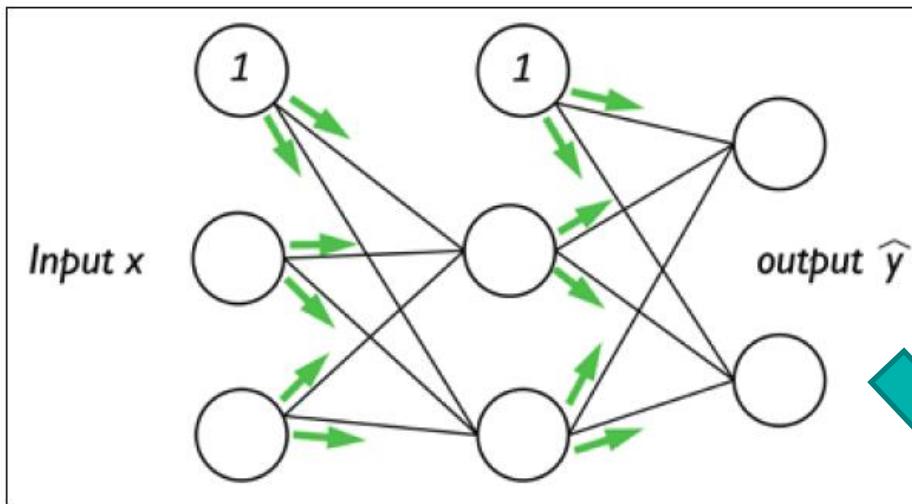


$$\Delta w_u^{(l)} = \eta \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su} \right) \text{out}_u^{(l)} (1 - \text{out}_u^{(l)}) \text{in}_u^{(l)}.$$

But the activation and output function need to be derivable!

Neural Networks

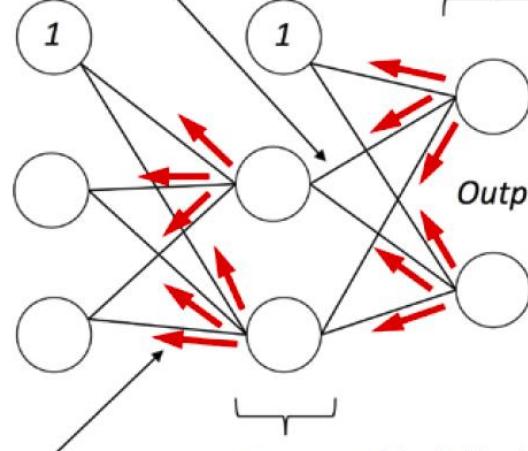
Overview Backpropagation Training of MLP



Compute the gradient:

$$\frac{\partial}{\partial w_{i,j}^{(out)}} J(\mathbf{W}) = a_j^{(h)} \delta_i^{(out)}$$

Error term of the output layer:
 $\delta^{(out)} = \mathbf{a}^{(out)} - \mathbf{y}$



Input x

Output \hat{y}

Target y

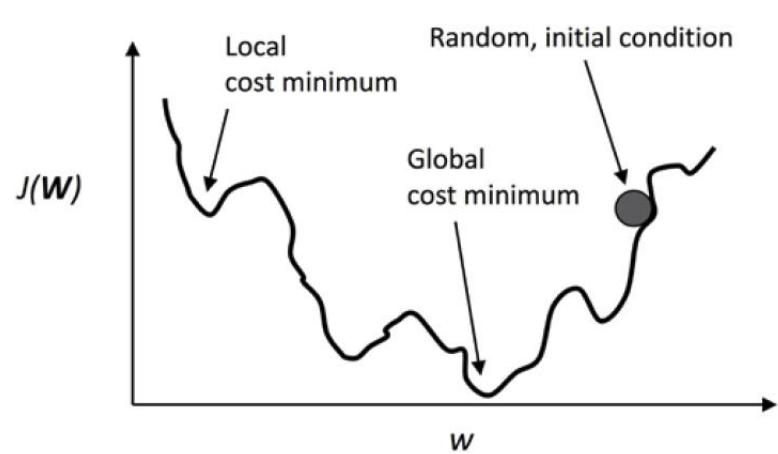
Compute the gradient:

$$\frac{\partial}{\partial w_{i,j}^{(h)}} J(\mathbf{W}) = a_j^{(in)} \delta_i^{(h)}$$

Error term of the hidden layer:

$$\delta^{(h)} = \delta^{(out)} (\mathbf{W}^{(out)})^T \odot \frac{\partial \phi(z^{(h)})}{\partial z^{(h)}}$$

[Raschka]



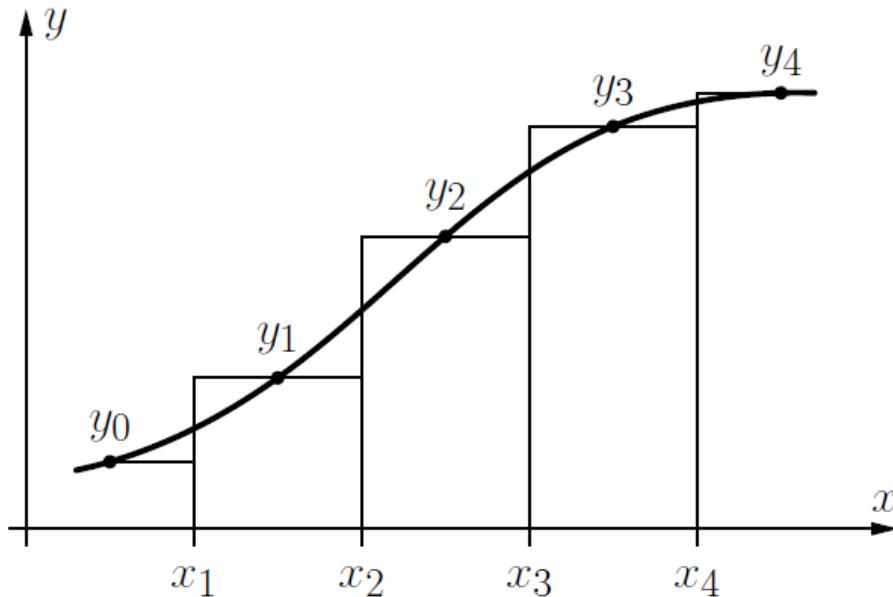
Training Data is very often stored in a data file.

1. Generate the network architecture: Number of Input neurons and neurons for the output, number of hidden layers and number of neurons for the hidden layers
2. Init of all weights and thresholds with random numbers
3. Put the training data to the net
4. Calculate outputs
5. Calculate error
6. Adjust weights and thresholds according to the Backpropagation Algorithm
7. Jump to 3. as long as error is larger than exit threshold

Neural Networks

Approximation of functions by MLP

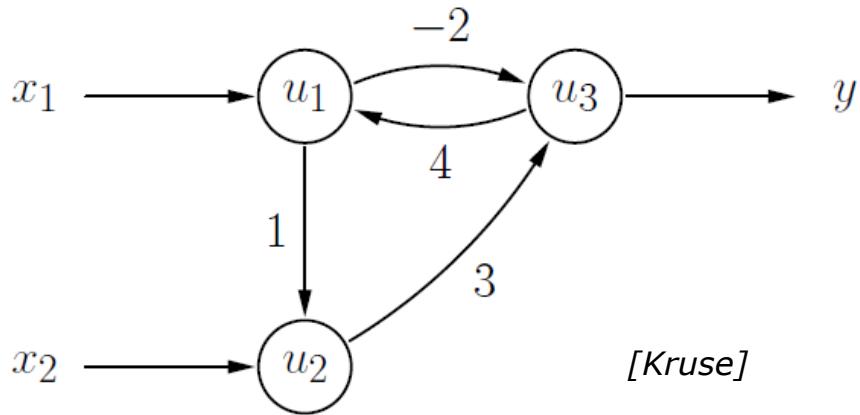
- > Approximate a given function by a step function
- > Design a MLP which calculates the step function
- > So every Riemann-integrable function can be approximated
- > It is mathematically provable that every steady function can be approximated with a three layer MLP



Neural Networks

Types of artificial Neural Networks

When the net uses parts which have a feedback (Cycles), then it belongs to the „**recurrent**“ or **feedback** Nets. Otherwise it is a „**Feedforward**“ Net.

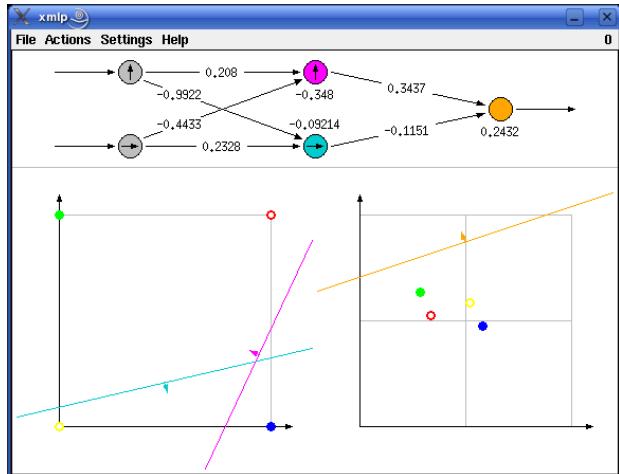


Recurrent Net

Neural Networks

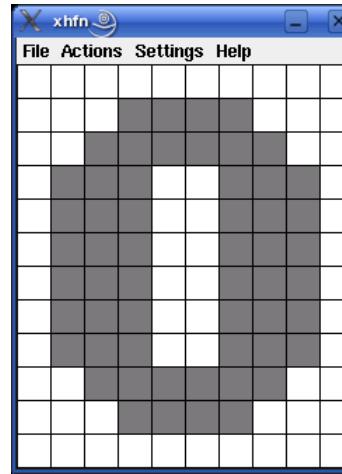
Examples

MLP Demo



<http://www.borgelt.net/doc/mlpd/mlpd.html>

Hopfield Demo



<http://www.borgelt.net/hfnd.html>



Pattern Classification
Iris Versicolor
With OpenNN

<http://opennn.cimne.com/download.asp>

Neural Networks

Applications



IN CS, IT CAN BE HARD TO EXPLAIN
THE DIFFERENCE BETWEEN THE EASY
AND THE VIRTUALLY IMPOSSIBLE.

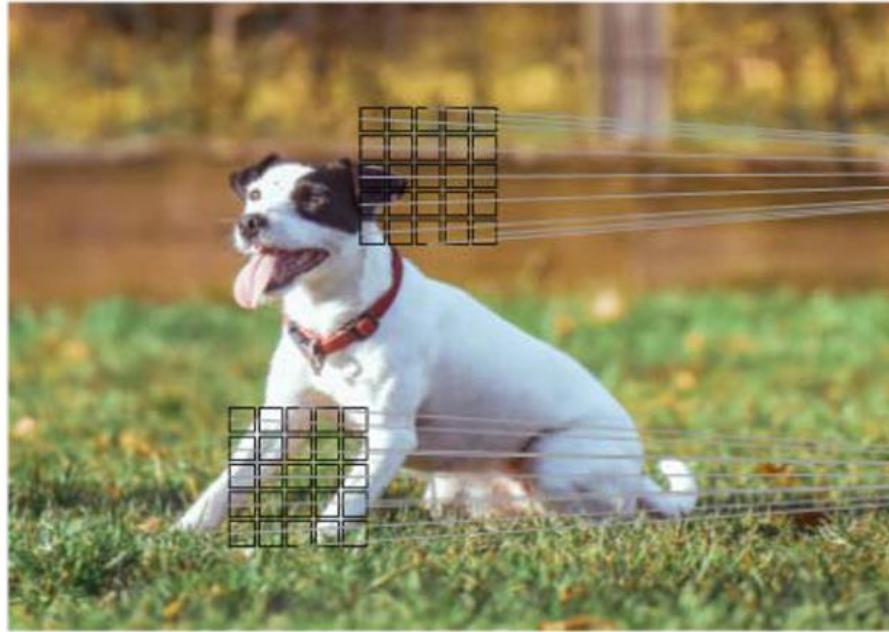
YOLO as a nice example

<https://pjreddie.com/darknet/yolo/>

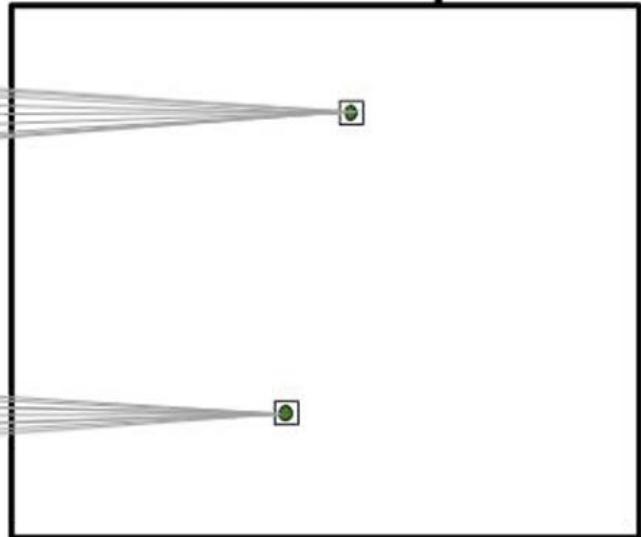
Neural Networks

Other Network Architectures III

Convolutional Neural Nets (CNN)



Feature map:

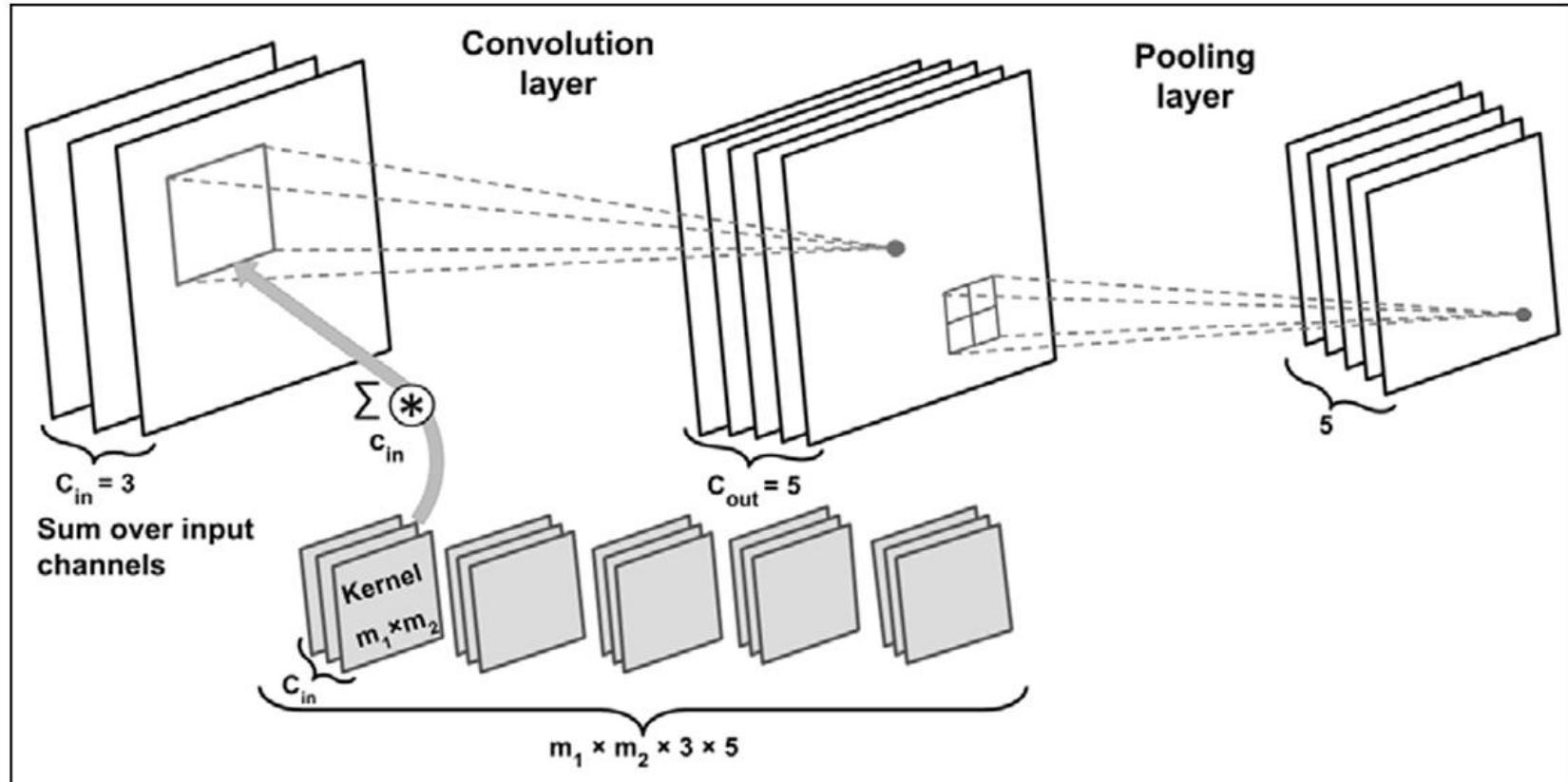


- > Inspired by the human visual Cortex
- > mostly used for image classification,
- > extracts features,
- > local receptive fields

Neural Networks

Other Network Architectures III

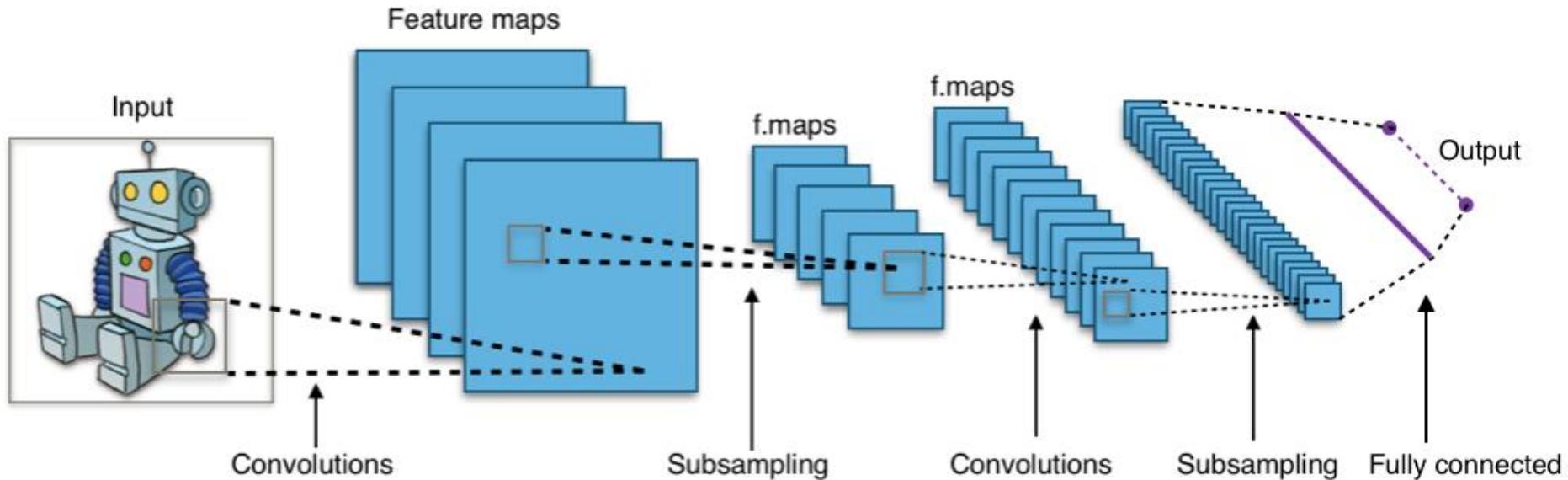
Building Blocks of CNNs: Convolutional and Pool Layers



Neural Networks

Other Network Architectures III

Building Blocks of CNNs: Convolutional and Pool Layers



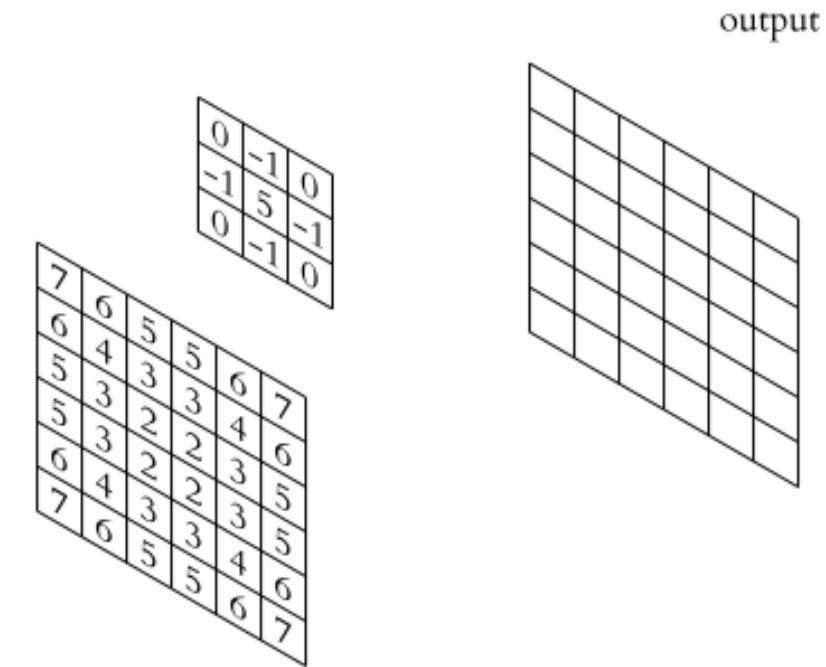
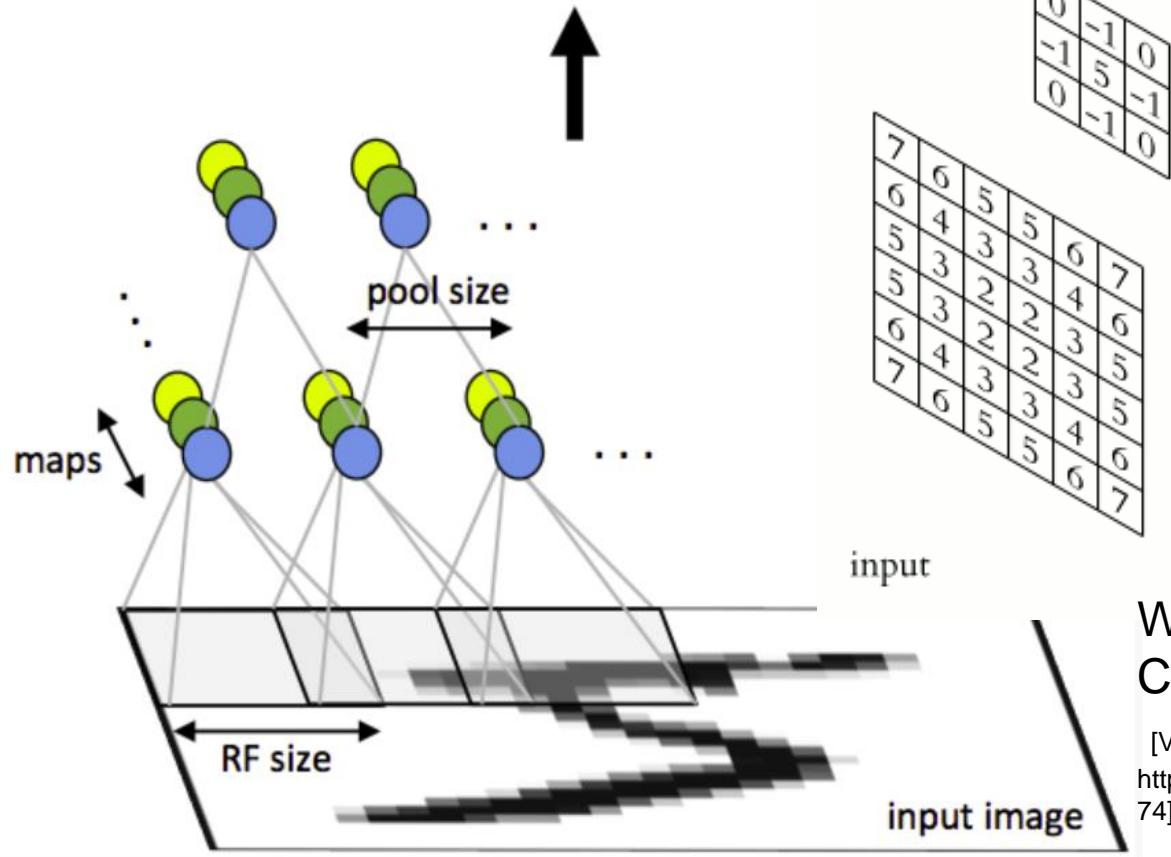
Typical setup of CNN

[Von Aphex34 - Eigenes Werk, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=45679374>]

Neural Networks

Other Network Architectures III

Convolutional and Pool Layers



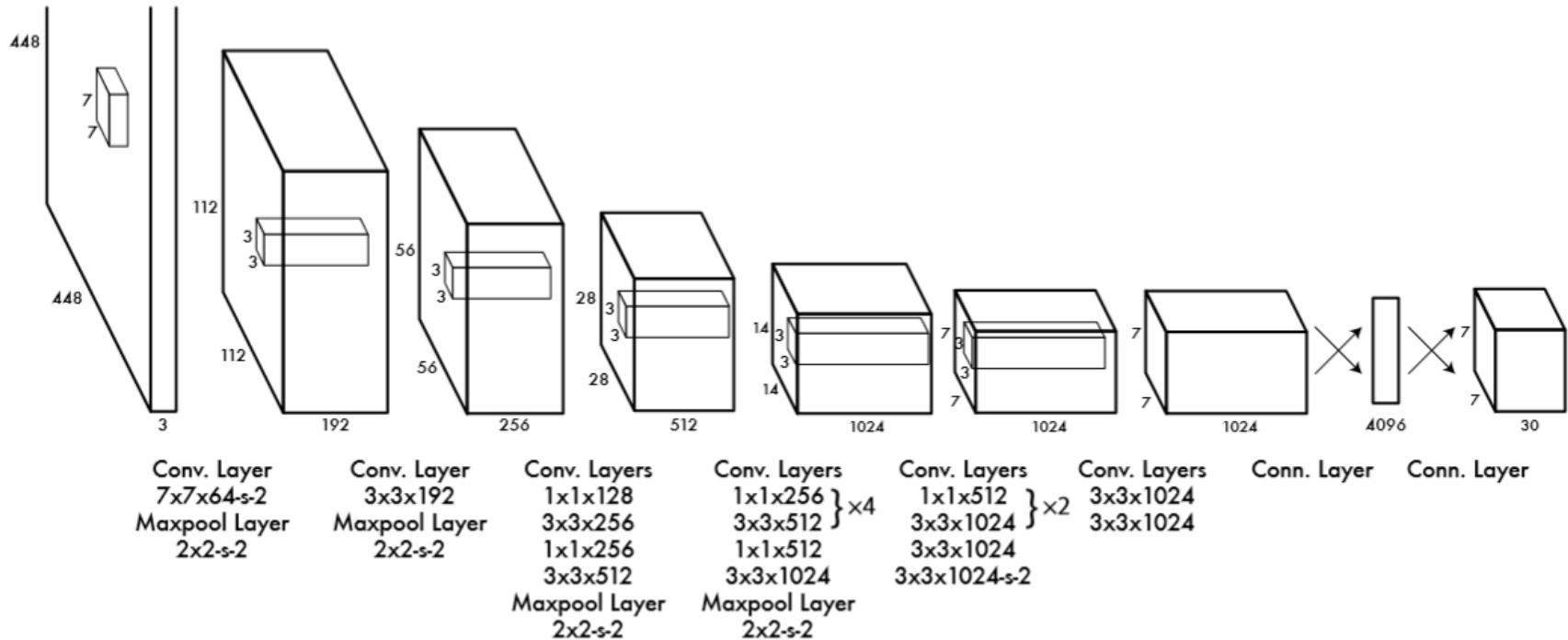
Working principle of
Convolutional Layer from

[Von Aphex34 - Eigenes Werk, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=45679374>]

Neural Networks

Other Network Architectures III

Example: YOLO net architecture



Neural Networks

Software



Open Source C++ Library, to statistically evaluate data sets for NN, to generate NN, train NN, and determine the performance. Quite fast and supports OpenMP and CUDA.



Commercial FrontEnd for OpenNN. Komfortable User Interface, Performance Indicator and Documentation.



OpenCV has an own Machine Learning Module, supporting MLP. Simple direct coupling to image processing possible



Neural Network library with Python interface for most of common architecture. Open Source! No more maintained. Last update 2010.

Neural Networks

Software, Frameworks, Books



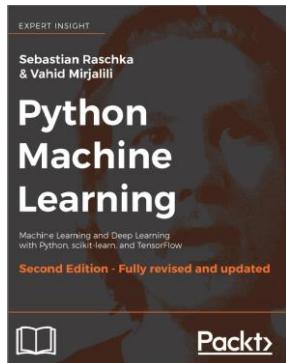
Open Source Framework for machine learning, very popular, pushed by Google, supports Deep convolutional networks, multiple CPUs and GPU. Available under all OS, including Windows, very vibrant!



Python machine learning framework with a lot of other algorithms beyond neural nets



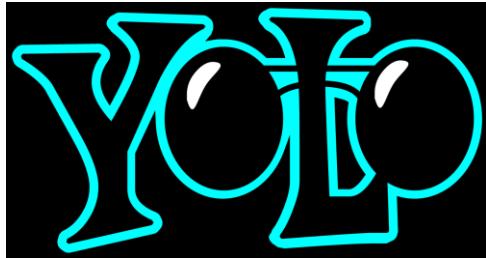
Very popular framework. Already quite difficult to install but powerful. CPU, GPU and Intel NCS support. For research...



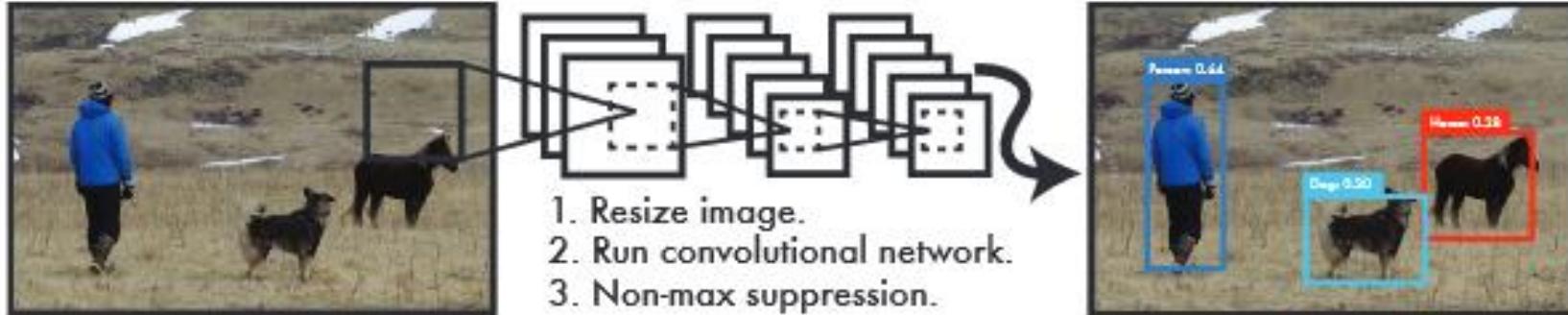
A nice book with a lot of practical examples about machine learning and enough theoretical background. Available as ebook for our course under:
<http://ebookcentral.proquest.com/lib/aachen/detail.action?docID=5050960>

Neural Networks

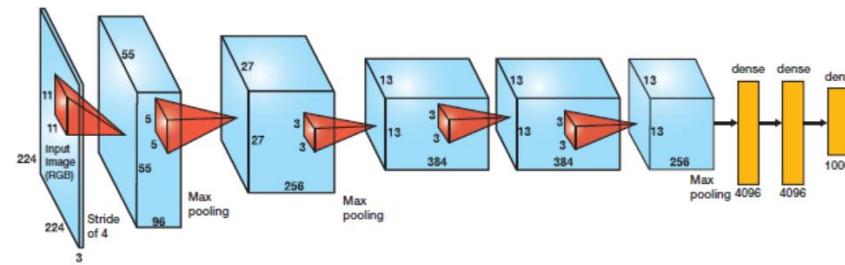
Ready to go Networks



Implementation of the YOLO net. Just compile and go. Trainable, fast and quite easy to use. Acceleration via CUDA and Intel NCS

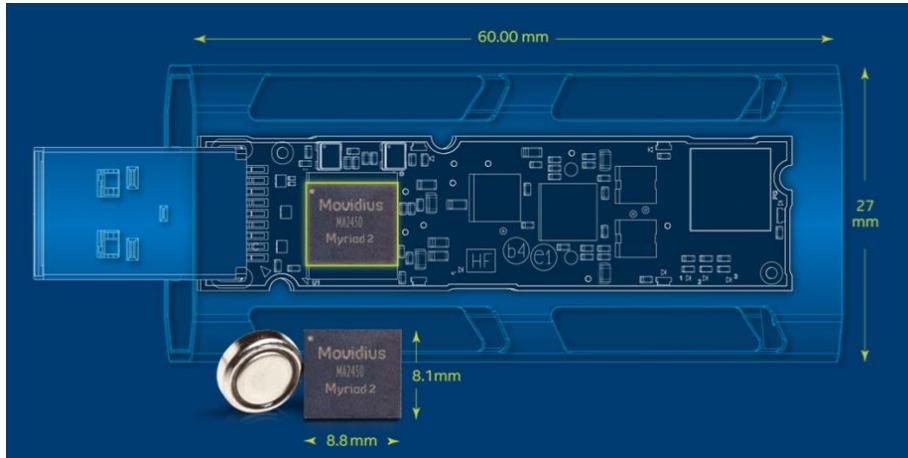


AlexNet, GoogLeNet,
Inception v1-3,
SqueezeDet, VGG16
and more to come



Neural Networks

Software, Ready to go Networks, Hardware



Hardware acceleration of Neural Nets with Intel NCS. The Movidius Myriad is a vector processor (VPU).



Neural Networks

Summary

Benefits

- > Generalization,
- > Fault tolerance,
- > Processing Speed,
- > Simple application by training,
- > Large choice of different types and
- > good to parallelize (FPGA, GPU, VPU, DSP).



Caffe



TensorFlow

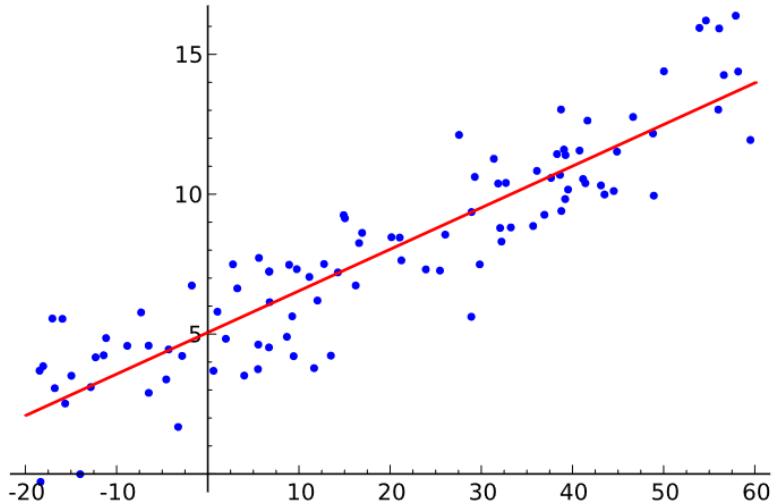


Drawback

- > Processing of the data for significant patterns can be difficult,
- > Sometimes Clustering and preprocessing of training data necessary
- > Black-Box behaviour,
- > Strategy to improve e.g. the classification results are often unclear

Weitere Klassifizierungsverfahren I

Lineare Regression



- > Einflussgröße X und Zielgröße Y
- > Es soll eine lineare Beziehung zwischen X und Y erzeugt werden, die möglichst „gut“ durch die Punktwolke (Merkmale) geht.

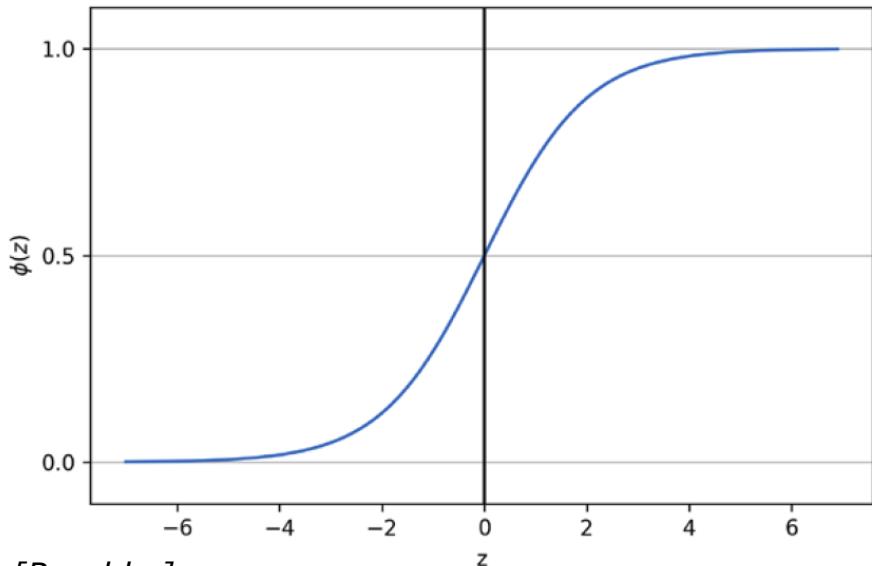
Von Sewaqu - Eigenes Werk, Gemeinfrei,
<https://commons.wikimedia.org/w/index.php?curid=11967659>

Nachteil

- > Viele Zusammenhänge sind nichtlinear oder hängen von mehreren Einflussgrößen ab.
- > Eingaben können schlecht skalierte Werte erzeugen.

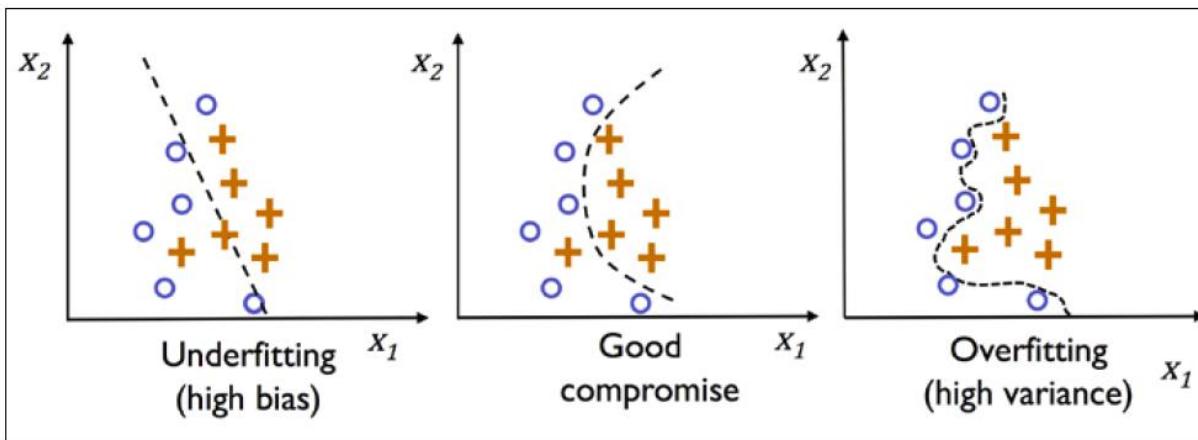
Weitere Klassifizierungsverfahren II

Logistische Regression



[Raschka]

- > Besitzt Wertebereich zwischen $[0,1]$, skaliert daher für alle Werte gut.
- > Ausgang der logit Funktion kann als Wahrscheinlichkeit des Ereignisses interpretiert werden.

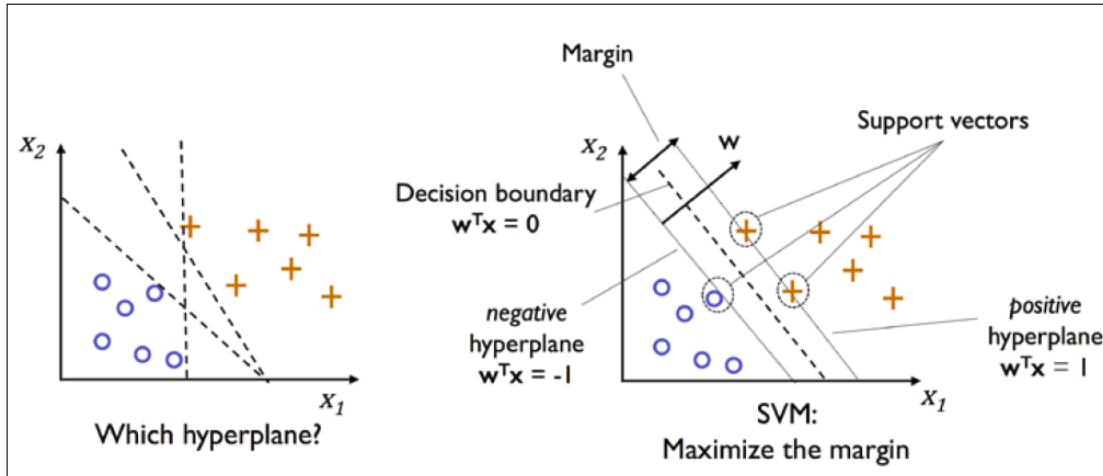


Nachteil

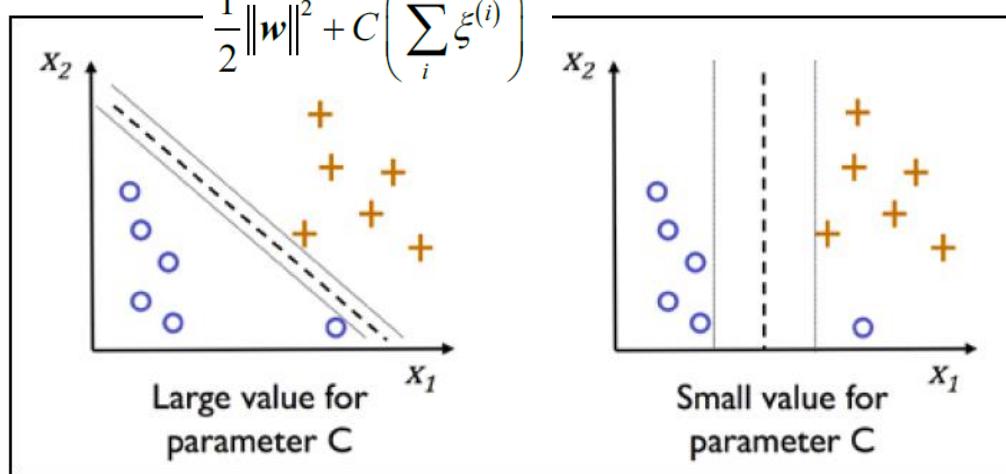
- > Möglichkeit der Überanpassung (Overfitting).

Weitere Klassifizierungsverfahren III

Support Vector Machine (Maximum Margin)



[Raschka]



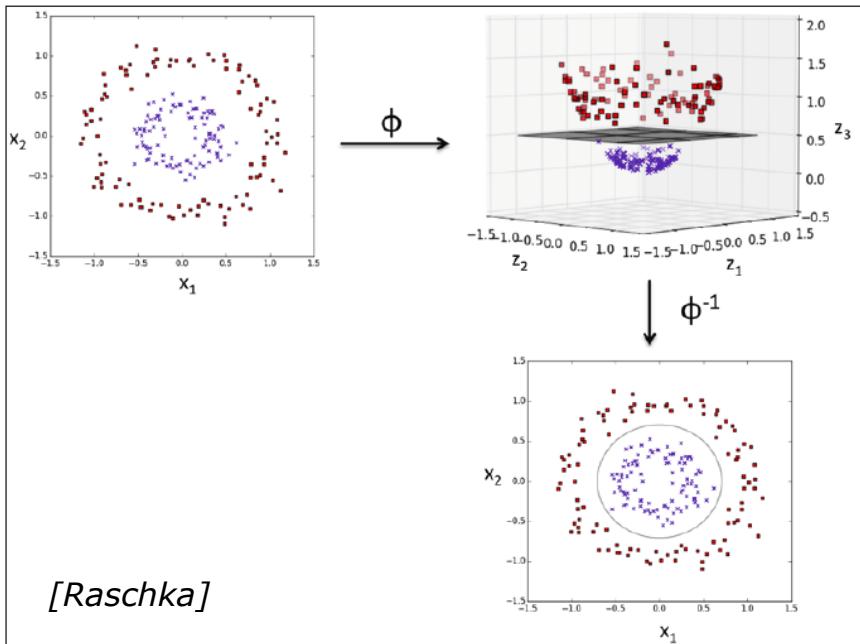
- > Erweiterung des Perzeptrons.
- > Möglichst breiter Rand (Margin) um die Klassengrenzen.
- > Trennende Hyperebene wird durch Stützvektoren definiert.
- > Bessere Verallgemeinerung als Perzeptron.
- > Steuerung der Margin durch Schlupfvariable ξ

Nachteil

- > Meist nur lineare Aufgaben lösbar.

Weitere Klassifizierungsverfahren IV

Support Vector Machine mit nichtlinearem Kernel



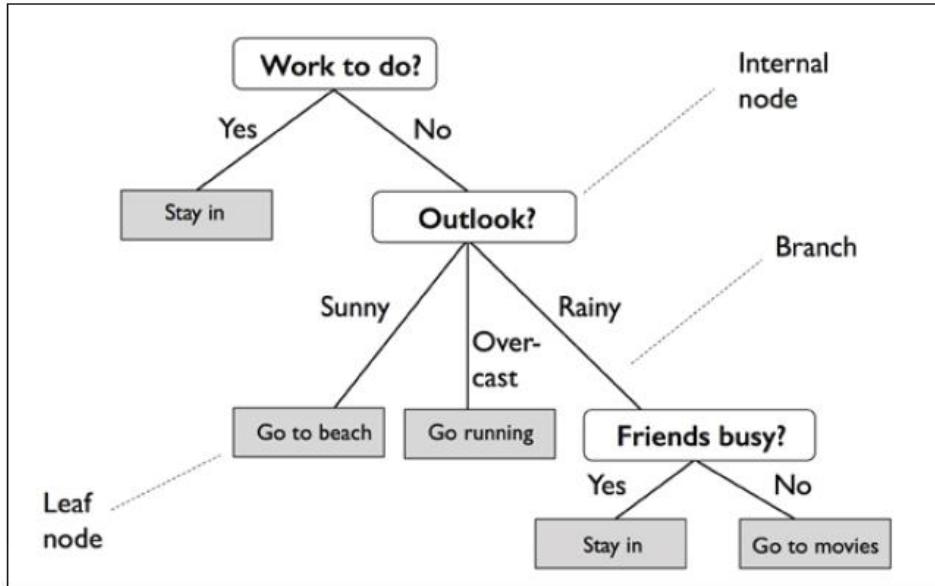
- > Erweiterung des SVM durch nichtlinearen Kernel.
- > Meist wird RBF (Radial Basis Function) eingesetzt.
- > Nichtlinear trennbare Merkmale werden durch eine Funktion Φ auf einen höherdimensionalen Raum abgebildet und anschließend linear getrennt.

Nachteil

- > Kann bei multidimensionalen Daten rechenaufwendig werden.

Weitere Klassifizierungsverfahren V

Lernen mit Entscheidungsbäumen



[Raschka]

- > Erlauben eine sehr gute Interpretierbarkeit des Modells.
- > Es werden anhand einer Reihe von Fragen Entscheidungen getroffen.
- > Klassenbezeichnungen werden erlernt.
- > Numerische Daten sind möglich.

Nachteil

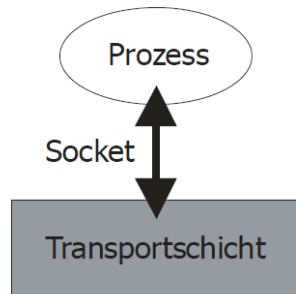
- > Kann leicht zu Überanpassung führen, wenn der Baum zu „tief“ ist.

Wie plaudert man mit anderen PCs?

Für die Kommunikation werden meist Sockets eingesetzt, die eine Schnittstelle zwischen dem Programm und dem Protokoll der Transportschicht darstellen. Mit Hilfe des Sockets funktioniert die Kommunikation mit dem Prozess des anderen Systems.

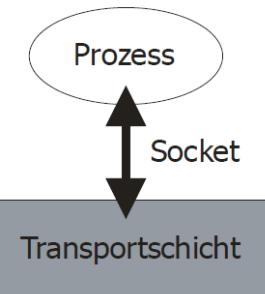
A: IP, Port

Rechner A:



B: IP, Port

Rechner B:

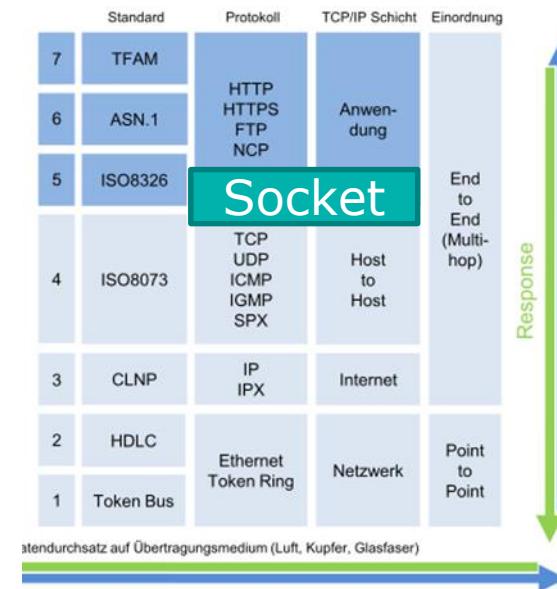


- > Zwei Rechner bzw. zwei Prozesse können kommunizieren, wenn die IP-Adresse des anderen Rechners und sein Port bekannt sind.

Nachrichten der
Transportschicht

[Michael Savorić,
Internet-Kommunikation
in Python mit Sockets]

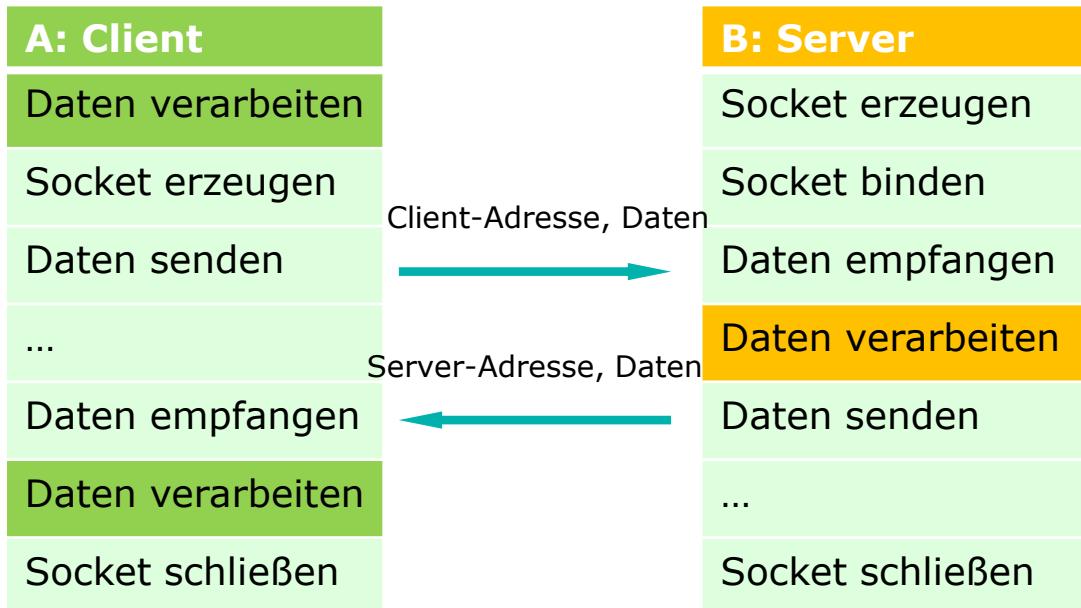
- > Der Socket verbindet die Transportschicht mit den höheren Schichten und abstrahiert die Komplexität der Kommunikation.



Wie plaudert man mit anderen PCs?

Es werden im Wesentlichen zwei Transportprotokolle unterschieden:
UDP (User Datagram Protocol) und TCP (Transmission Control Protocol).
Je nach Transportprotokoll ist die Kommunikation unterschiedlich.

UDP Kommunikation



[nach: Michael Savorić, Internet-Kommunikation in Python mit Sockets]

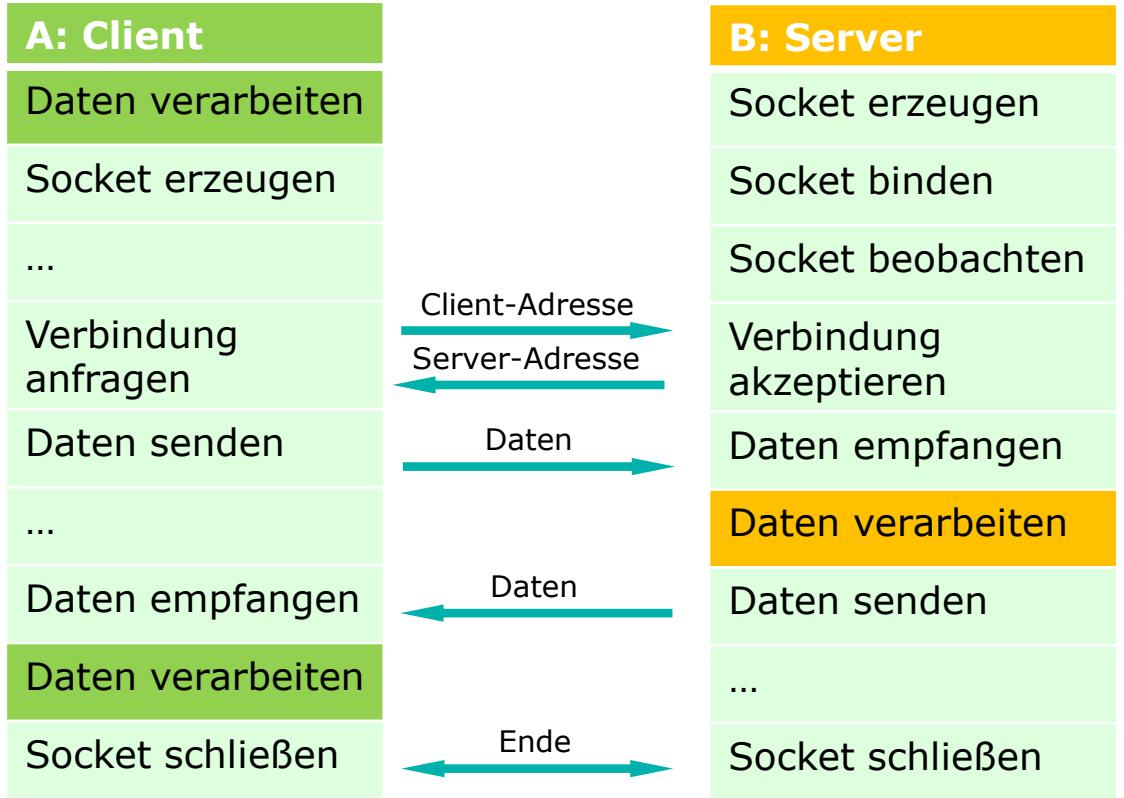
Socket binden: IP-Adresse und Port festlegen

- > Man unterscheidet die verbindungslose mit UDP, von der verbindungsorientierten TCP Kommunikation.
- > UDP wird benutzt, um eine „sofortige“ Datenübertragung von meist großen Datenmengen zu realisieren.
- > UDP hat keine (!) Fehlerkorrektur, so dass nicht empfangene Daten verloren gehen.
- > Typische Anwendungen sind Video- oder Audio Streams (VoIP) und Messdaten.

Wie plaudert man mit anderen PCs?

Die TCP Kommunikation stellt eine permanente Verbindung zwischen Client und Server her. Die Daten werden überprüft und Datenverlust während der Kommunikation festgestellt. Verlorene Daten werden erneut gesendet, so dass eine sichere Übertragung gewährleistet ist.

TCP Kommunikation



- > TCP besitzt eine eingebaute Fehlerkorrektur und Überlastkontrolle.
- > TCP wird benutzt, um eine „sichere“ Datenübertragung zu realisieren.
- > TCP baut eine Verbindung auf, überträgt die Daten und schließt am Ende den Socket.
- > Typische Anwendungen sind http oder ftp Anfragen, aber auch POP und SMTP (email).

[nach: Michael Savorić, Internet-Kommunikation in Python mit Sockets]

Netzwerke und Kommunikation

UDP Kommunikation Client



Das Python Modul `socket` ermöglicht sowohl den Aufbau einer UDP als auch einer TCP Verbindung. Beim Aufbau des Sockets wird sein Verhalten über seine Eigenschaften konfiguriert. Die UDP und TCP Kommunikation benutzt immer Byte Variablen, so dass alle Datentypen in Byte-Streams umgewandelt werden müssen!

```
from socket import *
server_addr = ("time.fu-berlin.de", 13)

client_socket = socket(AF_INET, SOCK_DGRAM)
client_socket.sendto("").encode(), server_addr)
daten, addr = client_socket.recvfrom(1024)
datenstring = daten.decode()
client_socket.close()
del client_socket
print(datenstring)
```

Abfrage des Date-Time Servers der FU Berlin

- > Die Server Adresse kann als Tuple mit dem Namen und dem Port des Servers generiert werden.
- > Als Kommunikation wird die Adress Family Internet (AF_INET) benutzt. Es existieren z.B. auch Möglichkeiten per Bluetooth oder CAN zu kommunizieren.
- > Für eine UDP Kommunikation wird der Typ SOCK_DGRAM gewählt.
- > Die zu sendenden Daten müssen als Byte mit `encode()` encodiert und die empfangenen mit `decode()` decodiert werden.

Die TCP Kommunikation ist bei der Programmierung etwas aufwendiger, besitzt aber bspw. den Vorteil der Fehlerkorrektur. Der Server wird mit dem TCP Protokoll initialisiert und wartet auf die Verbindung mit dem Client `listen()`. Wenn der Client verbunden ist, werden mit `recv()` die Daten empfangen und mit `send()` Daten über die Verbindung gesendet.

```
import socket

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 20
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

conn, addr = s.accept()
print('Connection address:', addr)
while 1:
    data = conn.recv(BUFFER_SIZE)
    if not data: break
    print("received data:", data)
    conn.send(data) # echo
conn.close()
```

- > Als Kommunikation wird wieder Adress Family Internet (AF_INET) benutzt.
- > Für die TCP Kommunikation wird der Typ SOCK_STREAM gewählt.
- > Die zu sendenden Daten müssen ebenfalls enkodiert und dekodiert werden.

Einfacher TCP Server mit echo Funktion

Der TCP Client kann relativ einfach aufgebaut werden. Wenn der TCP Server gestartet ist und den Socket zur Verfügung stellt, kann sich der Client mit `connect()` zum Server verbinden. Anschließend können die Daten vom Client gesendet `send()` und auch Daten empfangen werden `recv()`.

```
import socket

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
MESSAGE = "Hello, World!"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE)
data = s.recv(BUFFER_SIZE)
s.close()

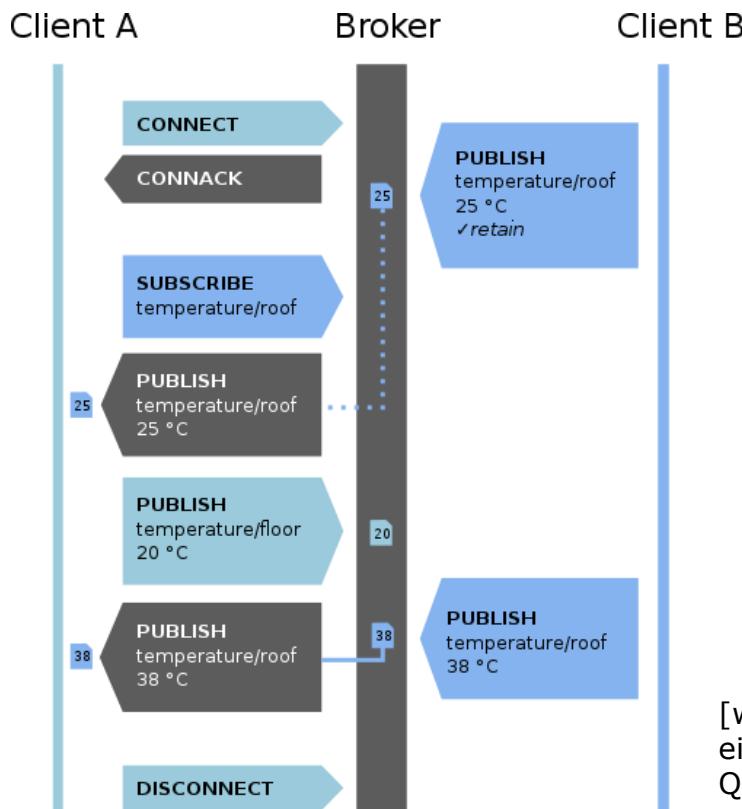
print("received data:", data)
```

Einfacher TCP Client, der nach dem Senden auch eine Nachricht empfängt.

Netzwerke und Kommunikation

MQTT Kommunikation

Sollen Daten zwischen „Maschinen“ übertragen oder gesammelt werden, bietet sich das Message Queuing Telemetry Transport (MQTT) Protokoll an. Es ist ein „machine-to-machine“ (M2M) Protokoll, welches für Telemetriedaten - also Übertragung von Messwerten - eingesetzt wird.



- > MQTT ist ein Client-Server (Broker) Protokoll, dass auf einer Publisher/Subscriber-Architektur basiert.
- > Der MQTT-Broker hält dabei den momentanen Zustand der Kommunikation mit den Clients als eine Art Datenbank fest. So können auch „unperformante“ Systeme wie Microcontroller Daten senden und empfangen.
- > MQTT benutzt Port 1883 und 8883.

[wiki: mqtt, Paketaustausch einer MQTT-Verbindung mit QoS = 0]

Der Austausch von Daten geschieht über „Topics“. Eine Nachricht besteht immer aus der Nachricht und dem Topic, welches hierarchisch eingestuft wird, z.B. car/1/temperature/roof, car/2/temperature/engine. Clients können diese Topics abonnieren. Die Topics werden dann vom Broker entsprechend weitergeleitet. Für python bietet sich paho-mqtt 1.5.0 an: <https://pypi.org/project/paho-mqtt>

```
# Einfacher mqtt Client als Subscriber
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("$SYS/#")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("mqtt.eclipse.org", 1883, 60)

client.loop_forever()
```

Aufbau eines mqtt Clients

- > Zuerst muss das Modul paho.mqtt.client importiert werden. Dann kann eine Verbindung zu einem mqtt Broker auf gebaut werden.
- > Es muss eine Callback Funktion on_connect() erzeugt werden, die bei erfolgreicher Verbindung aufgerufen wird und das topic festlegt.
- > Ein weiterer Callback on_message() wird beim Empfang einer Nachricht aufgerufen.

Netzwerke und Kommunikation

MQTT Kommunikation



Meist wird MQTT mit TCP zusammen benutzt. Das Fehlerverhalten bei einem Verbindungsabbruch kann mit dem Quality-of-Service (QoS) konfiguriert werden. Als Broker steht Eclipse Mosquitto™ als Open Source Software zur Verfügung (<https://mosquitto.org/>). Zum Test des eigenen mqtt Clients kann mqtt.eclipse.org benutzt werden.

```
# Einfacher mqtt Client als Publisher
import paho.mqtt.client as mqtt
import time

mqtt_server="mqtt.eclipse.org"
mqtt_port=1883
topic = "/text/fromyoga"

# Definition von on_connect und on_log weggelassen!

client = mqtt.Client()
client.on_connect = on_connect
client.on_log = on_log
client.connect(mqtt_server, mqtt_port, 60)

numberOfHellos = 1

while(True):
    stringToPublish = "Hello MQTT "+ str(numberOfHellos)
    client.publish(topic,stringToPublish)
    numberOfHellos += 1
    time.sleep(5)
```

Der Quality-of-Service (QoS) einer Nachricht bestimmt die Art der Verbindungsgüte:

- > At most once: die Nachricht wird einmal gesendet, kommt aber evtl. bei Verbindungsunterbrechung nicht an.
- > At least once: die Nachricht wird solange gesendet, bis der Empfänger den Empfang bestätigt.
- > Exactly once: auch bei Verbindungsunterbrechung wird sichergestellt, dass die Nachricht ankommt.
QoS kann bei publish() angegeben werden.

Netzwerke und Kommunikation

MQTT Kommunikation



Die zu lesenden Topics können auch mit Wildcards angegeben werden, so dass z.B. alle Temperaturen von car1 abonniert werden können mit dem topic: car/1/temperature/# Die Nachrichten können über TLS/SSL verschlüsselt werden.

```
import paho.mqtt.client as mqtt

mqtt_server=" mqtt.eclipse.org"
mqtt_port=1883
topic = "/time/#"

def on_connect(mqttc, obj, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe(topic)

def on_log(mqttc, obj, level, string):
    print(string)
    print('Connecting to MQTT broker')

def on_message(client, userdata, msg):
    timeReceived=str(msg.payload.decode('utf8'))
    print("Data Received:",timeReceived)

client = mqtt.Client()
client.on_message = on_message
client.on_connect = on_connect
client.on_log = on_log
client.connect(mqtt_server, mqtt_port, 60)
client.loop_forever()
```

- > Nebenstehender Client subscribed zu allen Topics, die sich auf dem mqtt.eclipse.org Server befinden und mit /time/ beginnen.
- > Für Smartphones existieren diverse mqtt Clients, wo zu den entsprechenden Topics subscribed werden kann.
- > Gerade im Bereich SmartHome wird die mqtt-Kommunikation eingesetzt, aber auch im IoT Bereich, also Industrie 4.0

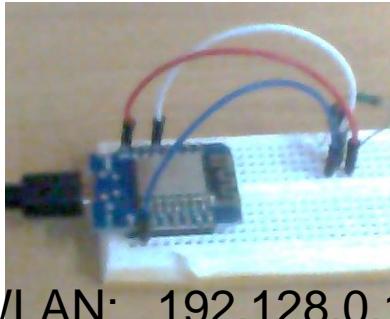
PT9 Vorbereitung

Vom Sensor in die Datenbank oder Cloud



COM (USB)

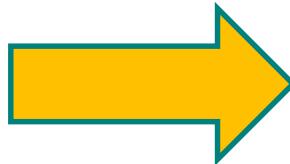
LAN: 149.201.213.219



WLAN: 192.128.0.105

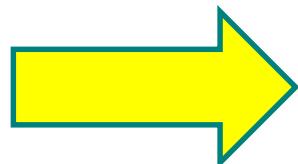
Temperatur und
Feuchtigkeitssensor

TCP/IP über LAN



149.201.213.219
Schicht 1: Kabel

MQTT über WiFi



192.128.0.105
Schicht 1: 2.4GHz



MQTT Broker und Gateway

WLAN: 192.128.0.102

LAN: 149.201.213.211

Datenkompression

Einleitung

Kompression zur **Verringerung**

- > des benötigten Speicherbedarfs und
- > der benötigten Bandbreite

Unterteilung in **verlustfreie** (echte) und **verlustbehaftete** Kompression.

Bei der **verlustfreien** Kompression können die Originaldaten aus den komprimierten Daten exakt wiederhergestellt werden (Redundanzdaten), z.B. Wiederholungen von Buchstaben, Endungen und Mustern.

Die **verlustbehaftete** Kompression reduziert die Daten indem „irrelevante“ Daten gelöscht werden, z.B. bestimmte Frequenzanteile bei Audiodaten (Psychoakustik).

Datenkompression

Beispiele

Beispiele verlustfreier Kompression:

- > Bild in Text konvertieren, Optical Character Recognition (OCR),
- > Huffman Kodierung, LZW-Algorithmus (ZIP), Lauflängenkodierung
- > Vorhandene Textfragmente (Token) erkennen.

Lauflänge:

Ausgangstext:	ppqqqppppqqqqqqqqqqq	(18 Zeichen)
Kodierter Text:	3p2q3p4q5p1q	(12 Zeichen)

Token:

Ausgangstext:	AUCH EIN KLEINER BEITRAG IST EIN BEITRAG
Kodierter Text:	AUCH EIN KLEINER BEITRAG IST -4 -3 [aus: wikipedia]

Datenkompression

Beispiele

Lempel-Ziv-Welch (LZW)-Algorithmus:

- > Häufig vorkommende Zeichenketten werden in Wörterbüchern gespeichert,
- > diese werden dann mit einer Abkürzung angesprochen und
- > das Wörterbuch (12 Bit) wird als Index mit in die Datei geschrieben.

```
initialisiere Mustertabelle mit (<leeres Muster>+zeichen) für alle Zeichen
muster := <leeres Muster>
solange noch Zeichen verfügbar
    zeichen := lies nächstes Zeichen
    wenn (muster+zeichen) in Mustertabelle dann
        muster := (muster+zeichen)
    sonst
        füge (muster+zeichen) zur Mustertabelle hinzu
        Ausgabe muster
        muster := zeichen
    wenn muster nicht <leeres Muster> dann
        Ausgabe muster
```

LZW Pseudocode [aus: Wiki]

Datenkompression

Beispiele

Ein Beispiel mit der Zeichenkette „LZW LZ78 LZ77 LZCLZM WLZAP“

Zeichenkette	gefundener Eintrag	Ausgabe	neuer Eintrag
LZW LZ78 LZ77 LZCLZM WLZAP	L	L	LZ (wird zu <256>)
ZWLZ78 LZ77 LZCLZM WLZAP	Z	Z	ZW (wird zu <257>)
WLZ78 LZ77 LZCLZM WLZAP	W	W	WL (wird zu <258>)
LZ78 LZ77 LZCLZM WLZAP	LZ (= <256>)	<256>	LZ7 (wird zu <259>)
78 LZ77 LZCLZM WLZAP	7	7	78 (wird zu <260>)
8 LZ77 LZCLZM WLZAP	8	8	8L (wird zu <261>)
LZ77 LZCLZM WLZAP	LZ7 (= <259>)	<259>	LZ77 (wird zu <262>)
7 LZCLZM WLZAP	7	7	7L (wird zu <263>)
LZCLZM WLZAP	LZ (= <256>)	<256>	LZC (wird zu <264>)
CLZM WLZAP	C	C	CL (wird zu <265>)
LZM WLZAP	LZ (= <256>)	<256>	LZM (wird zu <266>)
M WLZAP	M	M	MW (wird zu <267>)
WLZAP	WL (= <258>)	<258>	WLZ (wird zu <268>)
ZAP	Z	Z	ZA (wird zu <269>)
AP	A	A	AP (wird zu <270>)
P	P	P	-

Ausgangstext:

LZW LZ78 LZ77 LZCLZM WLZAP

(22 Zeichen)

Kodierter Text:

LZW <256> 78 <259> 7 <256> C <256> M <258> ZAP

(16 Zeichen)

Datenkompression

Beispiele

Beispiele verlustbehaftete Kompression:

- > MP3, JPG, MPG
- > Reduzierung der nichthörbaren Frequenzbereiche,
- > Farb reduzierung, Reduzierung des Detailreichtums, usw.
- > Algorithmen basieren z.B. auf: Tiefpassfilterung, speziellen Transformation, Huffman- oder arithmetische Kodierung.

Folgende Schritte werden bei der JPG Kompression durchgeführt:

1. Umrechnung des Farbraumes: Die RGB Daten werden in das YCbCr Farbmodell umgerechnet. Die Ortsauflösung des Auges ist deutlich unempfindlicher für Farb- als für Helligkeitsunterschiede.

$$\begin{bmatrix} Y' \\ Pb \\ Pr \end{bmatrix} \approx \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ -0,168736 & -0,331264 & 0,5 \\ 0,5 & -0,418688 & -0,081312 \end{bmatrix} \cdot \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \quad \text{„Farbunterabtastung“}$$

Datenkompression

Beispiele

2. Tiefpassfilterung der unterabgetasteten Chrominanzkanäle und vertikale und horizontale Unterabtastung der Cr und Cb Werte, z.B. nur für jedes 2. Pixel. RGB: 3x8Bit YCbCr: 8+2+2Bit
3. Blockbildung und Cosinustransformation (DCT)
Aufteilung des Bildes in 8x8 Blöcke. Zweidimensionale DCT

$$F_{xy} = \frac{1}{4} C_x C_y \sum_{m=0}^7 \sum_{n=0}^7 f_{mn} \cos \left[\frac{(2m+1)x\pi}{16} \right] \cos \left[\frac{(2n+1)y\pi}{16} \right] \quad C_x, C_y = \begin{cases} \frac{1}{\sqrt{2}} & \text{wenn } x, y = 0 \\ 1 & \text{sonst} \end{cases}$$

4. Quantisierung

Die Koeffizienten der DCT werden durch eine Quantisierungsmatrix geteilt und gerundet. Die Matrix ist

$$F^Q(x,y) = \left\lfloor \frac{F(x,y)}{Q(x,y)} \right\rfloor$$

Huffmann Kodierung I

Präfixfreier Bottom-Up Code (Graph oder Baum)

Als verlustfreie Komprimierung wird häufig der Huffman Code eingesetzt.

- > Präfixfrei, d.h. der Beginn jedes Codewortes ist eindeutig,
- > Keine Trennzeichen notwendig und
- > optimal bzgl. Entropie

Er gliedert sich in:

- > Aufbau des Baumes (k-närer Baum),
- > Konstruktion des Codeworts und
- > Kodierung.

Pseudocode zum Aufbau des Baums

1. Ermittlung der relativen Häufigkeiten p für das Quellsymbol
2. Jedes Quellsymbol ist ein Knoten mit einer Häufigkeit
3. Solange bis Baum komplett:
 - Wähle m Äste mit geringstem p
 - Fasse sie zusammen zu einem neuen Ast
 - Summe der einzelnen p kommt an den neuen Ast

Huffmann Kodierung II

Beispiel

Pseudocode zur Erstellung des Codebuchs

1. Ordne jedem Ast (Kante) des Baumes ein Codezeichen zu
2. Auslesen des Codes für jedes Quellsymbol
3. Solange bis Baum komplett:
 - Start an der Wurzel
 - Die Codezeichen an den Kanten ergeben das Codewort

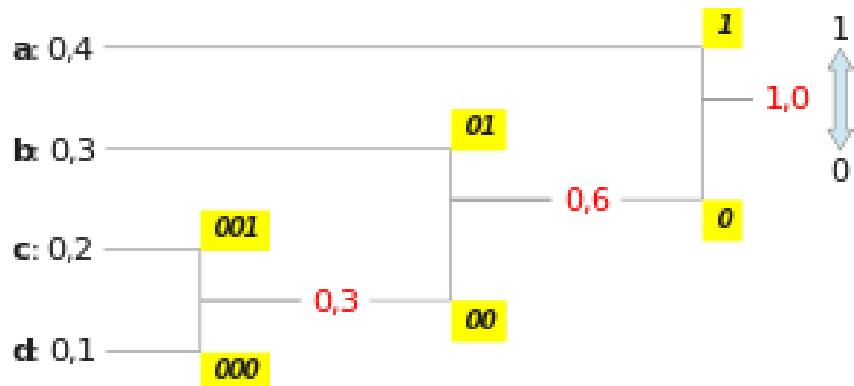
Pseudocode zur Codierung

1. Einlesen Quellsymbol
2. Ermittlung des Codeworts aus dem Codebuch
3. Ausgabe des Codewort

Generierter Baum für den Text:

„a ab abc abcd“

[wiki]



Huffmann Kodierung IV

Berechnung der mittleren Codelänge

Beispiel aus Wikipedia für die Codierung des Textes: „a ab abc abcd“ mit dem Binärcode

Ausgabe:

a	a	b	a	b	c	a	b	c	d
1	1	01	1	01	001	1	01	001	000

Die mittlere Codewortlänge kann aus den Einzelwahrscheinlichkeiten des Baums berechnet werden nach:

Es ergibt sich: $1,0+0,6+0,3=1,9$ Bit

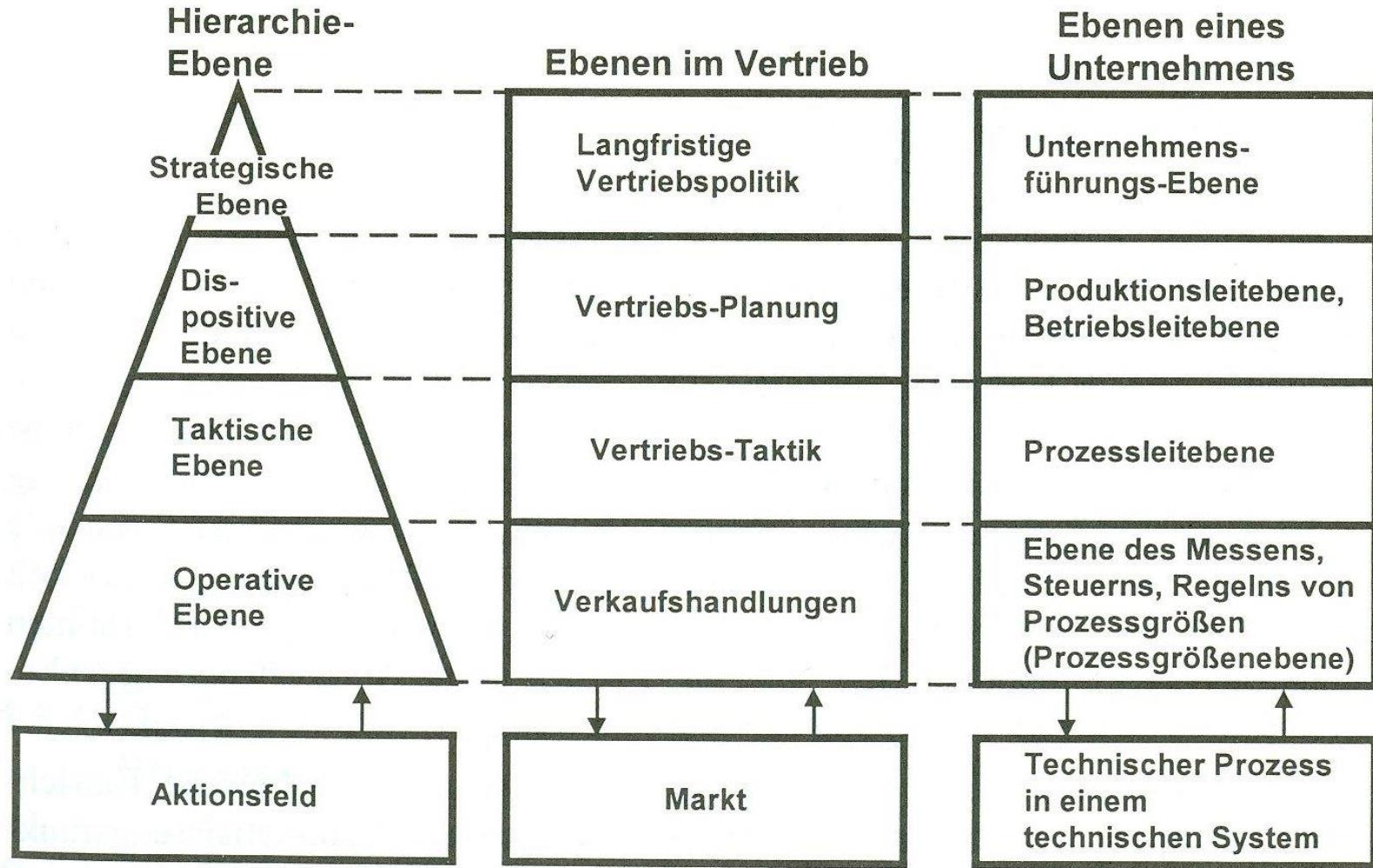
Bei „normaler“ Binärkodierung für vier Zeichen würden sich 2 Bit ergeben.

$$\bar{l} = \sum_{x \in X} p_x l_x$$

$$\bar{l} = \log_2 m$$

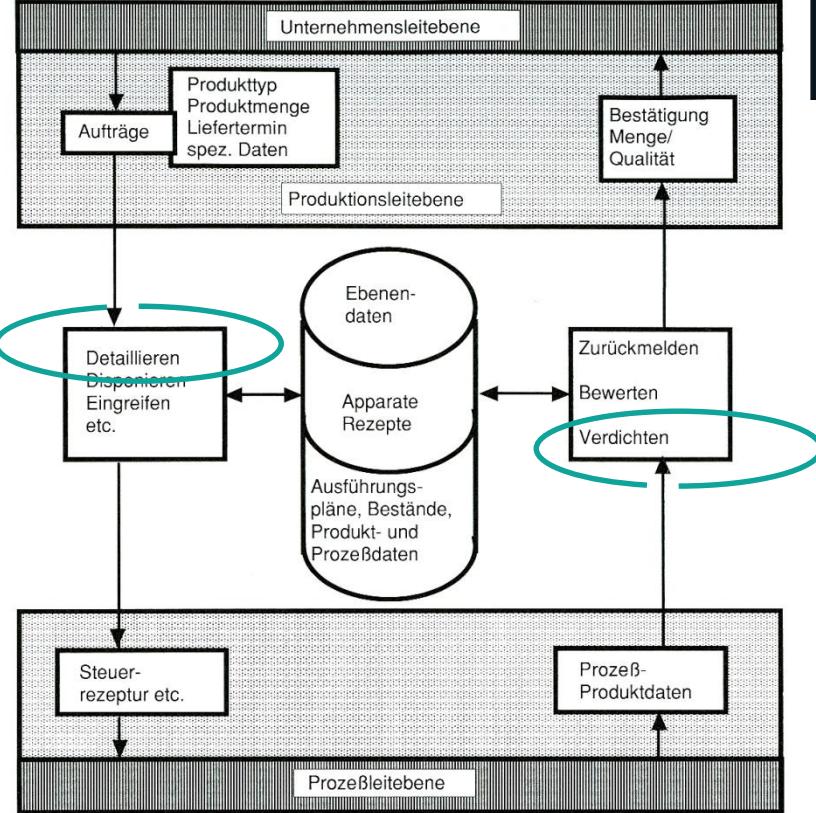
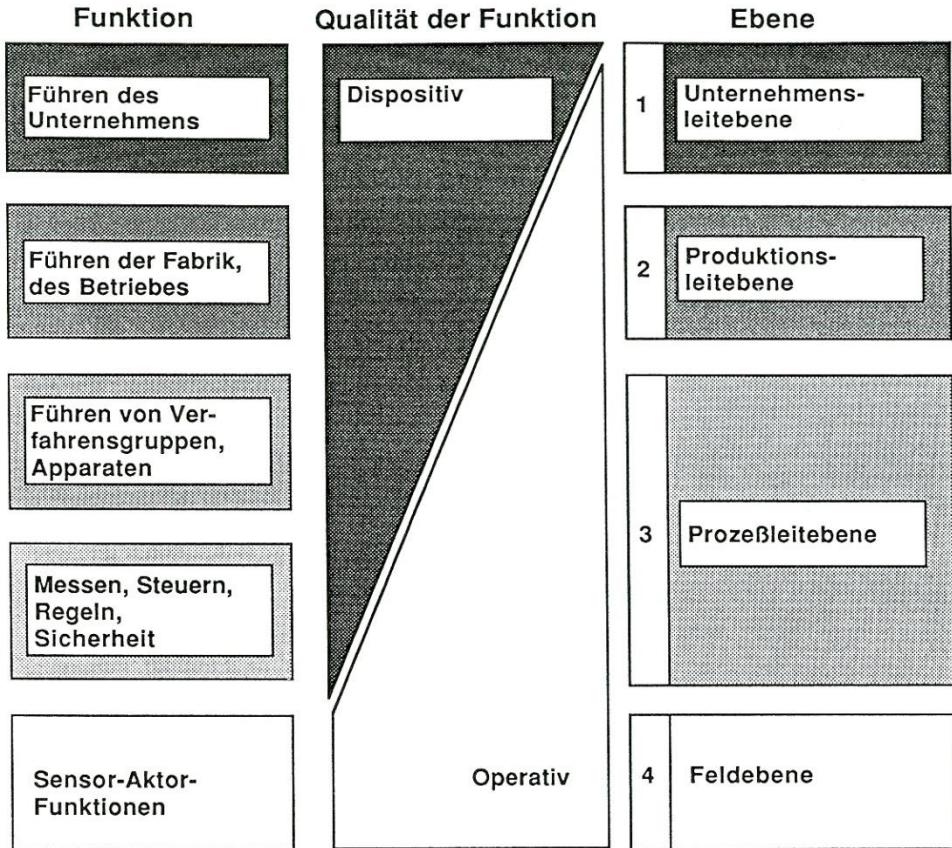
Informationsflüsse im Betrieb

Ebenenmodell



Informationsflüsse im Betrieb

Ebenenmodell

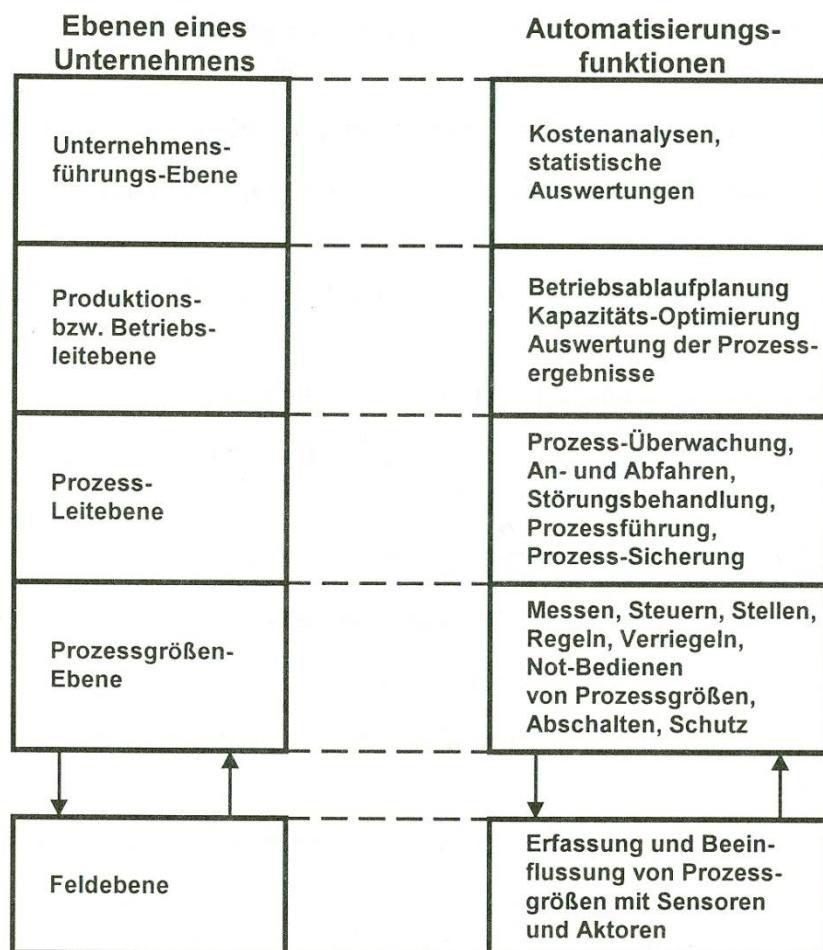


Operativ (unmittelbar wirksam) ->
Dispositiv (anordnend)

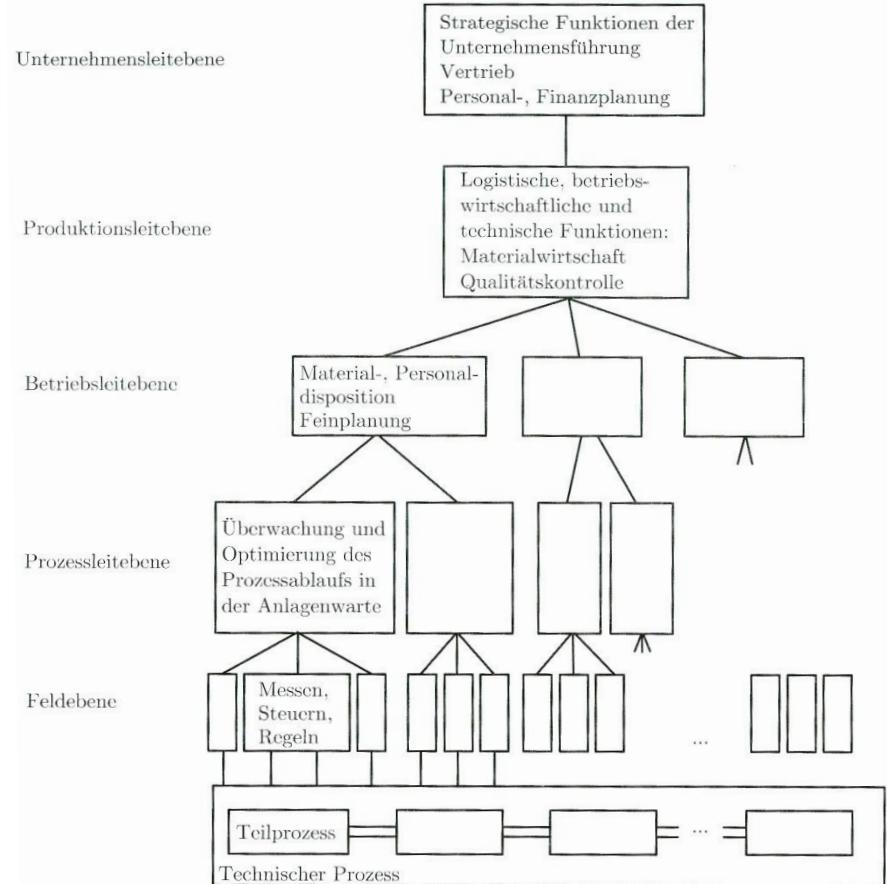
Verdichten vs. Detaillieren
von Information

Informationsflüsse im Betrieb

Ebenenmodell



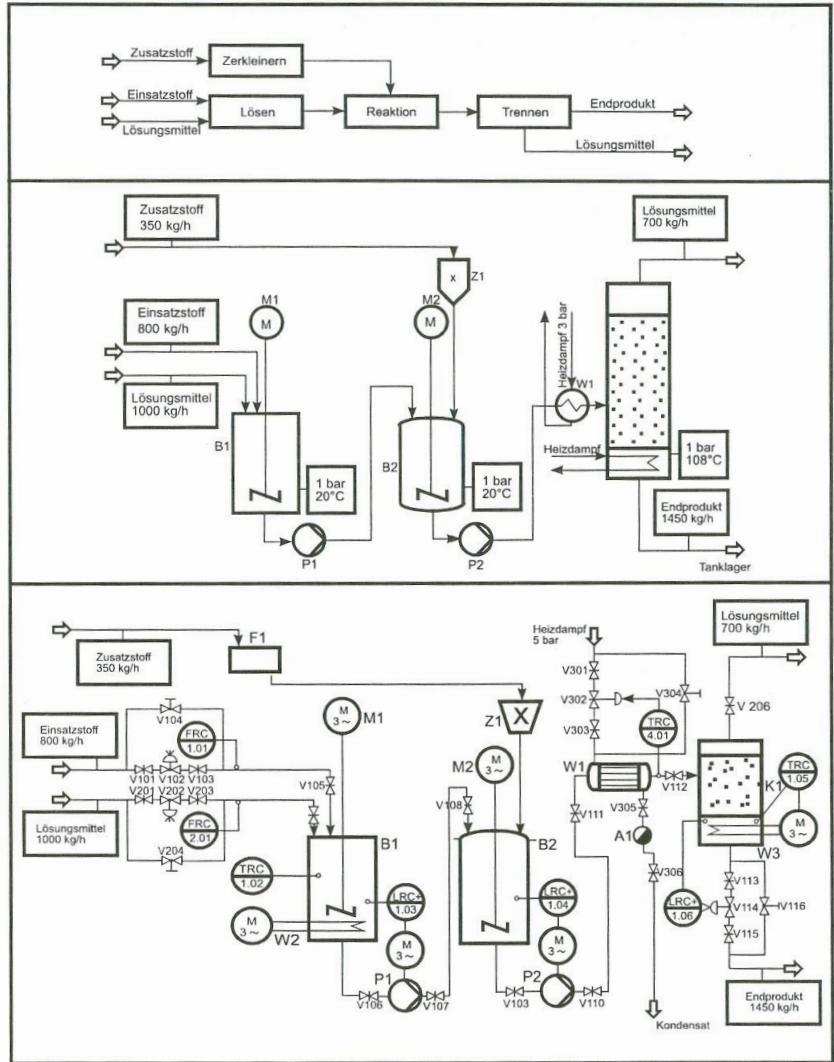
Ebenen u. Funktionen (Polke)



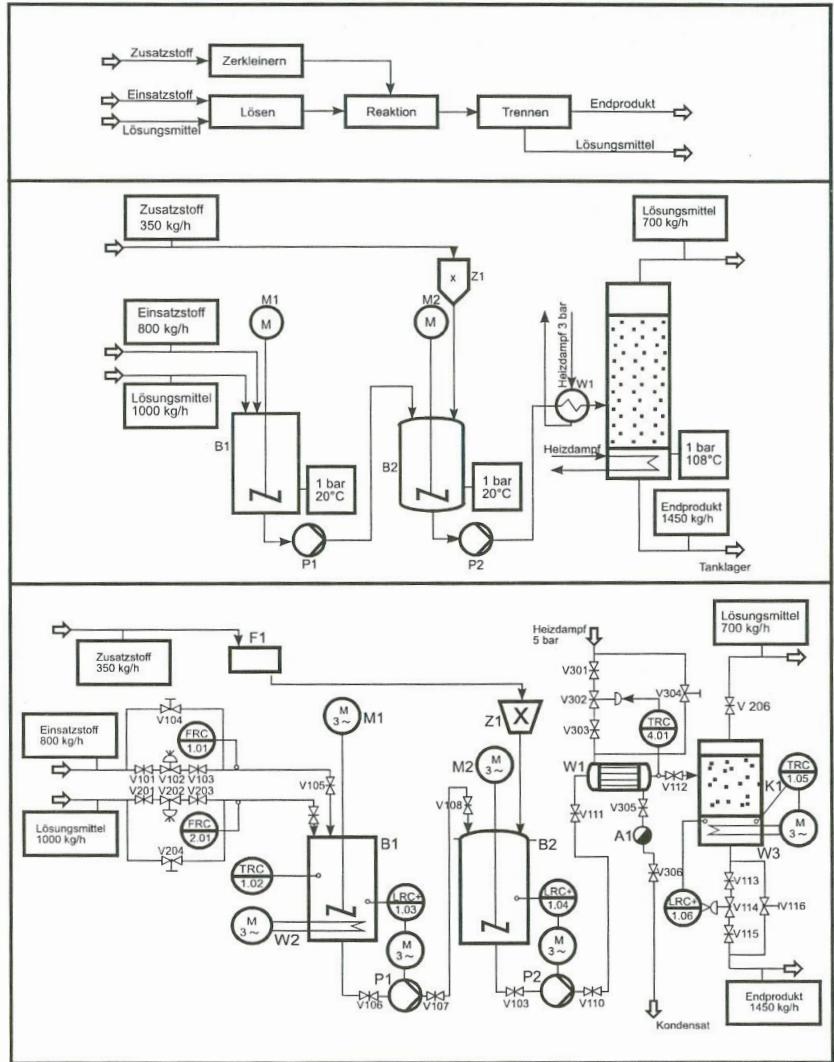
Ebenen u. Funktionen (Lunze)

Informationsflüsse im Betrieb

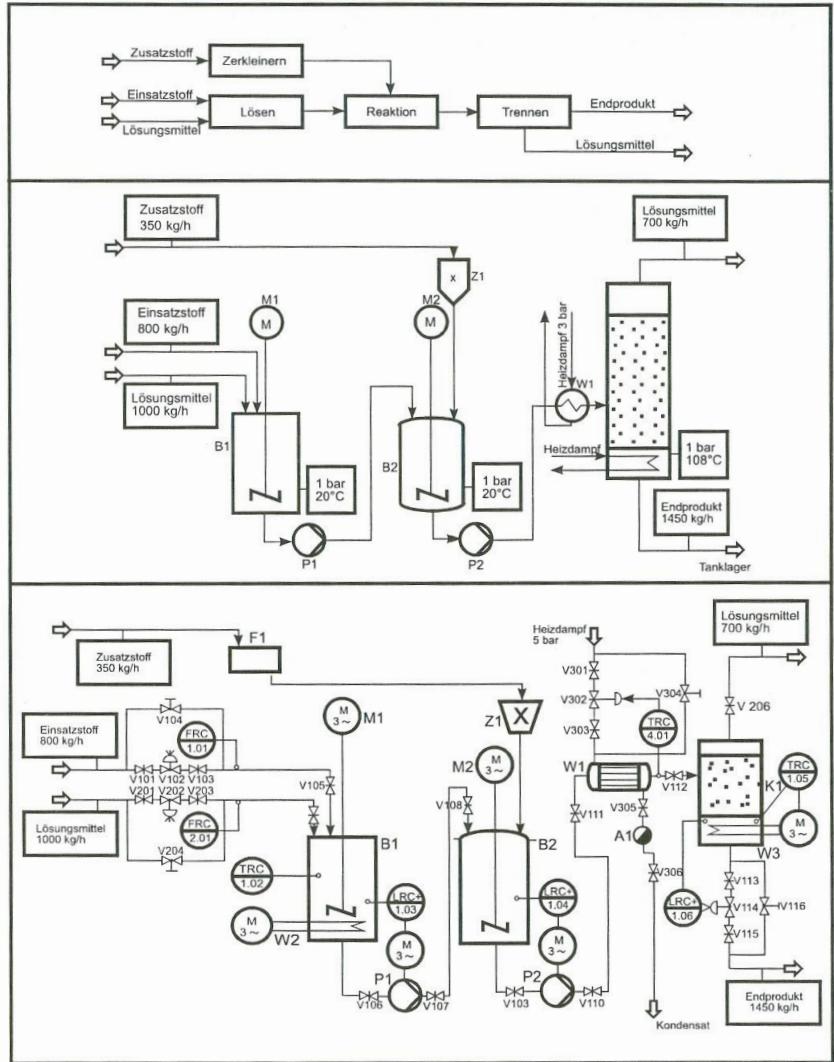
Phasenmodell



Grundfließbild



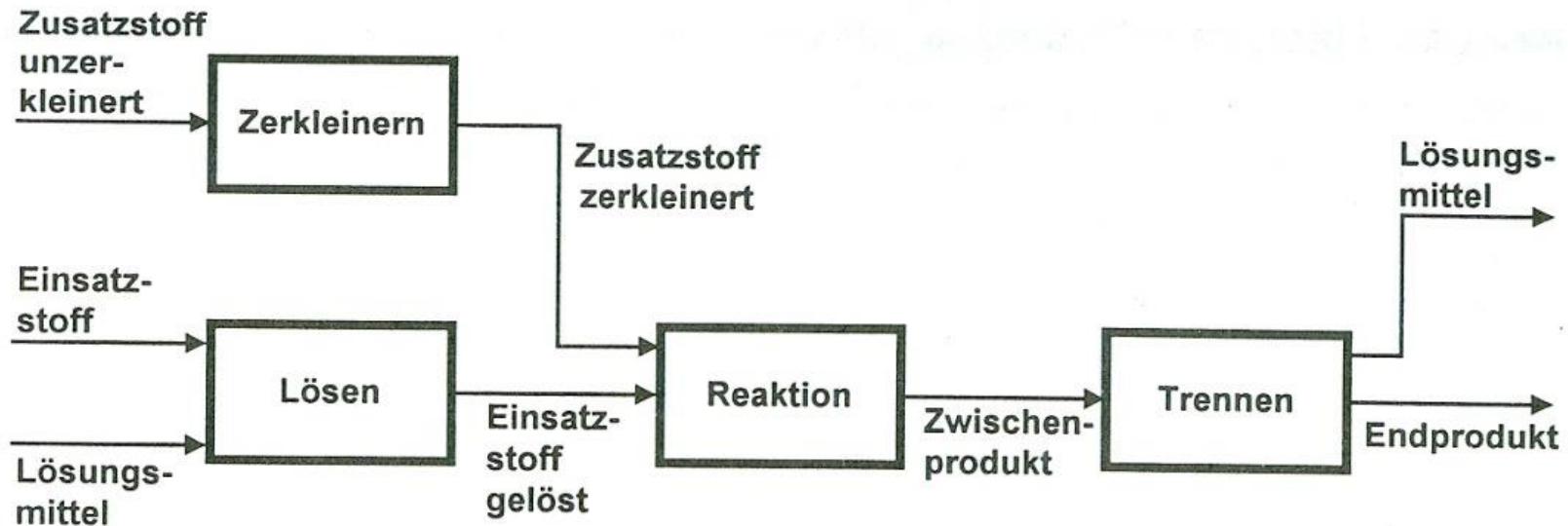
Verfahrensfließbild



MSR-Fließbild
(auch: RI- bzw. PI-Fließbild)

Informationsflüsse im Betrieb

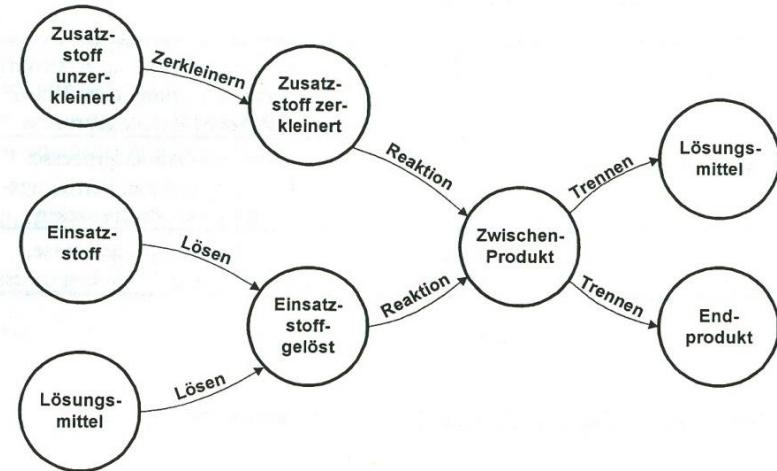
Grundfließbild



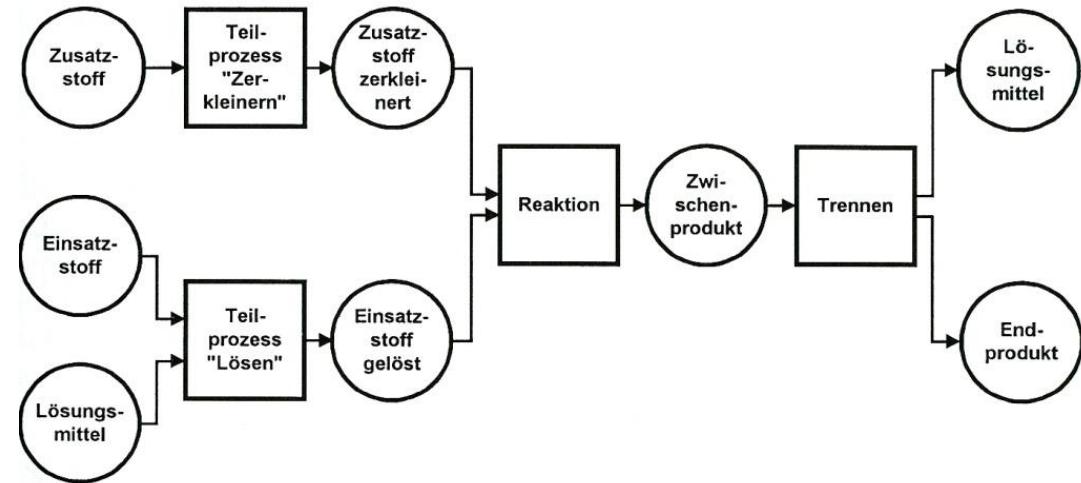
Informationsflüsse im Betrieb

Grundfließbild

Zustandsdiagramm:



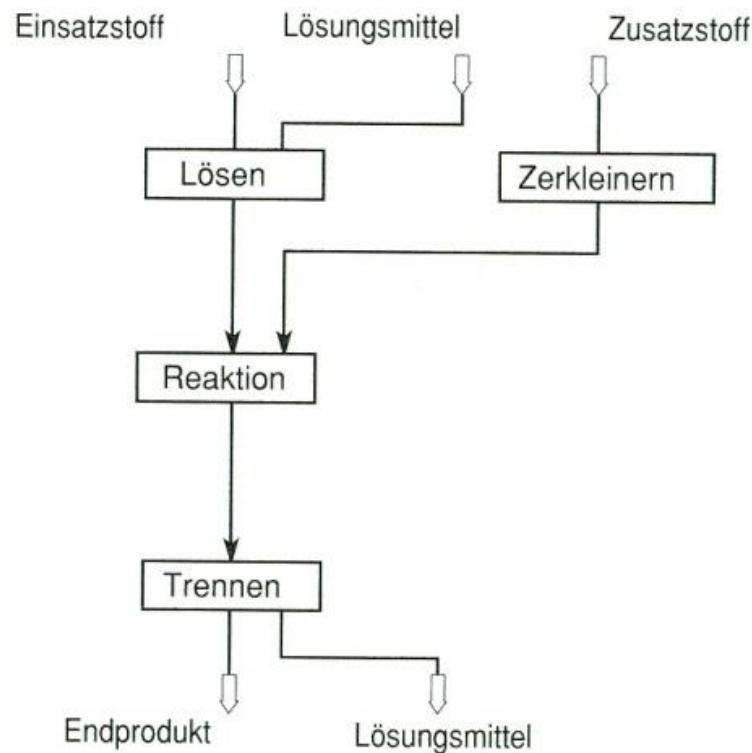
Phasenmodell:



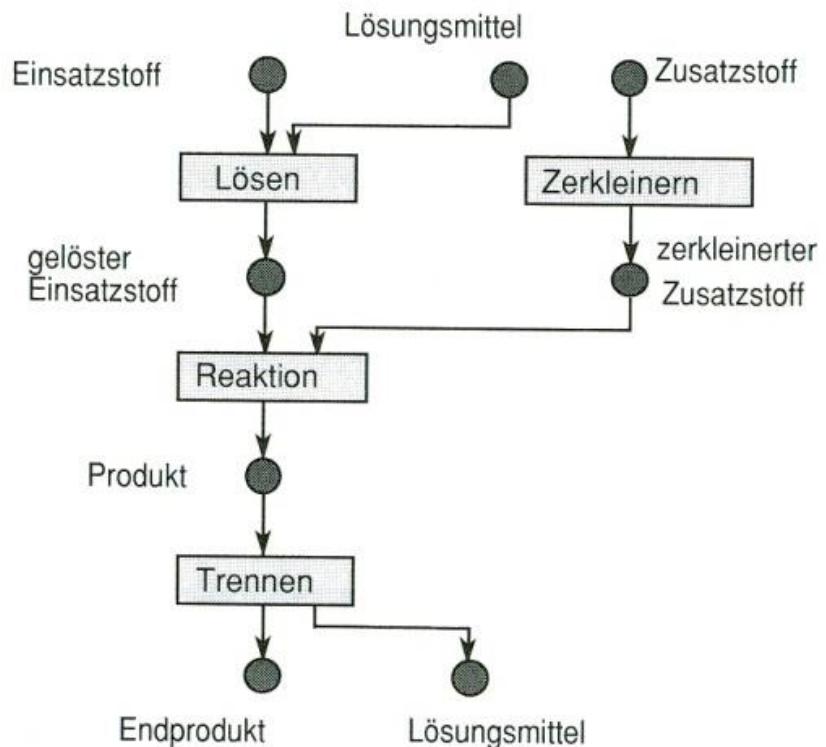
Informationsflüsse im Betrieb

Grundfließbild/Phasenmodell

Grundfließbild [nach DIN28004]

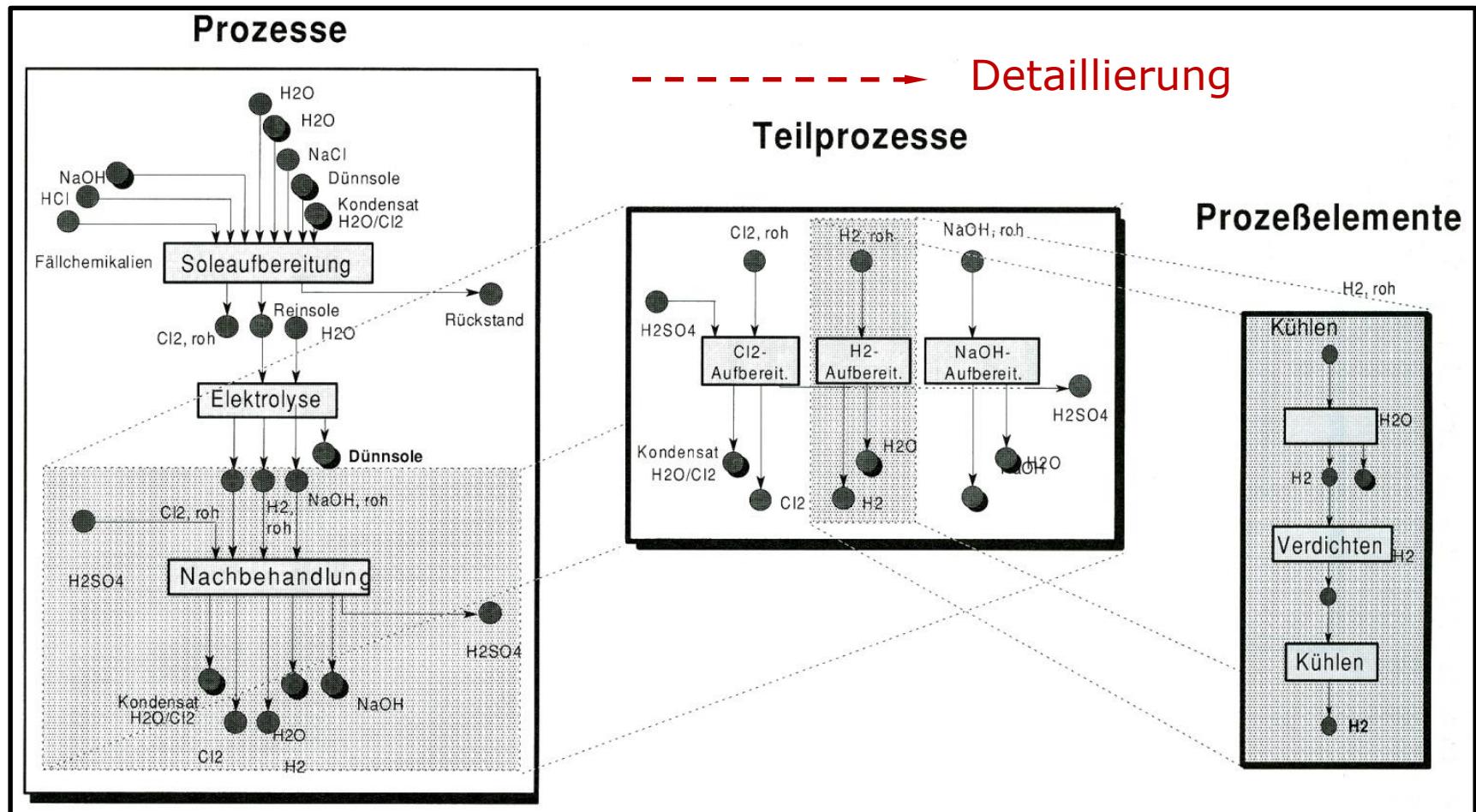


Phasenmodell



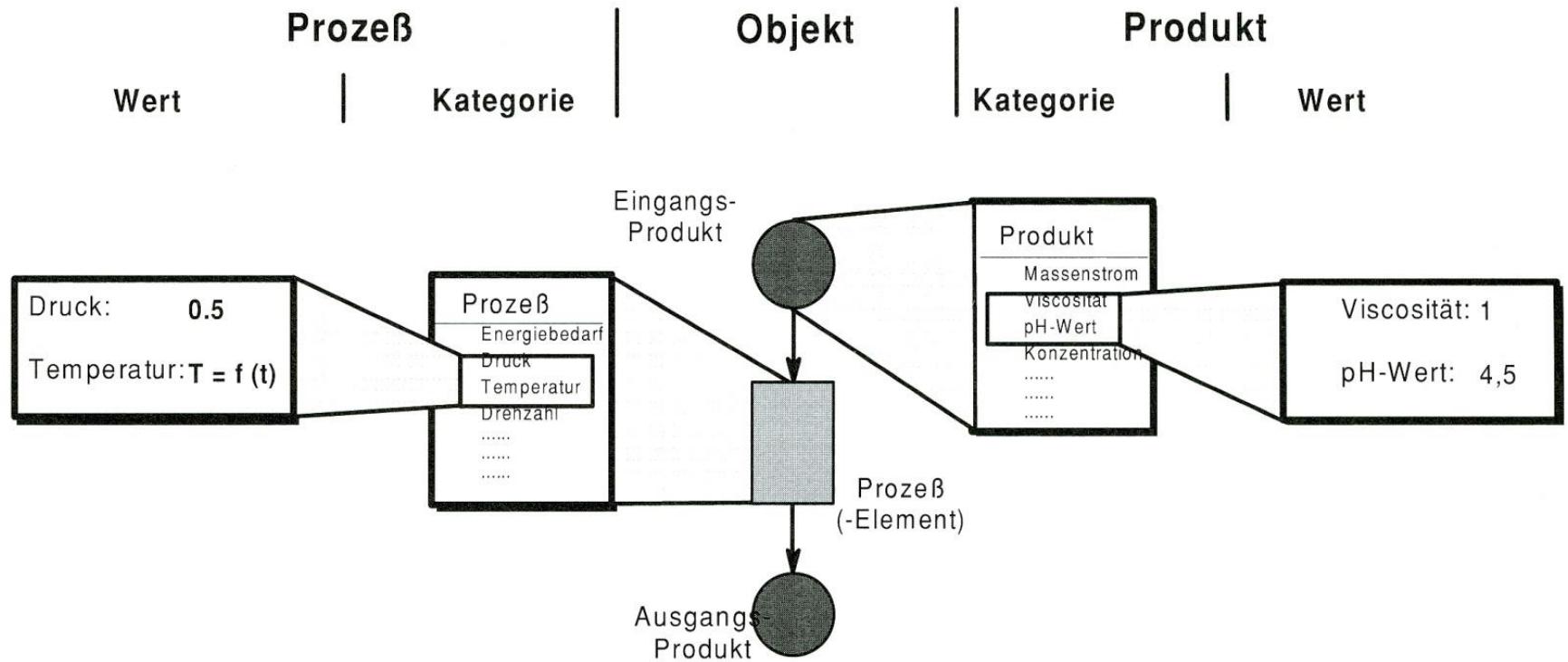
Informationsflüsse im Betrieb

Beispiel: Chlor-Alkali-Elektrolyse



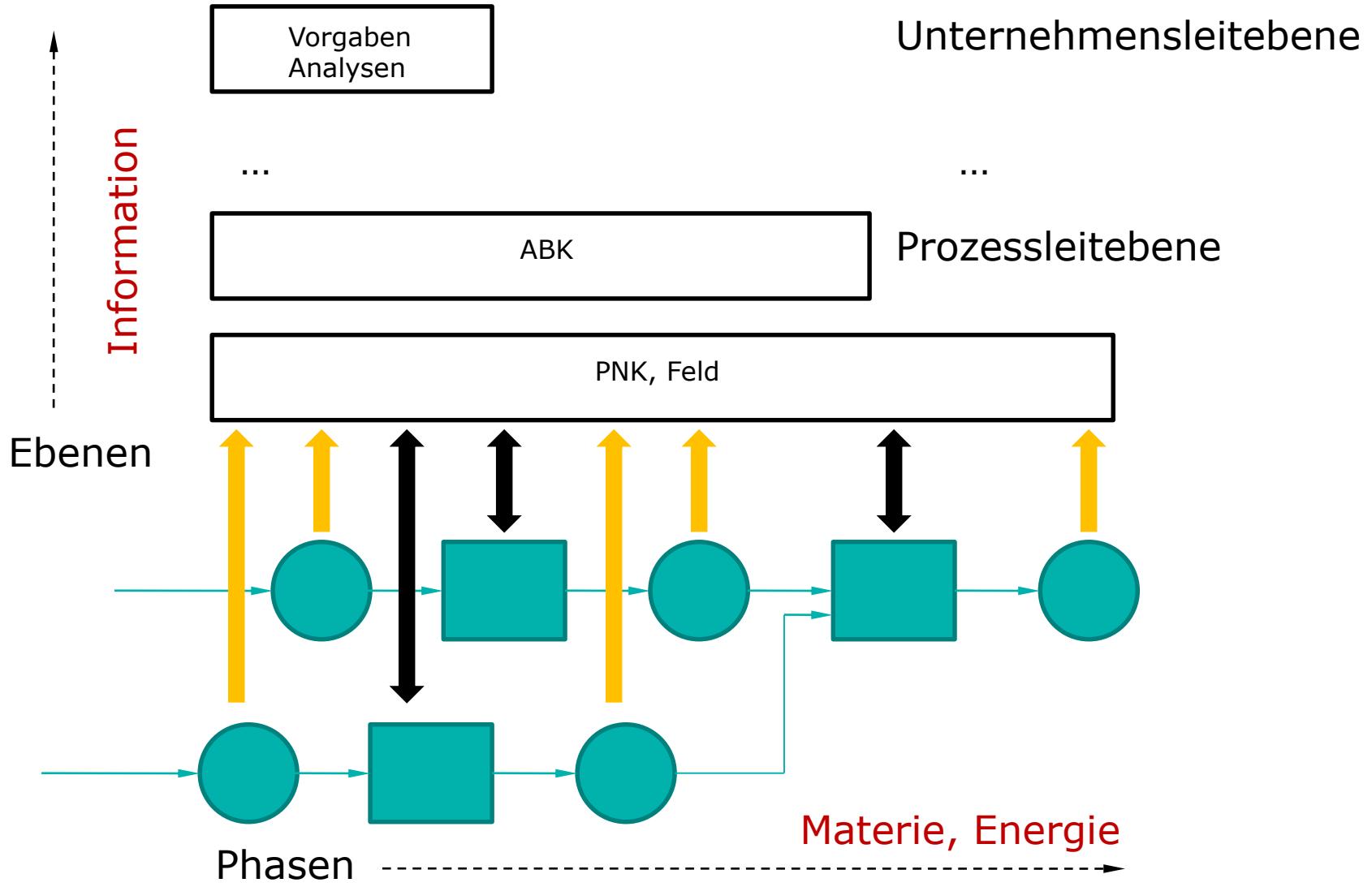
Informationsflüsse im Betrieb

Weitere Detaillierung



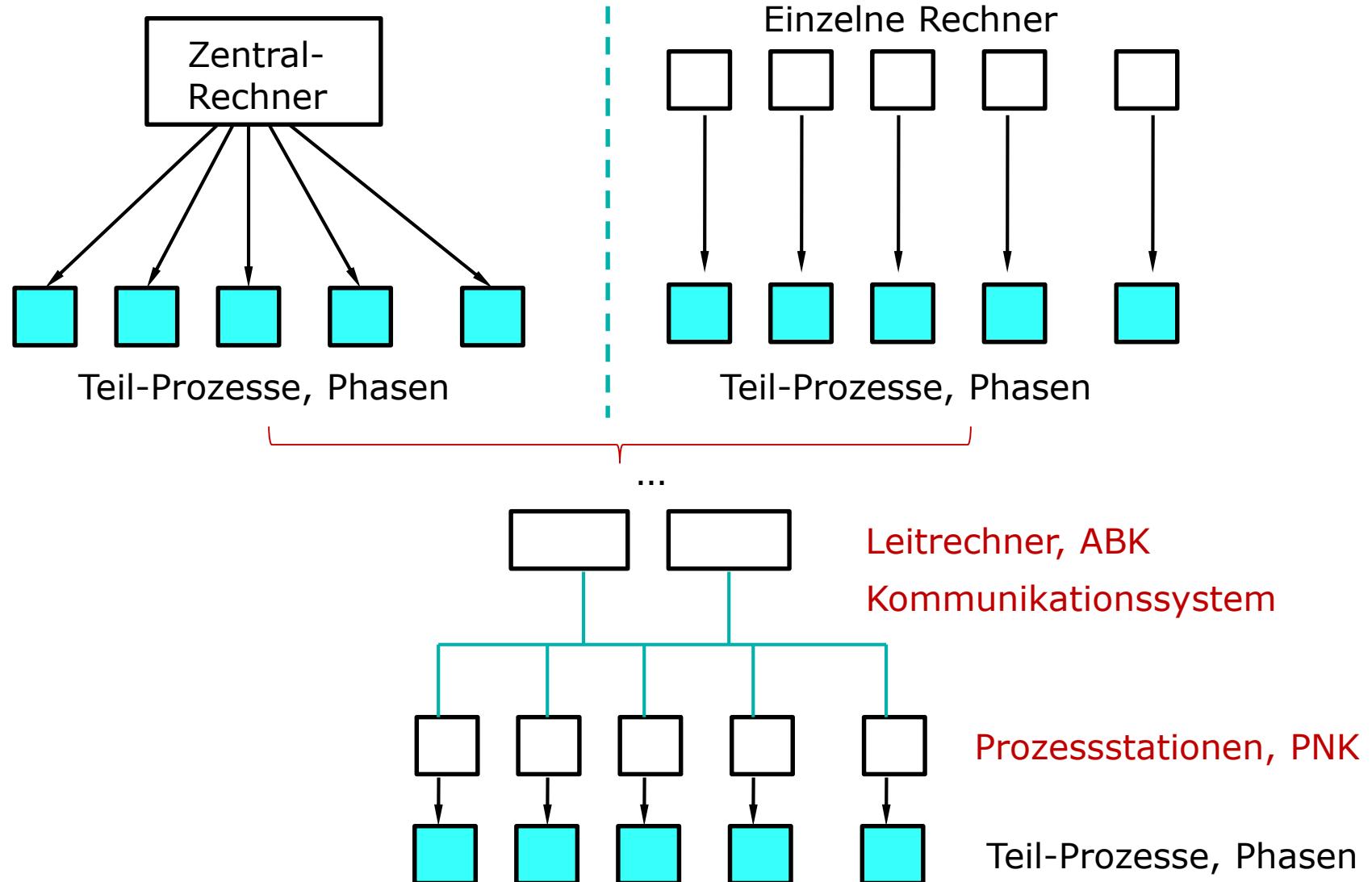
Informationsflüsse im Betrieb

Ebenen- und Phasenmodell

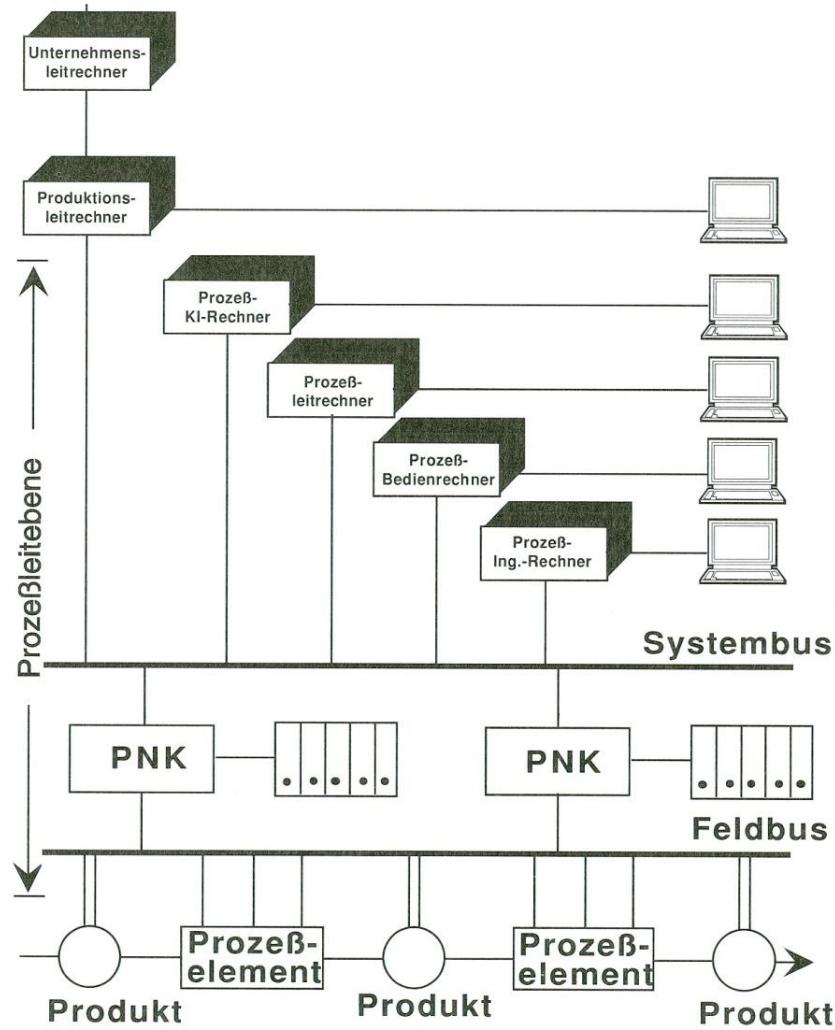


Informationsflüsse im Betrieb

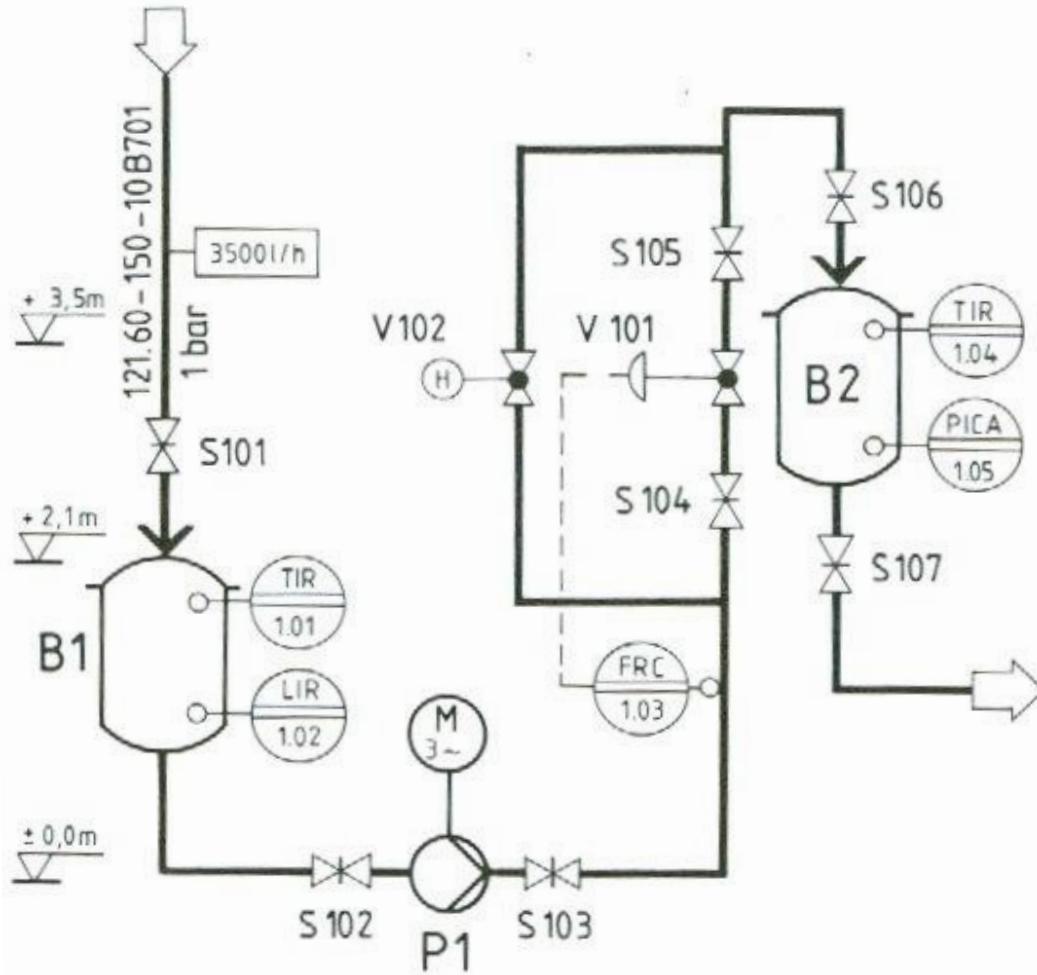
Hardware



Informationsflüsse im Betrieb Leitebenen

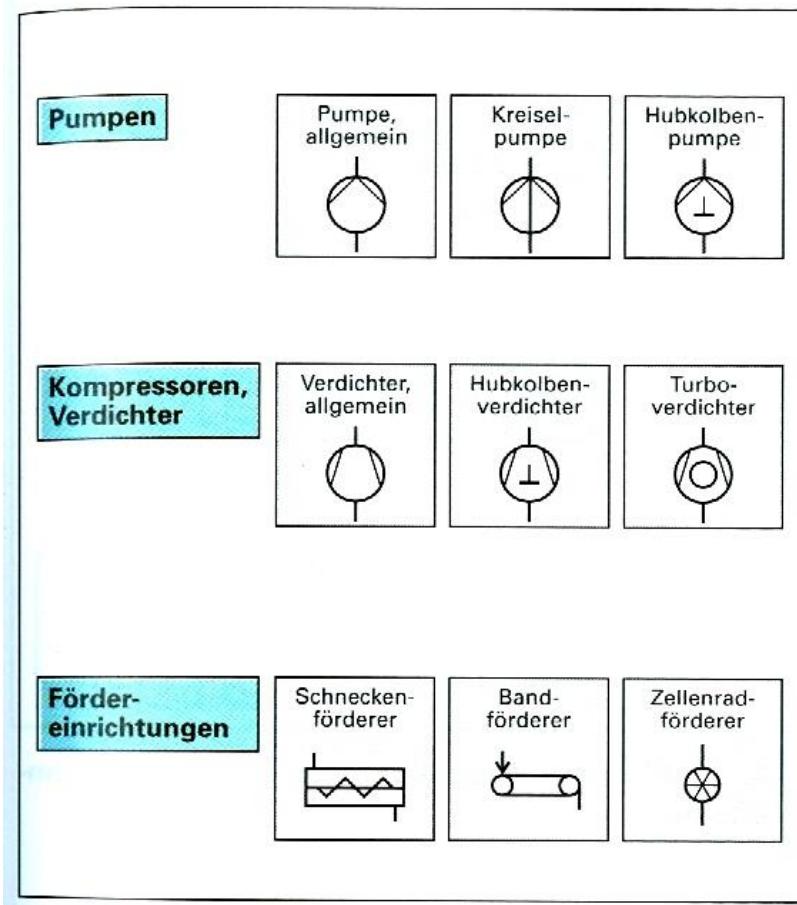


Fließbilddarstellung Rührkessel



Fließbilddarstellung Maschinen, Apparate, Armaturen

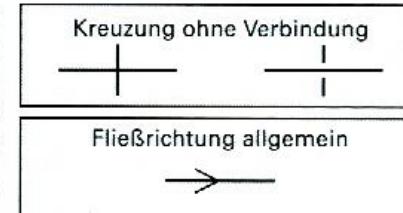
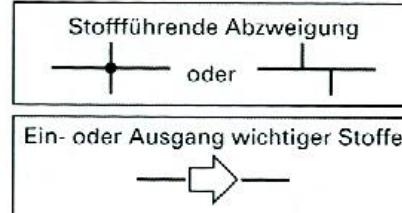
Nach DIN 19227, 2429, EN ISO10628



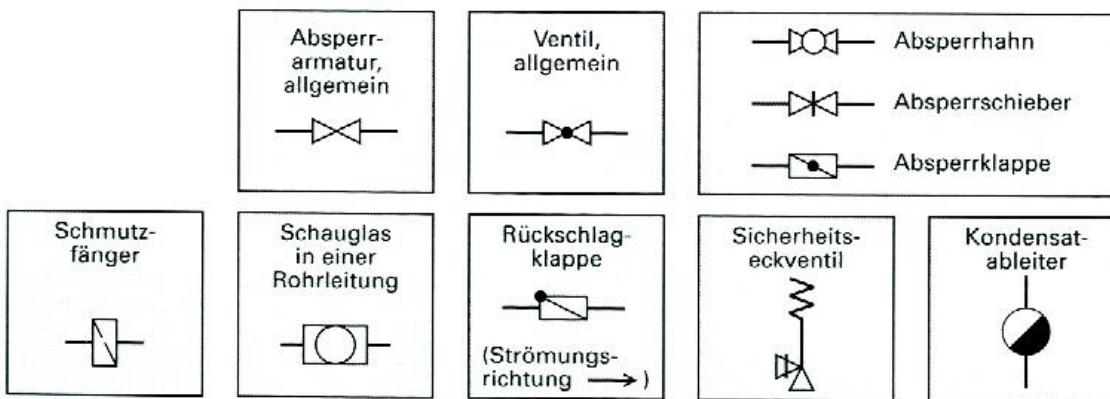
Fließbilddarstellung Leitungen, Armaturen, Stellgeräte

Leitungen

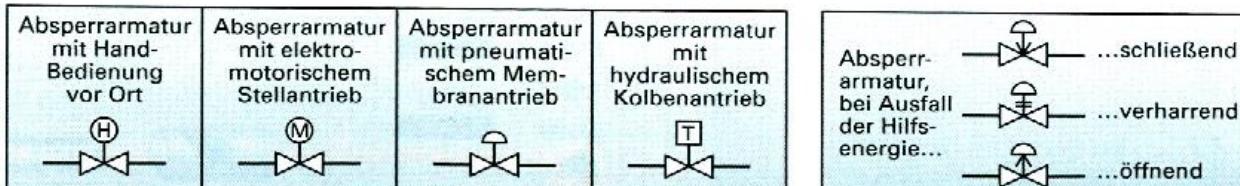
1,0mm	Hauptprodukt-leitung
0,5mm	untergeordnete Produktleitung
0,25mm	Steuer- oder Signalleitung



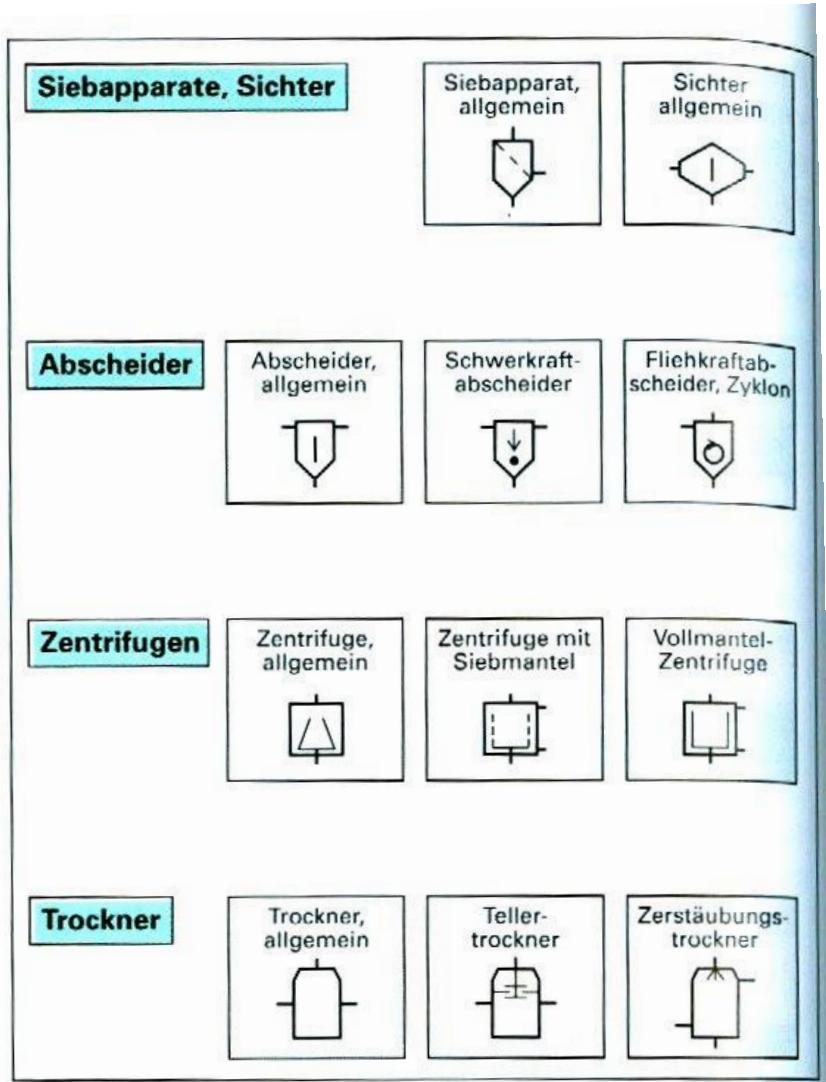
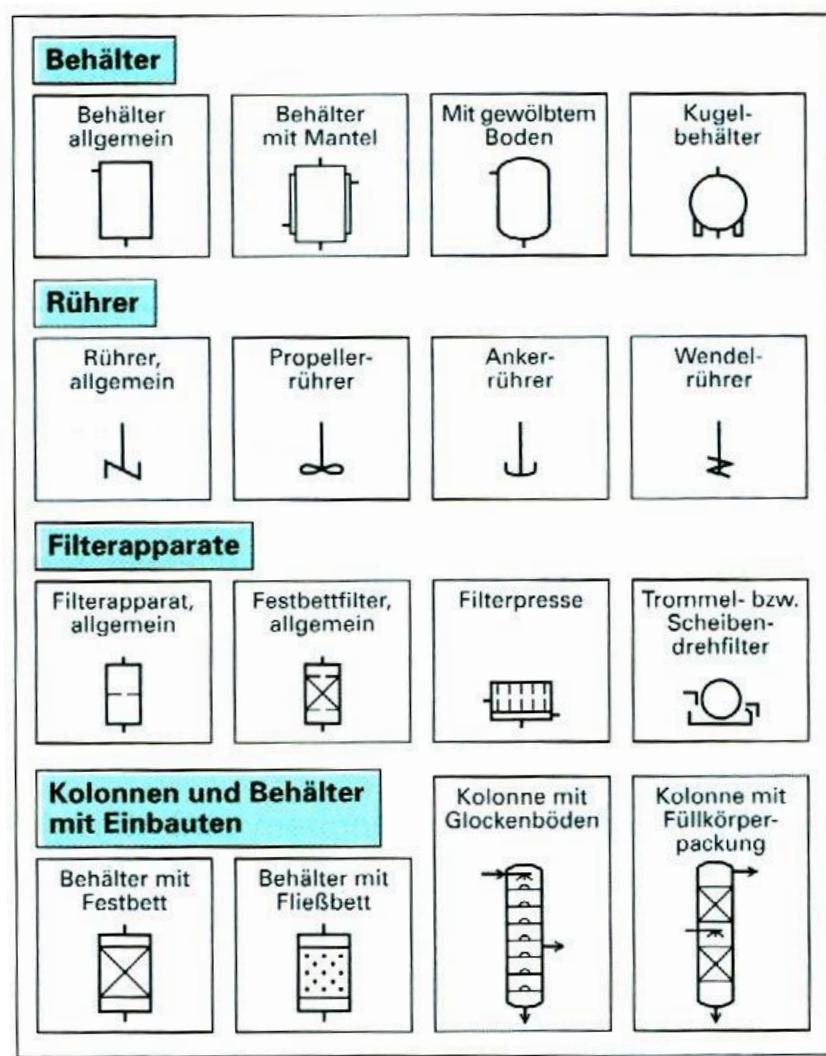
Ventile und Armaturen



Stellantriebe



Fließbilddarstellung Behälter, Speicher, Reaktoren



Fließbilddarstellung

Kennbuchstaben nach DIN 19227

Kennbuchstabe		
Erstbuchstabe	Ergänzungsbuchstabe	Folgebuchstabe
D Dichte E Elektrische Größe (z. B. Spannung, Strom) F Durchfluss (engl. flow) G Abstand, Länge, Stellung, Getriebestellung, Hub (engl. gear) L Füllstand oder Stand einer Trennschicht (engl. level) M Feuchtigkeit (engl. moisture) P Druck (engl. pressure) Q Stoffeigenschaft, Qualitätskenngröße R Strahlungsgrößen (radio) S Geschwindigkeit, Drehzahl, Frequenz (engl. speed) T Temperatur U Zusammengesetzte Größe V Viskosität (Zähigkeit) W Gewicht (engl. weight) X, Y Sonstige Größen (betriebsintern festlegbar)	D Differenz F Verhältnis (engl. factor) J Messstellenabfrage, -umschaltung Q Integral, Summierung (engl. quotilization)	A Alarmierung, Grenzwertmeldung C Regelung, Konstanthaltung I Anzeige E Noteingriff (engl. emergency) O Optisches Sichtzeichen R Registrierung, Speicherung des Verlaufs T Messaufnehmerfunktion (Transmitter) S Schaltung, Ablaufsteuerung Y Rechenfunktion Z Noteingriff

Darüber hinaus ist es üblich, ein „H“ als Erstbuchstabe für „Handeingaben“ oder „Handeingriffe“ zu verwenden, obwohl es sich dabei um keinen Messwert bzw. um keine Prozessgröße handelt. Darüber hinaus bedeuten die Symbole

- + oder H: ... bei zu hohem Wert
- oder L: ... bei zu niedrigem Wert

Hinweise:

- Fehlt der Querstrich im EMSR-Stellen-Symbol, erfolgt keine Fernübertragung, sondern nur eine örtliche Anzeige.
- Kennbuchstaben für die EMSR-Stellen sind nicht zu verwechseln mit den Ausrüstungsbezeichnungen wie z. B. HV 11 für Handventil Nr.11.
- Sie sind ebenfalls nicht zu verwechseln mit den üblichen Formelzeichen für Sollwert (w), Stellwert (y) und Störgröße (z).

Fließbilddarstellung Kennbuchstaben nach DIN 19227

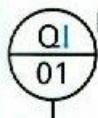
Beispiele



Temperaturanzeige mit Grenzwertmeldung
bei zu hohem und zu niedrigem Wert,
EMSR-Stellennummer 11



Schaltung per Hand, EMSR-Stellennummer 406



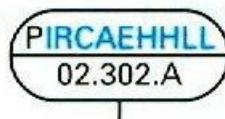
Anzeige einer
Stoffeigenschaft (pH-Wert),
EMSR-Stellennummer 01



Durchfluss-Summierung mit Anzeige und Abschaltfunktion
(aus z. B. 3 Liter pro Sekunde werden durch die Summierung
über 60 Sekunden → 180 Liter),
EMSR-Stellennummer 304.C



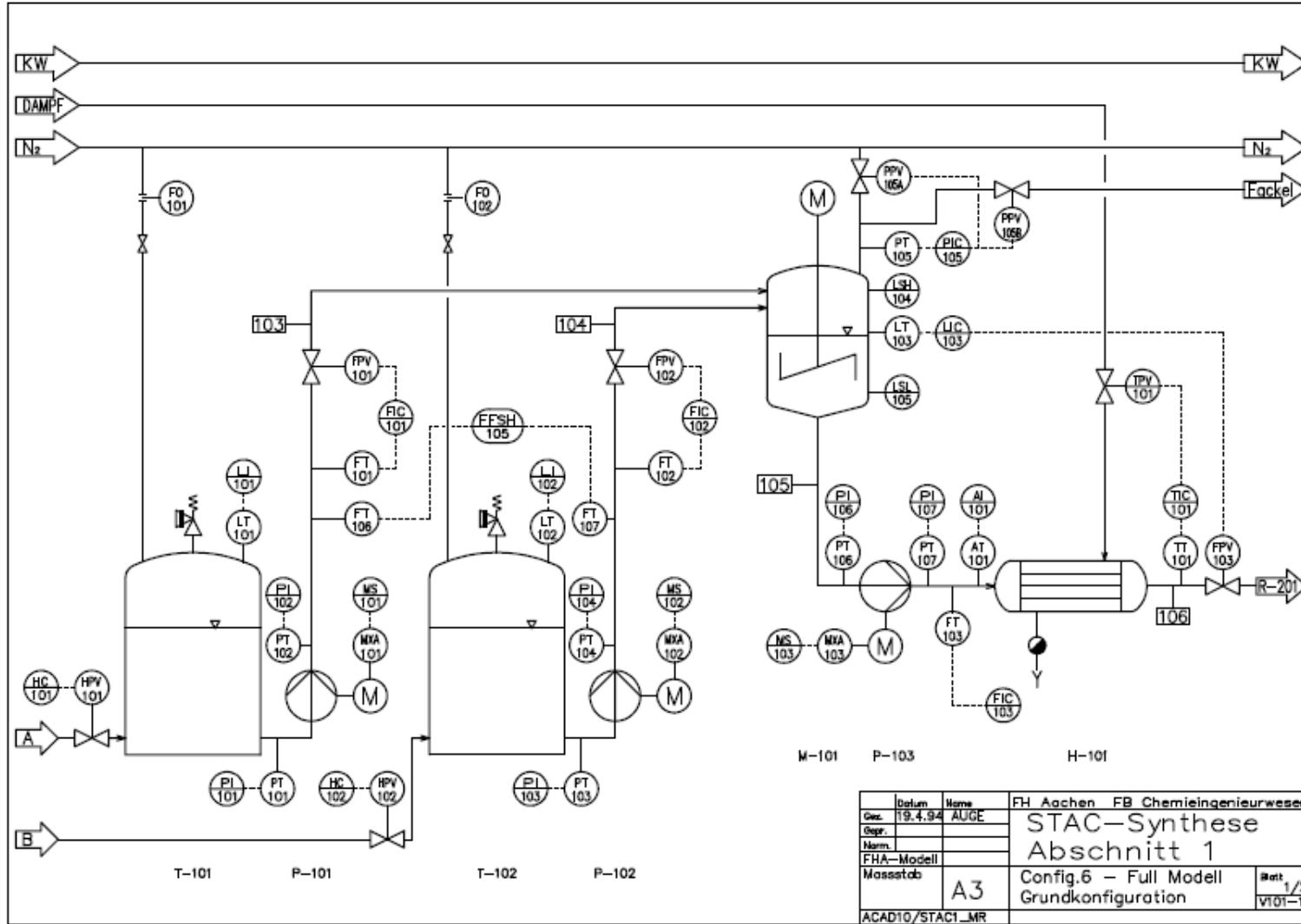
Druckdifferenz-
Anzeige,
EMSR-Stellennummer 23



Anzeige, Registrierung und Regelung eines Druckes.
Alarmierung bei zu hohem und bei zu niedrigem Wert.
Notabschaltung bei extrem zu hohen oder zu niedrigen
Werten, EMSR-Stellennummer 02.301.A

Hinweis: Der Aufbau der EMSR-Stellennummer ist nicht genormt, sondern wird betriebsintern festgelegt.

Beispiel: Synthese-Prozess, erster Prozessabschnitt „Lagerung, Mischen, Vorwärmern“

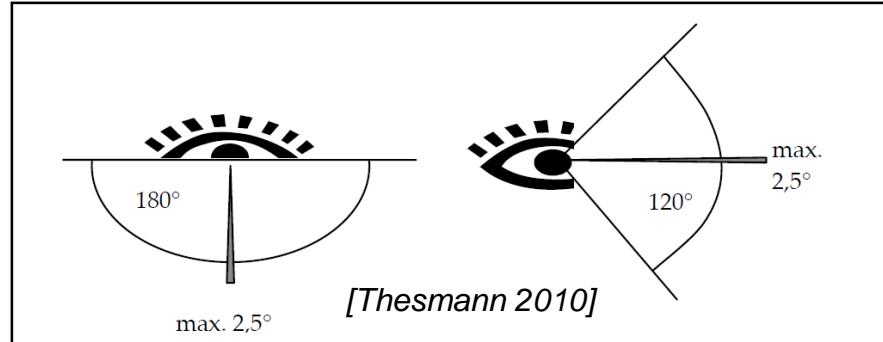


Design grafischer Benutzerschnittstellen

Physiologie des Benutzers

Gesichtsfeld

- > Scharfes Sehen nur $\sim 2,5^\circ$
- > Rest wird durch schnelle Augenbewegung erfasst.
- > Vermeidung unnötiger Augenbewegung durch Gruppierung, im Bereich von $\sim 5^\circ$, d.h. 14 Buchstaben und 7 Zeilen



Hemisphärenmodell

- > Text im rechten und Bilder im linken Bereich
- > Räumliches Wahrnehmungsvermögen im linken Bereich besser. Also: Navigationsleisten links!



Die linke Gehirnhälfte nimmt den Text auf. Logik, Analyse, Zahlen, Rationalität, Rechnen, Sprache u. ä. gehören zu den Stärken des linken Hemisphärenbereichs.

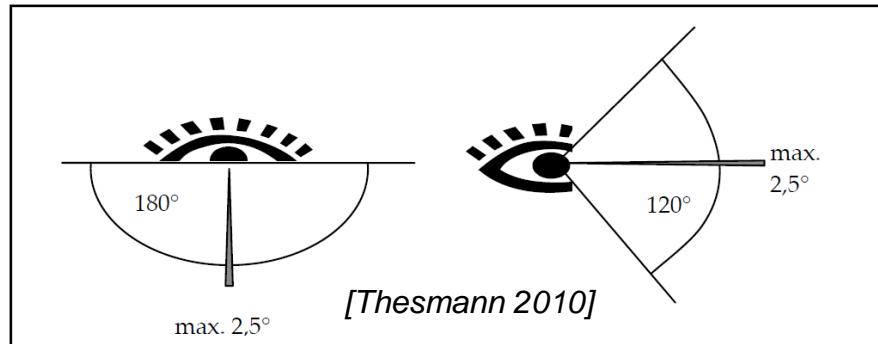


[Thesmann 2010]



Design grafischer Benutzerschnittstellen

Physiologie des Benutzers



Die linke Gehirnhälfte nimmt den Text auf. Logik, Analyse, Zahlen, Rationalität, Rechnen, Sprache u. ä. gehören zu den Stärken des linken Hemisphärenbereichs.



[Thesmann 2010]



Gesichtsfeld

- > Scharfes Sehen nur $\sim 2,5^\circ$
- > Rest wird durch schnelle Augenbewegung erfasst.
- > Vermeidung unnötiger Augenbewegung durch Gruppierung, im Bereich von $\sim 5^\circ$, d.h. 14 Buchstaben und 7 Zeilen

Hemisphärenmodell

- > Text im rechten und Bilder im linken Bereich
- > Räumliches Wahrnehmungsvermögen im linken Bereich besser. Also: Navigationsleisten links!

Design grafischer Benutzerschnittstellen

Gruppierung

Begrenzte Kognition

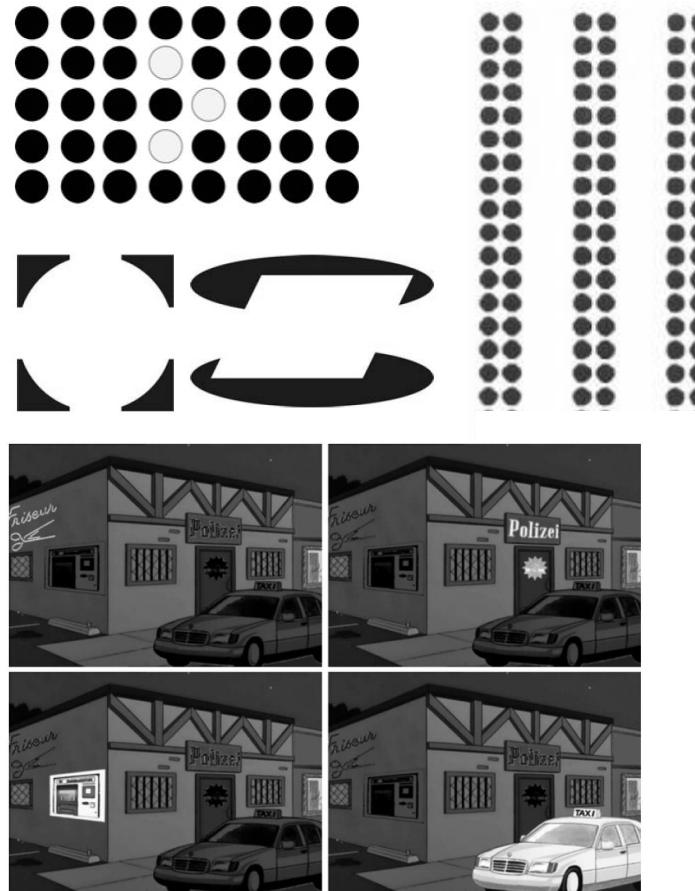
- > Hohe Aufmerksamkeit für ein Objekt erzeugt schärferes Sehen.

Zusammengehörigkeit

- > Ähnlichkeit
- > Nähe
- > Geschlossenheit
- > Kontinuität
- > Prägnanz

Visuelles Gewicht

- > Große Objekte/kleine
- > Helle Objekte/dunkle
- > Abstand zum Mittelpunkt
- > Rechts vor links
- > Rund vor eckig und
- > Senkrecht vor waagerecht



- ▶ Festplatten
- ▼ Grafikkarten
 - AGP-Grafikkarten
 - AGP-Grafikkarten, ATI-Chipsatz
 - AGP-Grafikkarten, nVidia-Chipsatz
 - Computer-Netzteile
 - Computer-Netzteile Zubehör
 - Externe Grafikkarten
 - Grafikkartenküller
- ▶ PClexoress-Grafikkarten

Design grafischer Benutzerschnittstellen

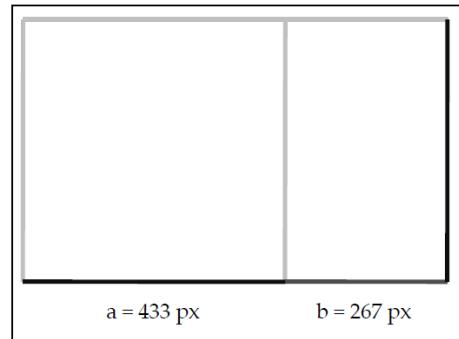
Aufmerksamkeit

Blickreihenfolge

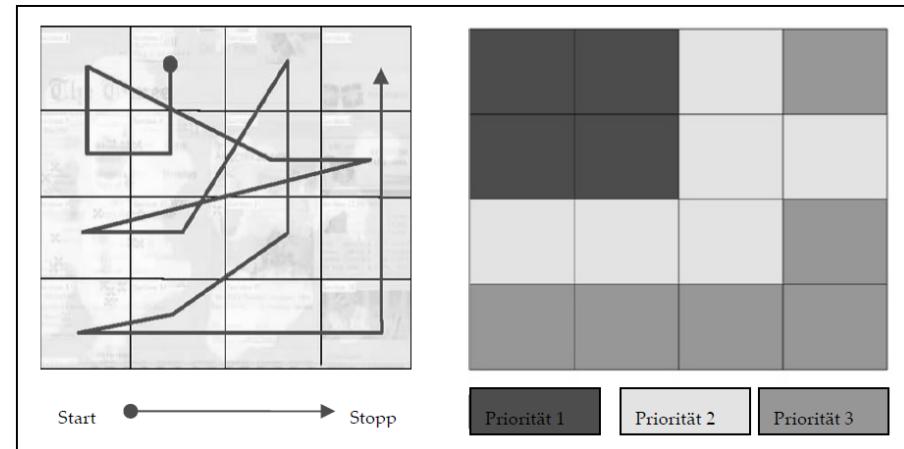
- > Studie mit Eye-Tracking erzeugt F-Shape Pattern für Web Sites.
- > Zwei Horizontale Linien gefolgt von einer vertikalen.
- > Prioritätsverlauf
- > Erwartung

Harmonie

- > Goldener Schnitt 1,618



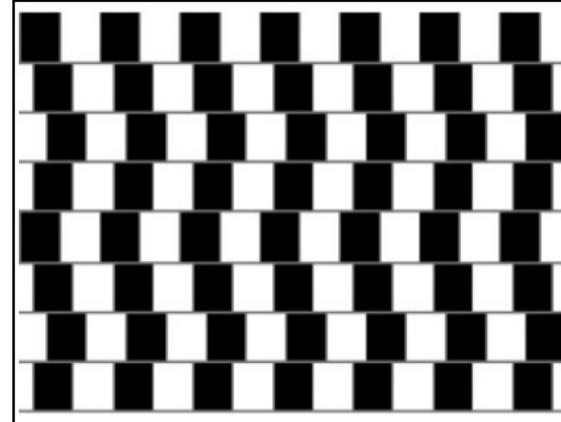
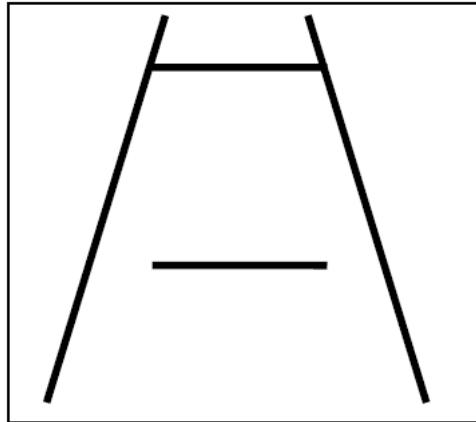
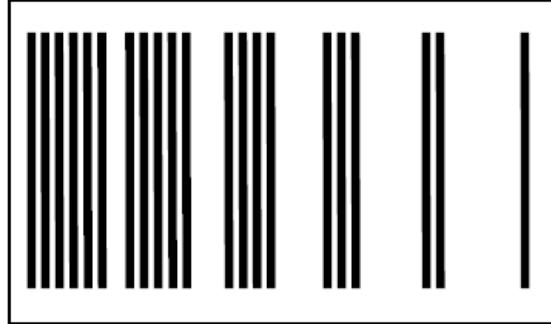
[nngroup.com]



Design grafischer Benutzerschnittstellen

Geometrische Grundformen

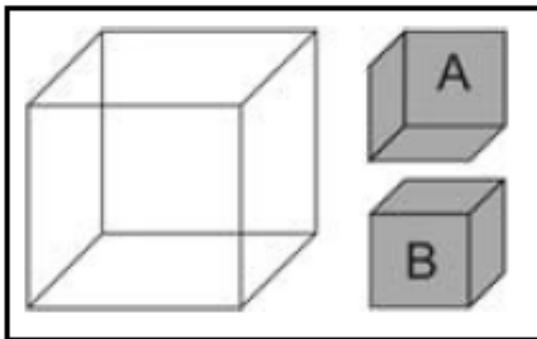
- > Formen können Statik oder Dynamik erzeugen
- > Optische Täuschung



Design grafischer Benutzerschnittstellen

Geometrische Grundformen

- > Formen werden kontextabhängig interpretiert
- > Erzeugung von 3D Effekten



eckig, hart, bestimmt, rational, männlich, Hinweis



spitz, wandelbar, problematisch, indifferent, Gefahr



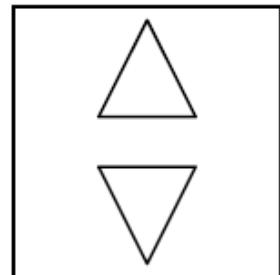
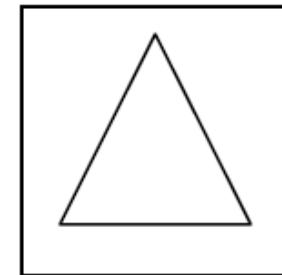
rund, weich, unbestimmt, emotional, weiblich, Verbot



Achtung! Vorsicht!



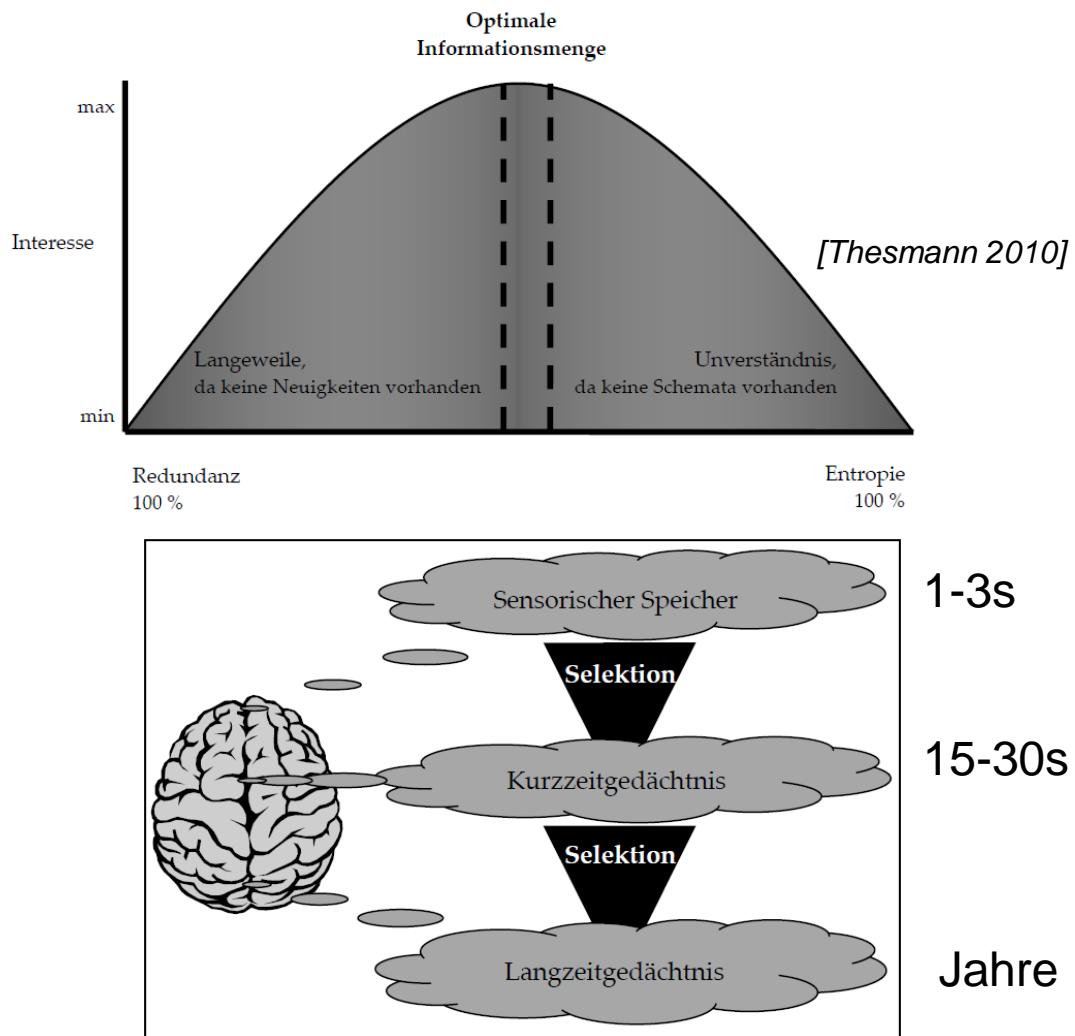
Halt! Stop!



Größe	Beleuchtung	Unschärfe	Vollständigkeit	Perspektive

Design grafischer Benutzerschnittstellen

Informationsfluss



Informationsmenge

- > Gehirn ist nur begrenzt aufnahmefähig und filtert Daten aus
- > Verschiedene Formen der Speicherung und der Speicherdauer
- > Informationen werden zwischen den einzelnen Schichten transferiert durch z.B. bewusstes Wiederholen und Strukturieren

Design grafischer Benutzerschnittstellen

Erste Prinzipien

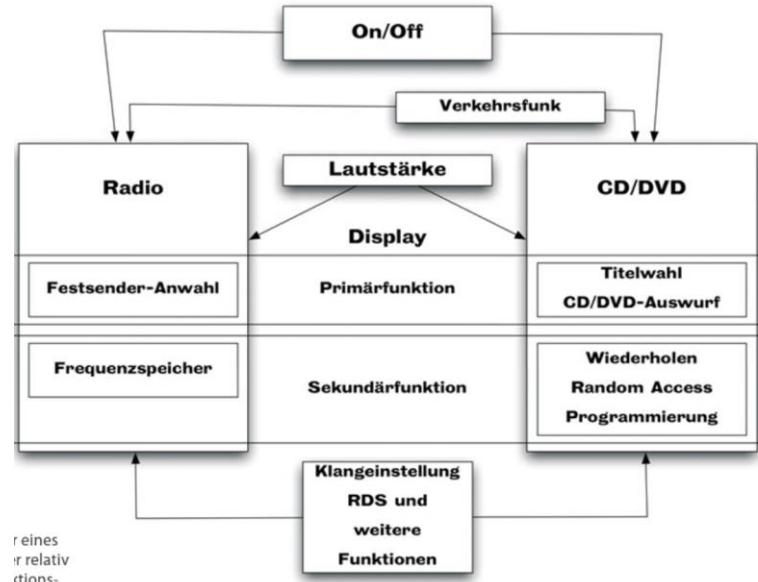
Fokus auf den **Benutzer** und seine **Aufgaben** – nicht auf die Technologie

- > Wer nutzt die Software? Wofür ist sie?
Welche Kenntnisse und Fähigkeiten hat der Benutzer? Barrierefreiheit (Rot-Grün Sehschwäche)?



Zuerst die **Funktion** festlegen, dann die **Präsentation**

- > Welche Daten werden generiert, manipuliert oder visualisiert? Wo kommen die Daten her? Welche Konzepte werden dafür verfolgt? Welche Optionen, Auswahl- und Kontrollmöglichkeiten sind verfügbar?



Design grafischer Benutzerschnittstellen

Erste Prinzipien

Unterstützung des Benutzers und **seiner Sicht** der Aufgabe

- > Wie wird der Benutzer geleitet? Werden unübliche, komplizierte oder verwirrende Tätigkeiten verlangt?
Erwartungskonformität



Verwirrung oder **Ablenkung** des Benutzers vermeiden

- > Werden interne oder zu viele Daten präsentiert? Fehlertoleranz.
Fehlermeldungen.

Vereinfachung des **Lernprozesses**

- > Konsistenz. Individualisierbarkeit.
Selbstbeschreibungsfähigkeit. Undo.



Design grafischer Benutzerschnittstellen

Einführung



Liefern von **Informationen** –
nicht von **Daten**

> Der Bildschirm gehört dem Benutzer. Nur die Daten ändern, die verändert wurden.

Antwortzeiten berücksichtigen

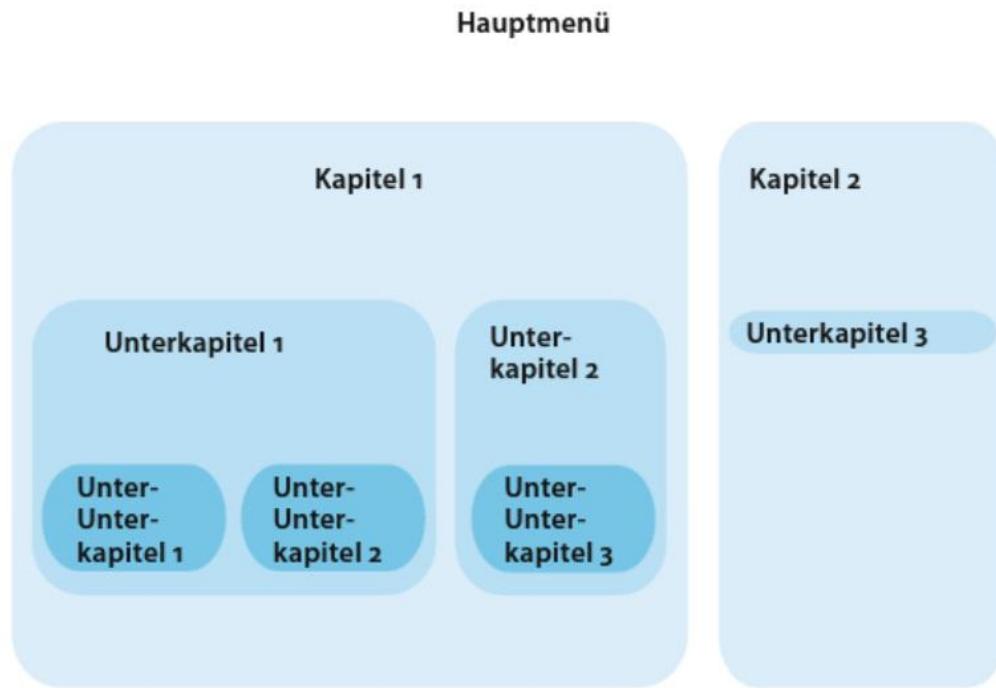
> Bekommt der Benutzer schnell genug Feedback? Progress Balken. Wichtige Informationen zuerst!

Test mit Benutzer und
Verbesserung

> Wie geht der Benutzer mit der Applikation um? Welche Probleme treten auf?

Design grafischer Benutzerschnittstellen

Zusammenfassung

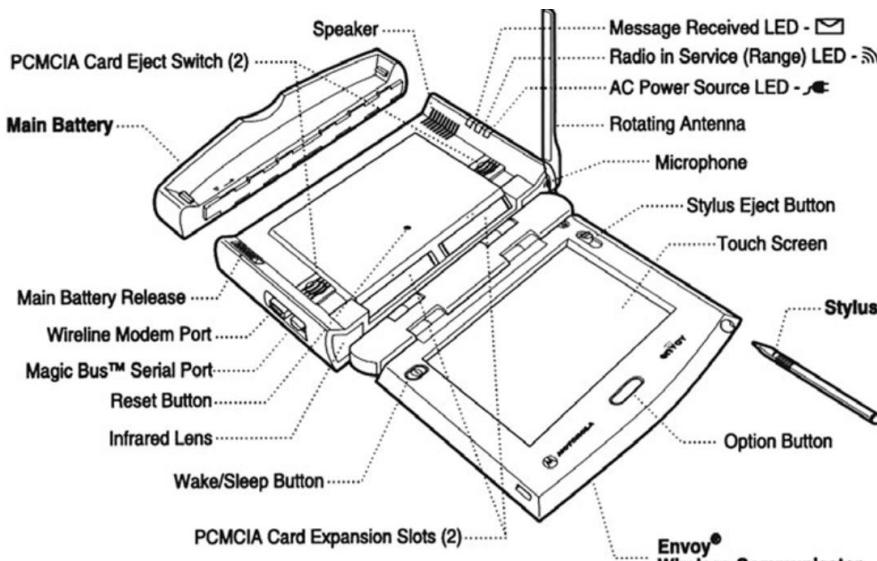


Flowchart

- > Angemessen an Aufgaben
- > Selbstbeschreibungsfähigkeit
- > Steuerbarkeit
- > Erwartungskonformität
- > Fehlerrobustheit
- > Individualisierbarkeit
- > Erlernbarkeit
- > Usability Test

Design grafischer Benutzerschnittstellen

Navigationshilfen und User Controls



Design grafischer Benutzerschnittstellen

Benutzung von Kontrollelementen

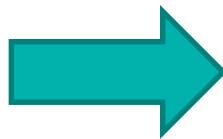
Radio Buttons

Registrierung

Preis: 50€

Schlecht

Der einsame Button



Orientierung

Horizontal

Vertikal

Radio Button zur Auswahl einer Option.
Keine leeren oder beeinflussenden Default Radio Buttons!

Check Boxes

Welches Semester:

Sommer

Winter

Schlecht

Check Boxes als Radio Buttons benutzt



Center Image

Scale to Fit

Guter Gebrauch von Check Boxes.
Keine Negative Check Boxes verwenden!

Design grafischer Benutzerschnittstellen

Fehler beim Gebrauch von User controls

Check Boxes

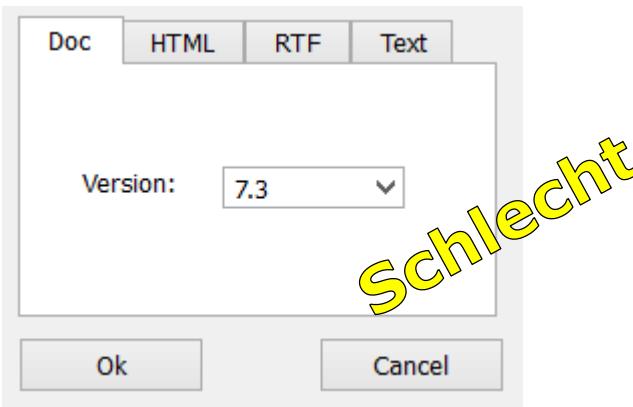


Check Box zur Auswahl
nichtgegensätzlicher
Parameter



Ersatz durch Radio Button

Tabs

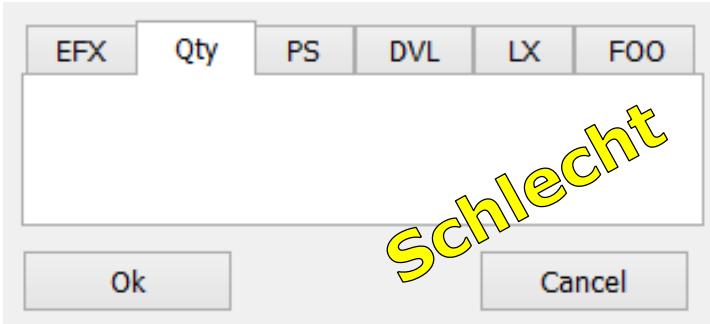


Tabs sollten nur zur Navigation eingesetzt werden. Hier kann der User zwischen verschiedenen Dokumentformaten unterscheiden und je nach Veränderung im Tab wird das Dokument unterschiedlich gespeichert.

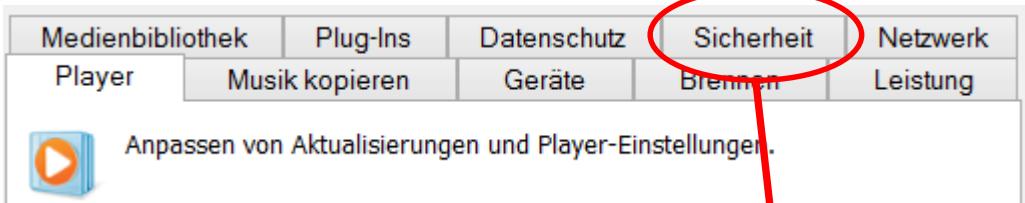
Design grafischer Benutzerschnittstellen

Fehler beim Gebrauch von User controls

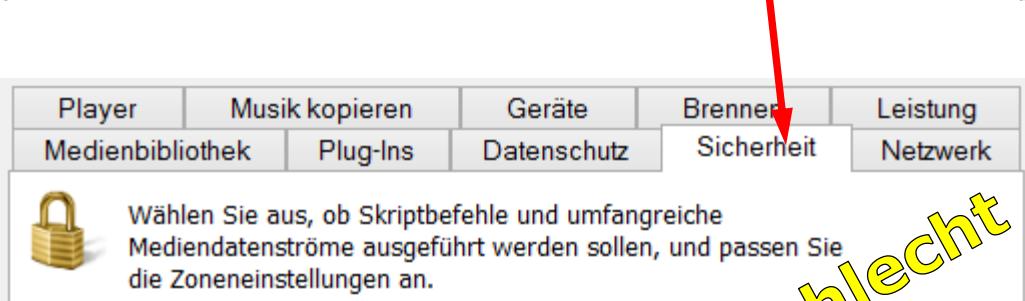
Tabs



Zu viele Tabs mit Abkürzungen. Benutzer muss die Abkürzungen lernen.



Tanzende Tabs. Bei Auswahl des Sicherheit-Tab verändert sich die gesamte Anordnung.

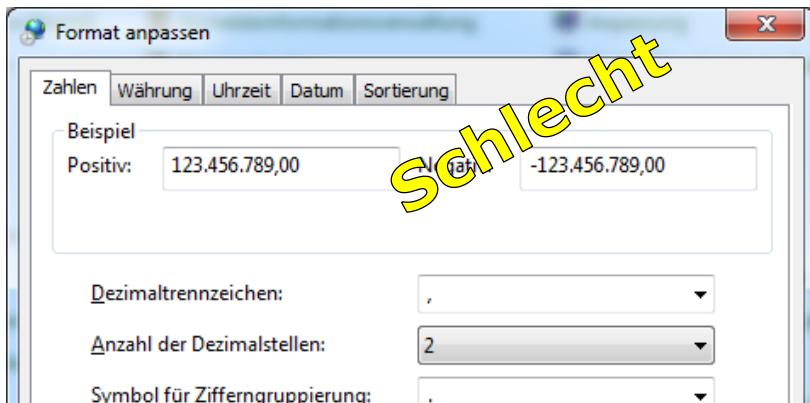


Tip: Tabs sparsam verwenden und nur zur Navigation

Design grafischer Benutzerschnittstellen

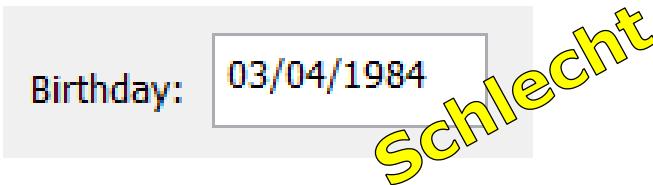
Fehler beim Gebrauch von User controls

Textfelder



Benutzung von editierbaren Textfeldern
für nichteditierbaren Text

Zu häufige Benutzung von editierbaren
Textfeldern



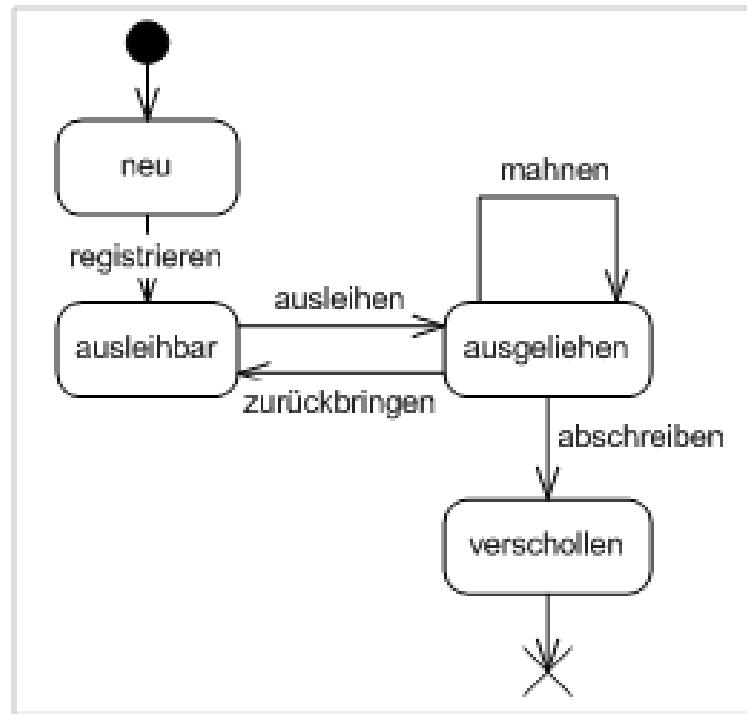
Strukturierte Daten im
Textfeld



Strukturierte Daten in entsprechende
Container.
Länge der Felder sollte ungefähr der
Länge der erwarteten Daten entsprechen.

Modellierungssprache UML

Zustandsautomat



Zustandsautomat mit UML [aus: Wiki]



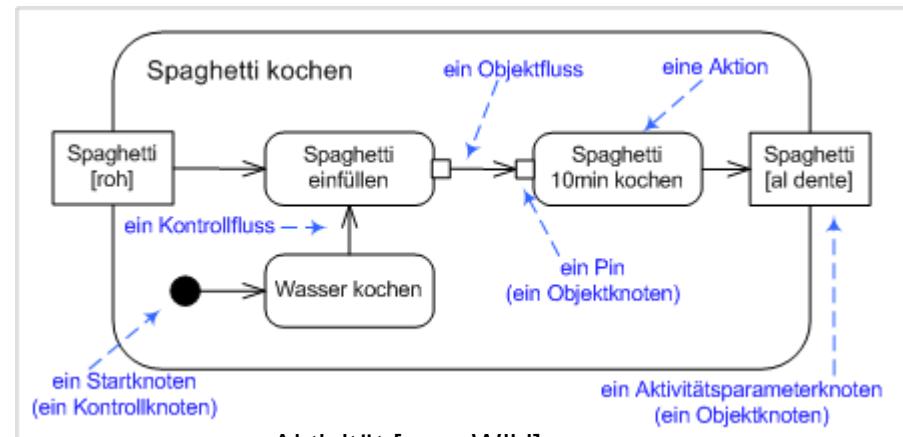
Aktion [aus: Wiki]

Unified Modelling Language

- > Metamodellierung,
- > Schichten,
- > Spracheinheiten.

Spracheinheiten

- > Aktionen, Aktivitäten, Allgemeines Verhalten
- > Anwendungsfälle, Informationsflüsse
- > Interaktionen, Klassen, Komponenten
- > Modelle, Profile, Schablonen
- > Verteilungen und Zustandsautomaten



Aktivität [aus: Wiki]

Clean Coding

Wie schreibt man „saubere Programme“

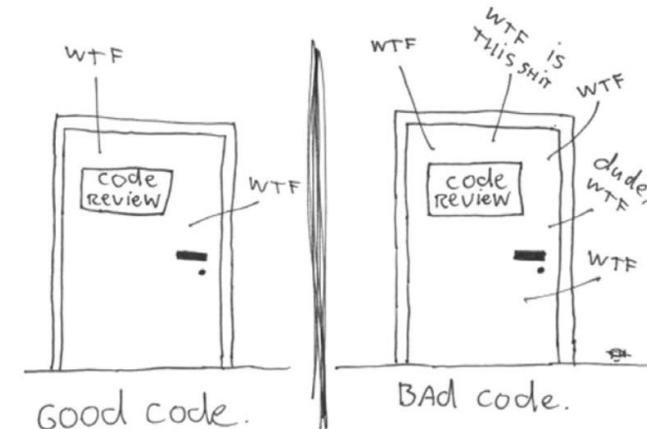
Clean Coding von C. Martin

Prinzipien

- > Don't Repeat Yourself (DRY)
- > Keep it simple, stupid (KISS)
- > Vorsicht vor Optimierungen!
- > Komposition an Stelle von Vererbung

Praktiken

- > Die Pfadfinderregel beachten:
„Verlasse den Campingplatz sauberer,
als Du ihn vorgefunden hast!“
- > Root Cause Analysis
- > Ein Versionskontrollsystem einsetzen
- > Einfache Refaktorisierungsmuster
anwenden
- > Täglich reflektieren



The only valid measurement
of code quality: WTFs/minute

Namensgebung Beispiel

```
int d; // elapsed time in days
```

Schlecht

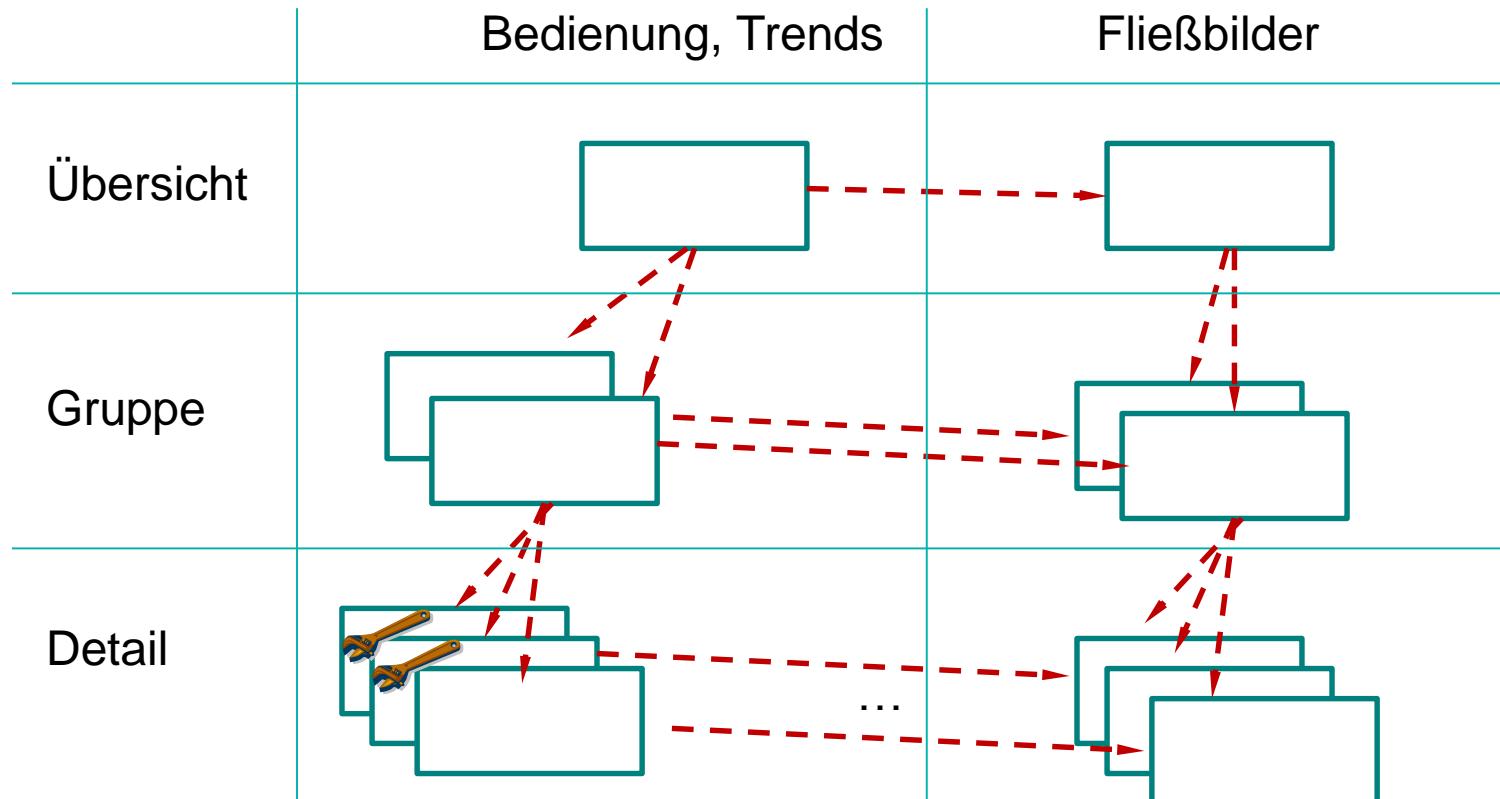
```
public List<int[]> getThem() {  
    List<int[]> list1 = new ArrayList<int[]>();  
    for (int[] x : theList)  
        if (x[0] == 4)  
            list1.add(x);  
    return list1;  
}
```

Schlecht

```
int elapsedTimeInDays;  
int daysSinceCreation;  
int daysSinceModification;  
int fileAgeInDays;
```

```
public List<int[]> getFlaggedCells() {  
    List<int[]> flaggedCells = new ArrayList<int[]>();  
    for (int[] cell : gameBoard)  
        if (cell[STATUS_VALUE] == FLAGGED)  
            flaggedCells.add(cell);  
    return flaggedCells;  
}
```

Hierarchie der „Sichten“ und „Bedienbarkeiten“ in den Prozessen

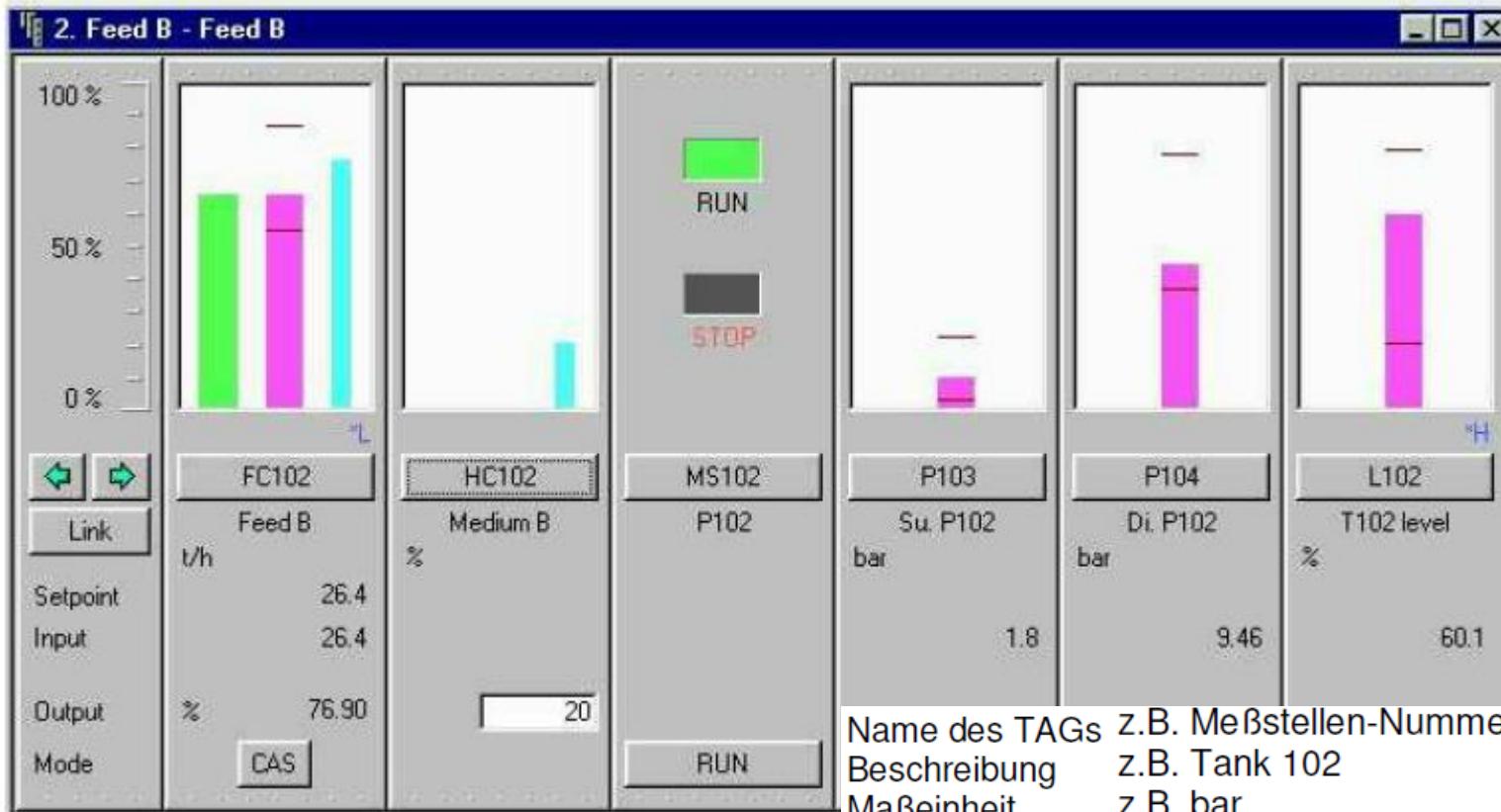


Visualisierung Übersichtsbild eines Synthese-Prozesses

Overview Grid									
Main	Feed B	Feed A	Mixer	Heater					
Md	h d d	h d d		M d M					
Pump 1	Pump 2	Pump 3	M . h d	Regler in Betriebsart HAND Regler in Betriebsart AUTOMATIK --Länge des Striches gibt die Regelabweichung an -- TAG nur manuell bedienbar, z.B. Ventil Meßwertanzeige Schalter	(PID) (PID) (HIC) (DAC) (DIG)				
Trip	Overrd		Zeichen rot Zeichen gelb	-- Alarm ist aufgetreten; -- Alarm wurde quittiert					

Ziel: auf einen Blick kritische Situationen im Prozess erkennen

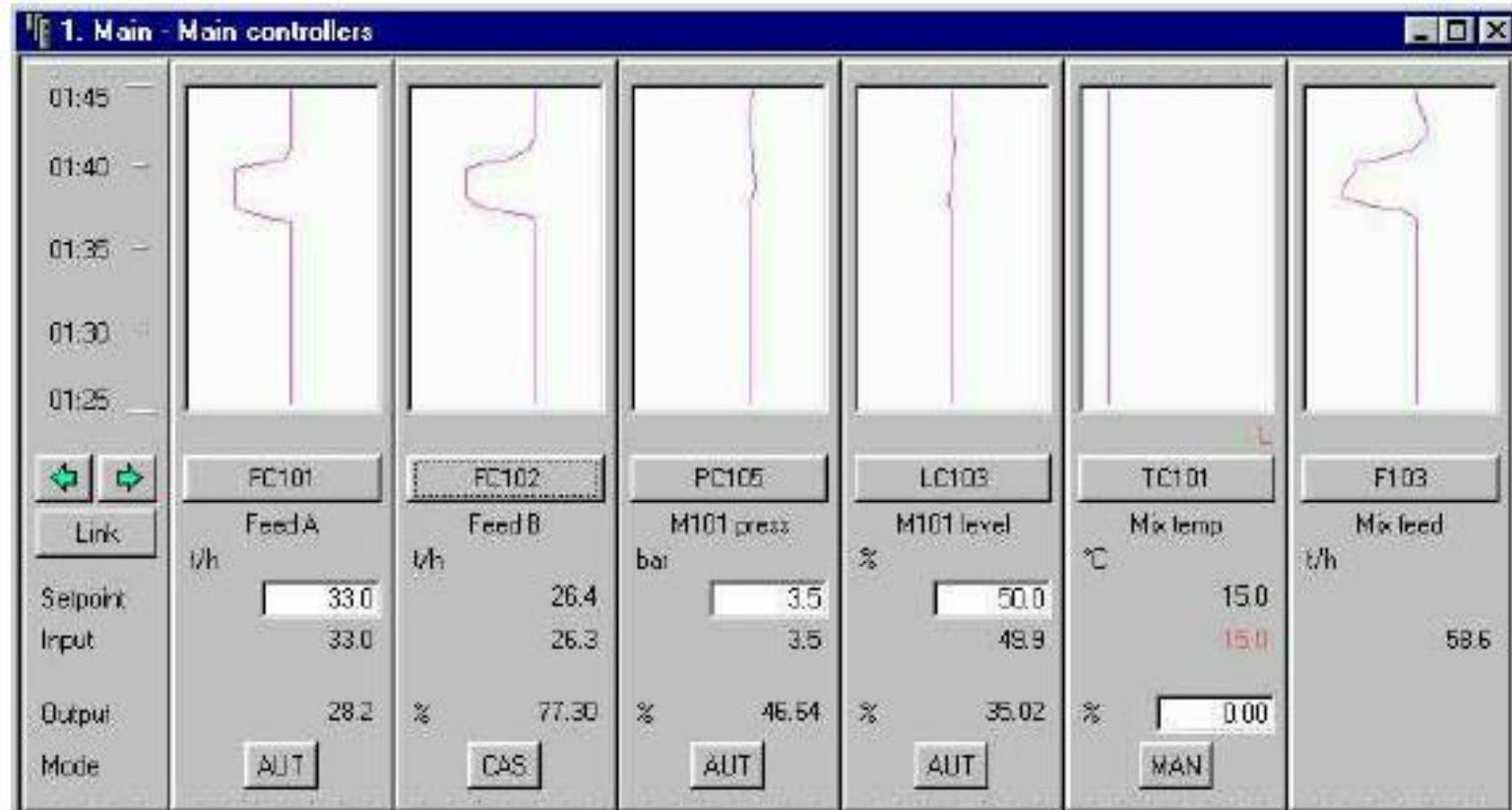
Visualisierung Gruppenbild



Name des TAGs	z.B. Meßstellen-Nummer FC102
Beschreibung	z.B. Tank 102
Maßeinheit	z.B. bar
Setpoint	Sollwert
Input	Meßwert
Output	Stellgröße
Mode	Betriebsmodus z.B. RUN oder AUT

Visualisierung

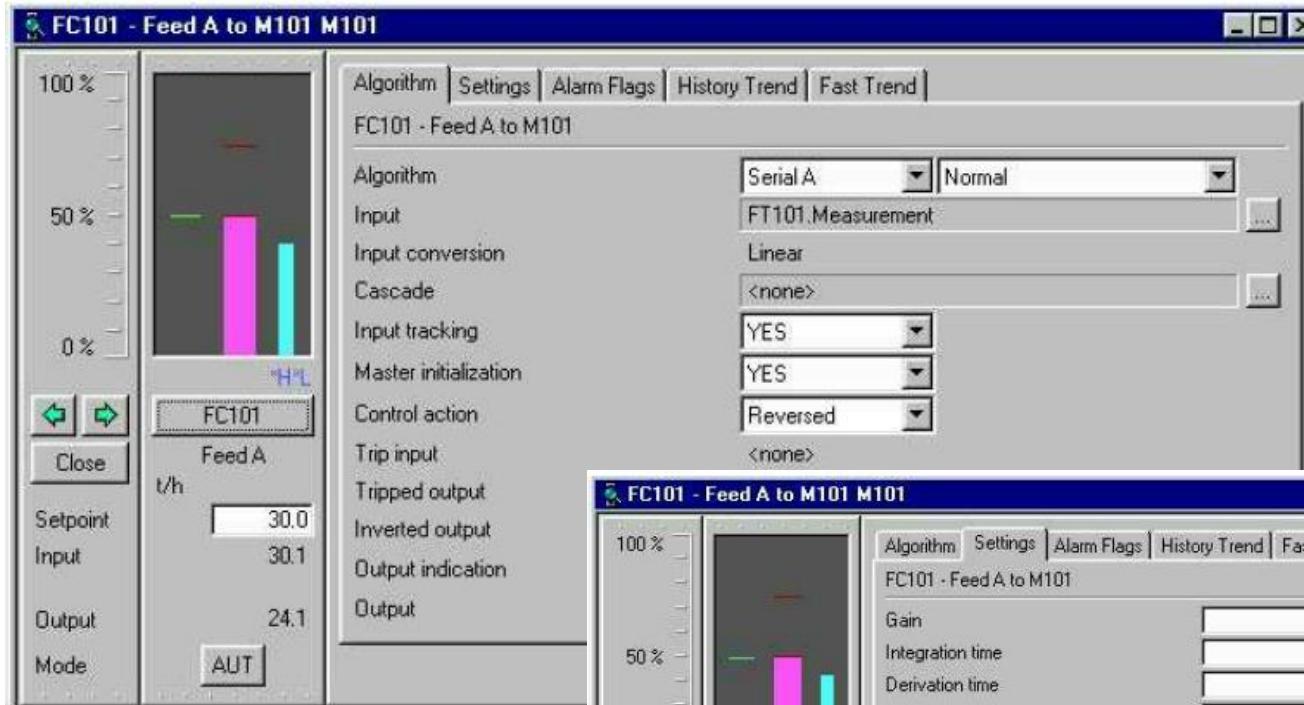
Trenddarstellung im Gruppenbild



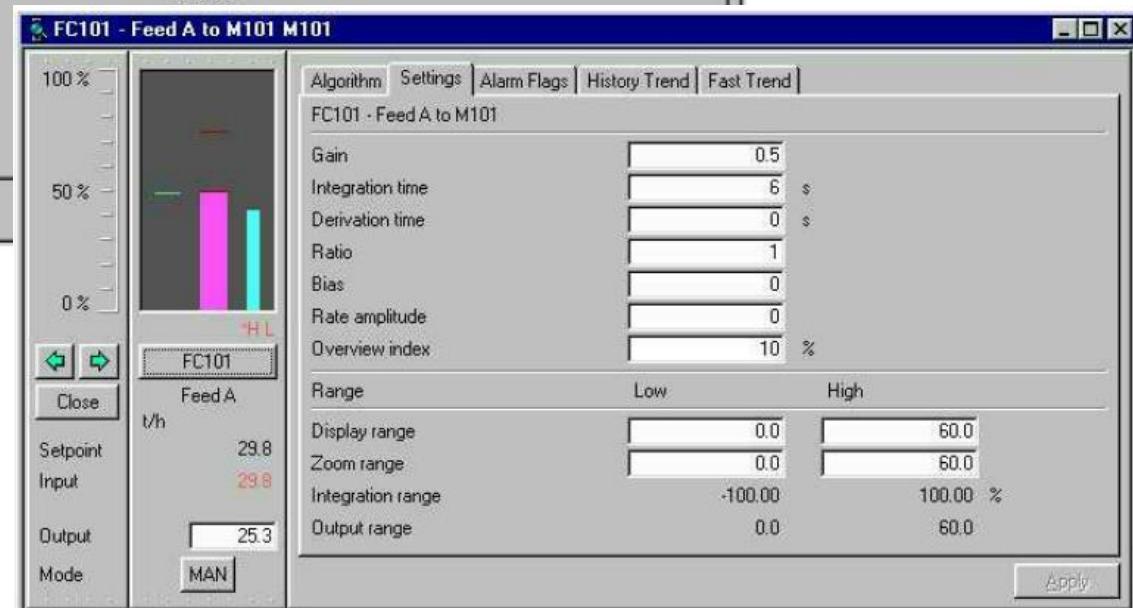
Im Gruppenbild können die Bedienungsoperationen in drei Bereiche aufgeteilt werden:

1. Wechseln der Bilder
2. Bedienung für besondere Funktionen
3. Bedienung der TAGs

Visualisierung Detailbild eines Reglers



Algorithmen

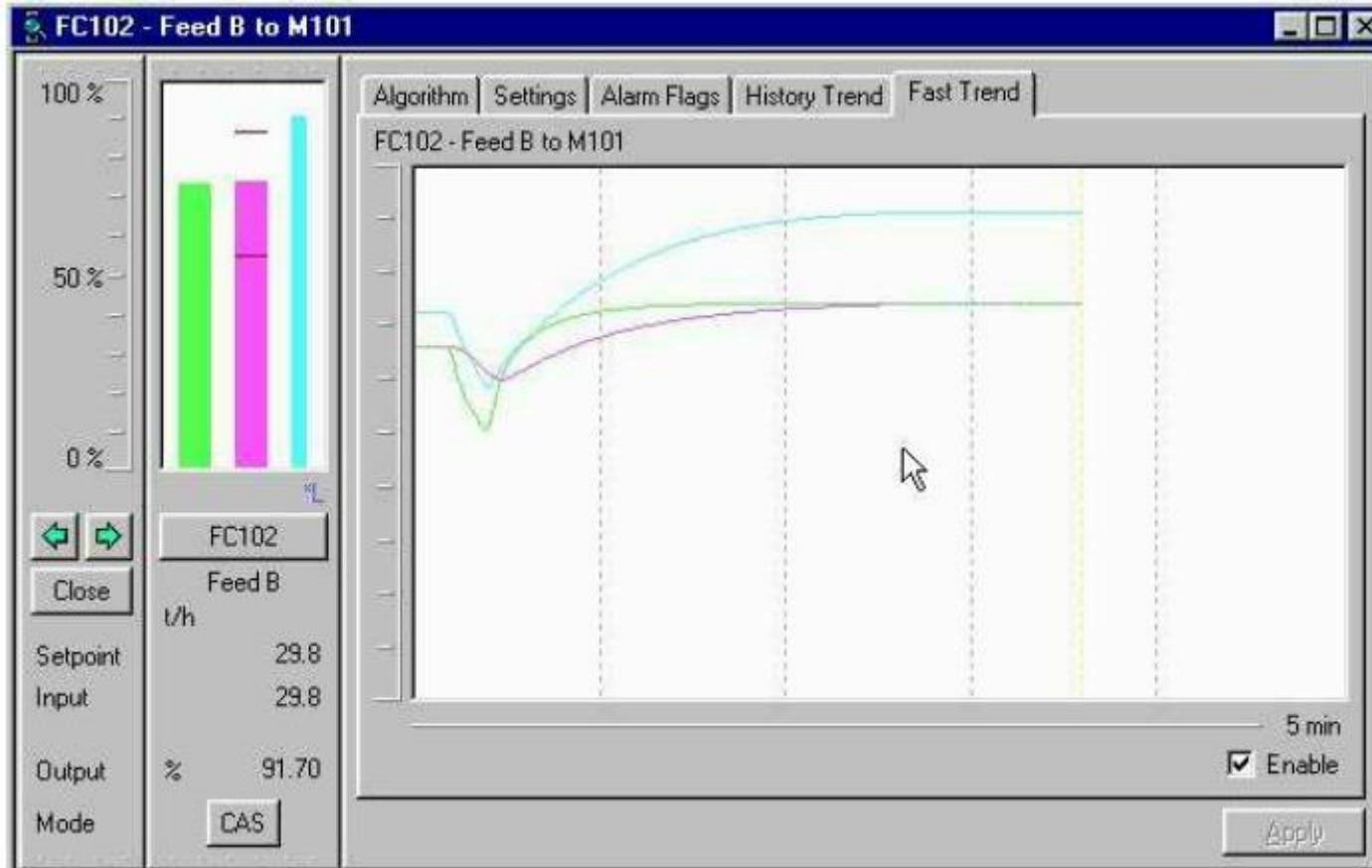


Settings

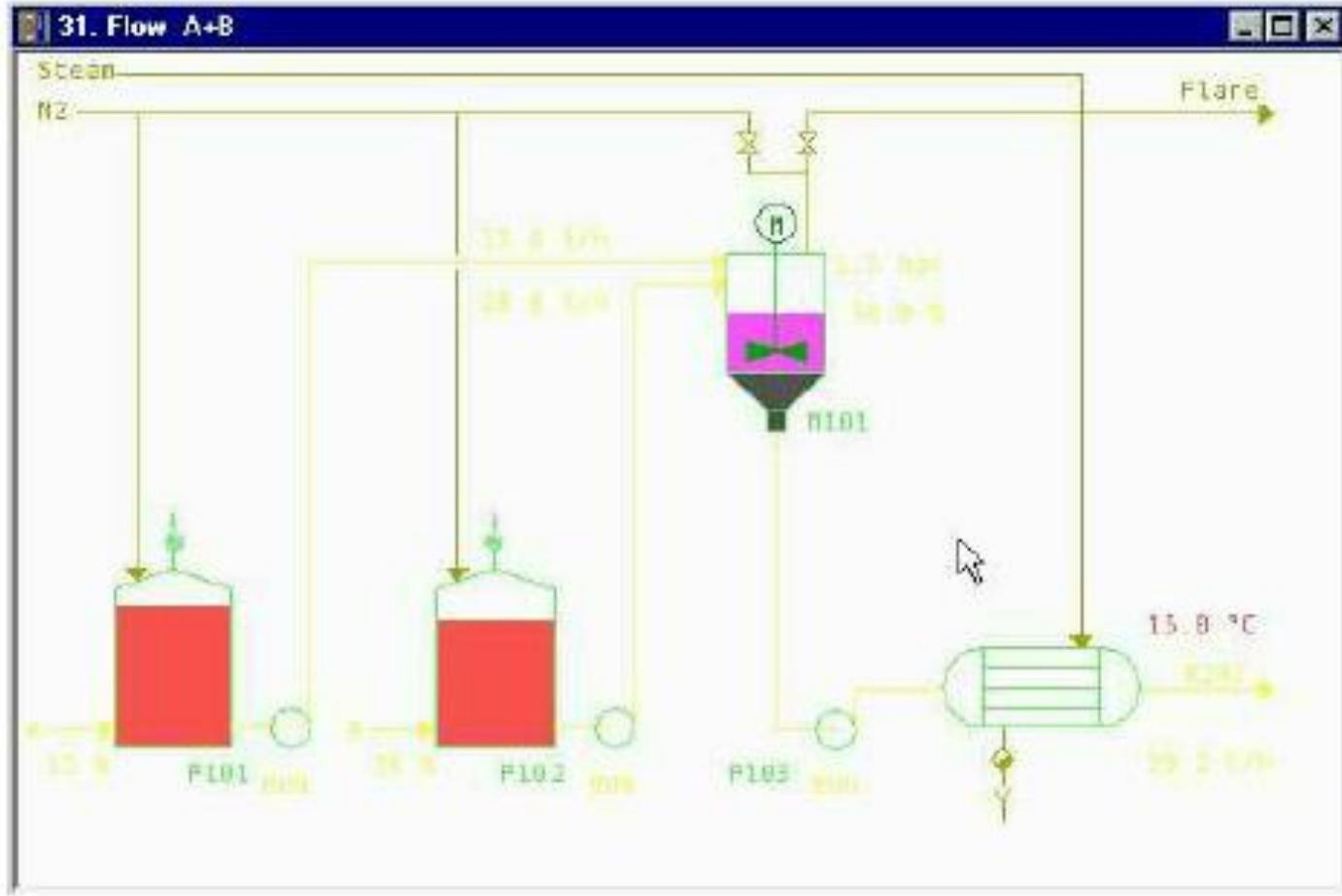
Visualisierung

Detailbild eines Reglers

Fast Trend

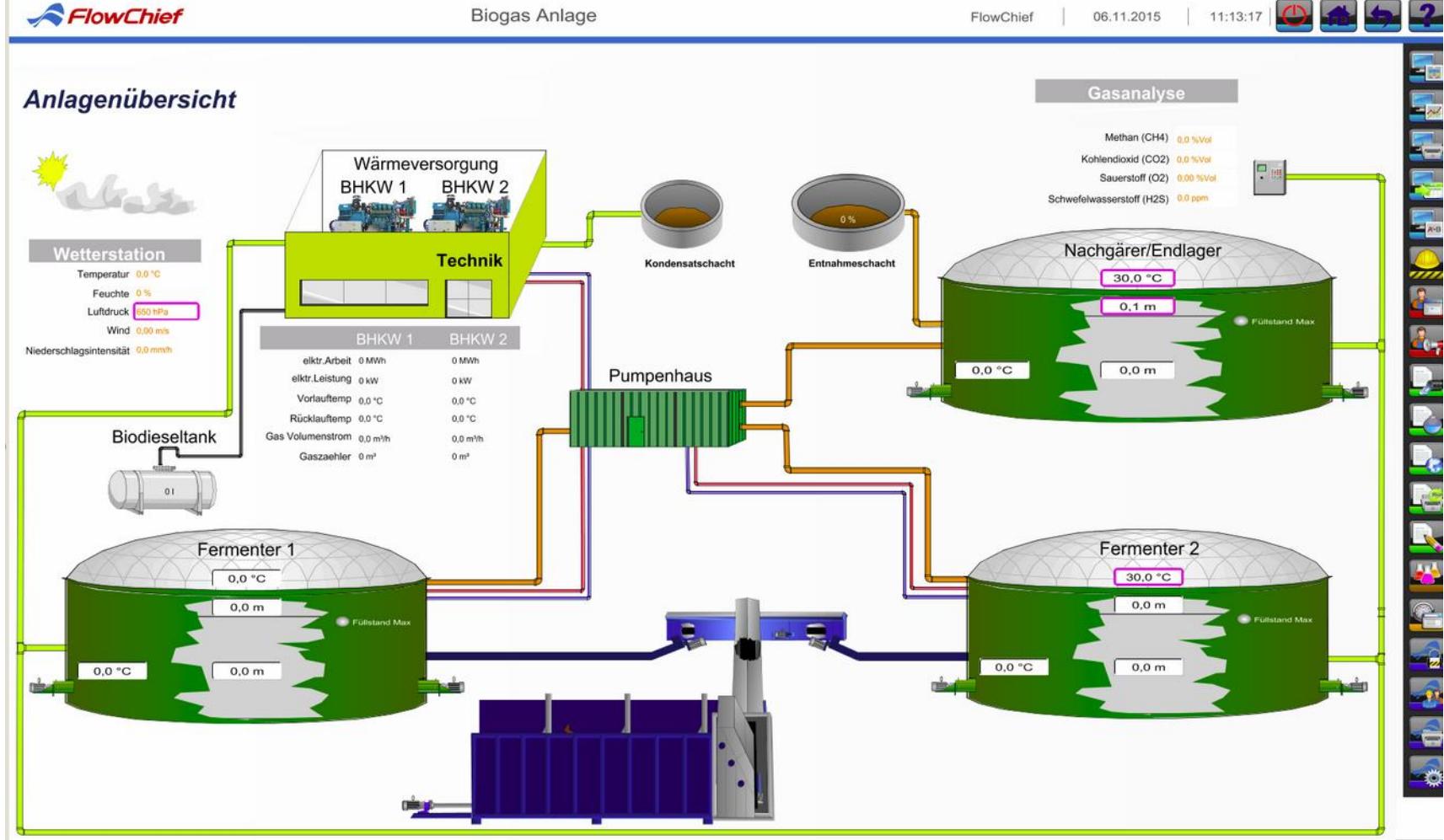


Visualisierung Übersichtsfließbild

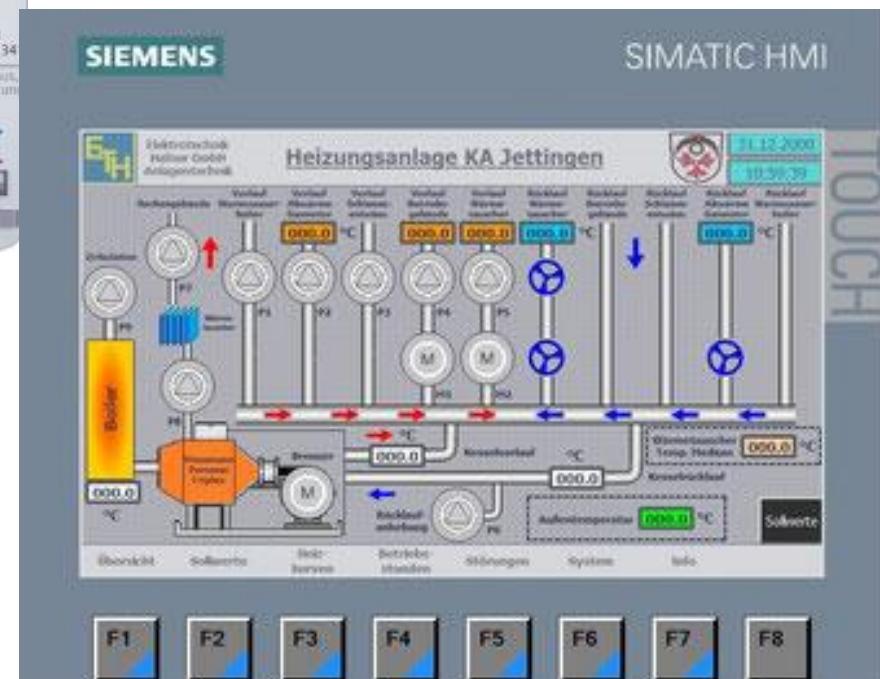
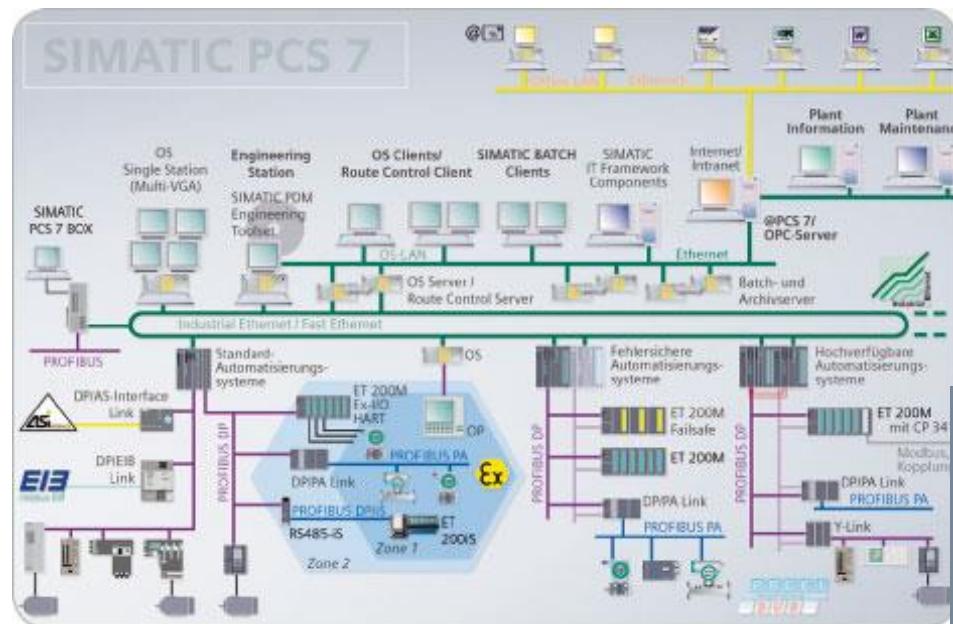


Visualisierung

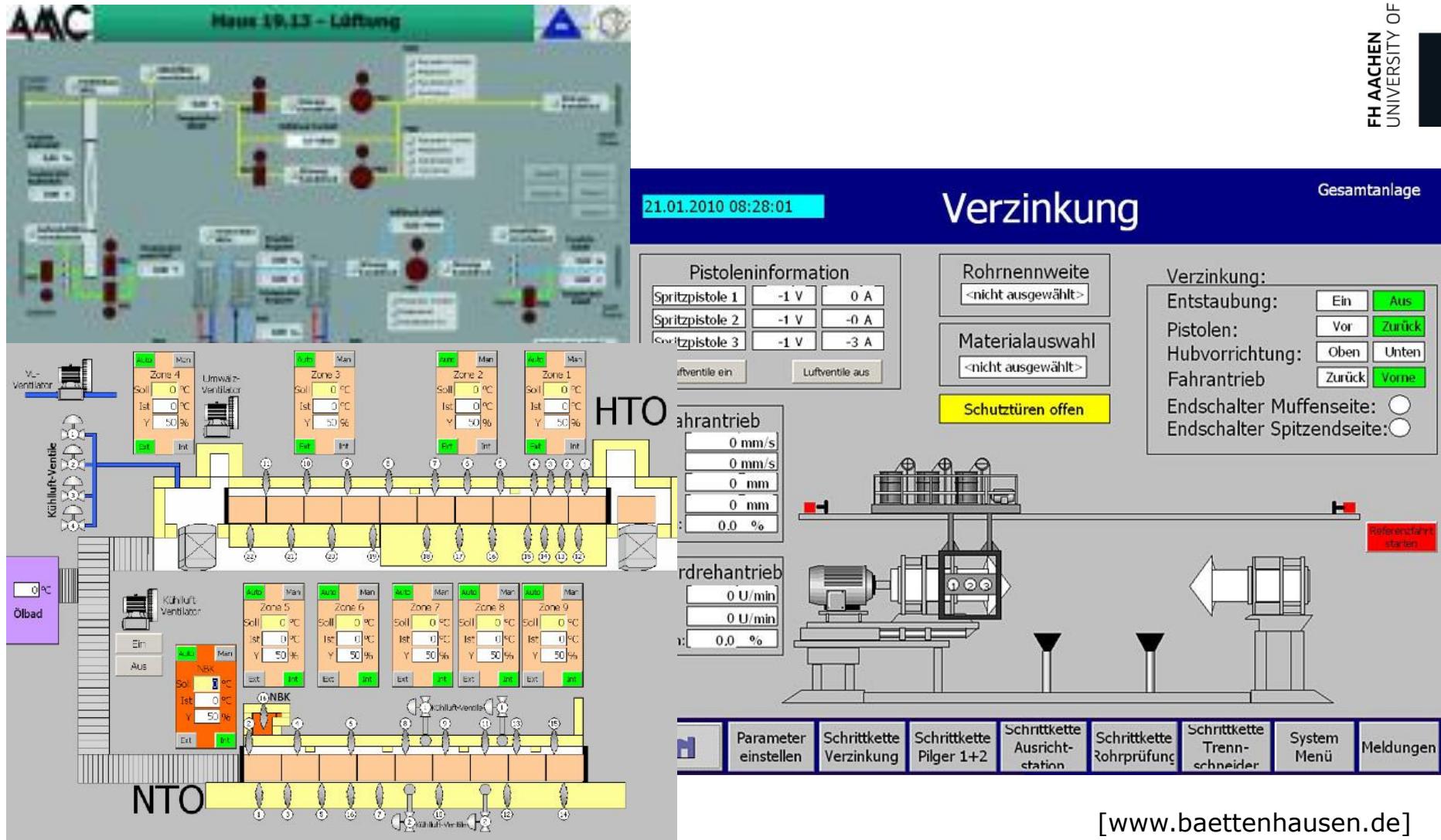
Webfähige Darstellung



Visualisierung SIMATIC

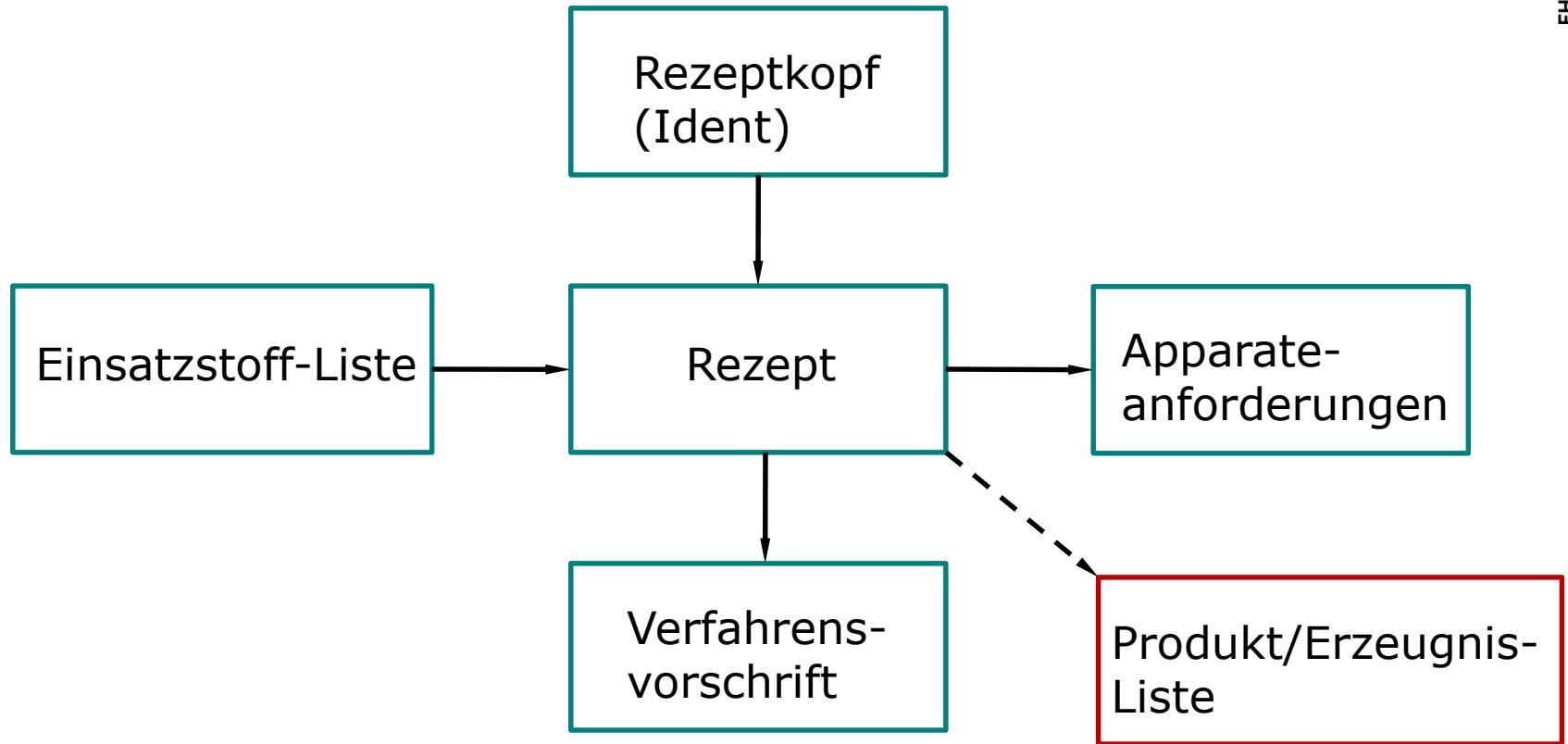


Visualisierung LabView und andere



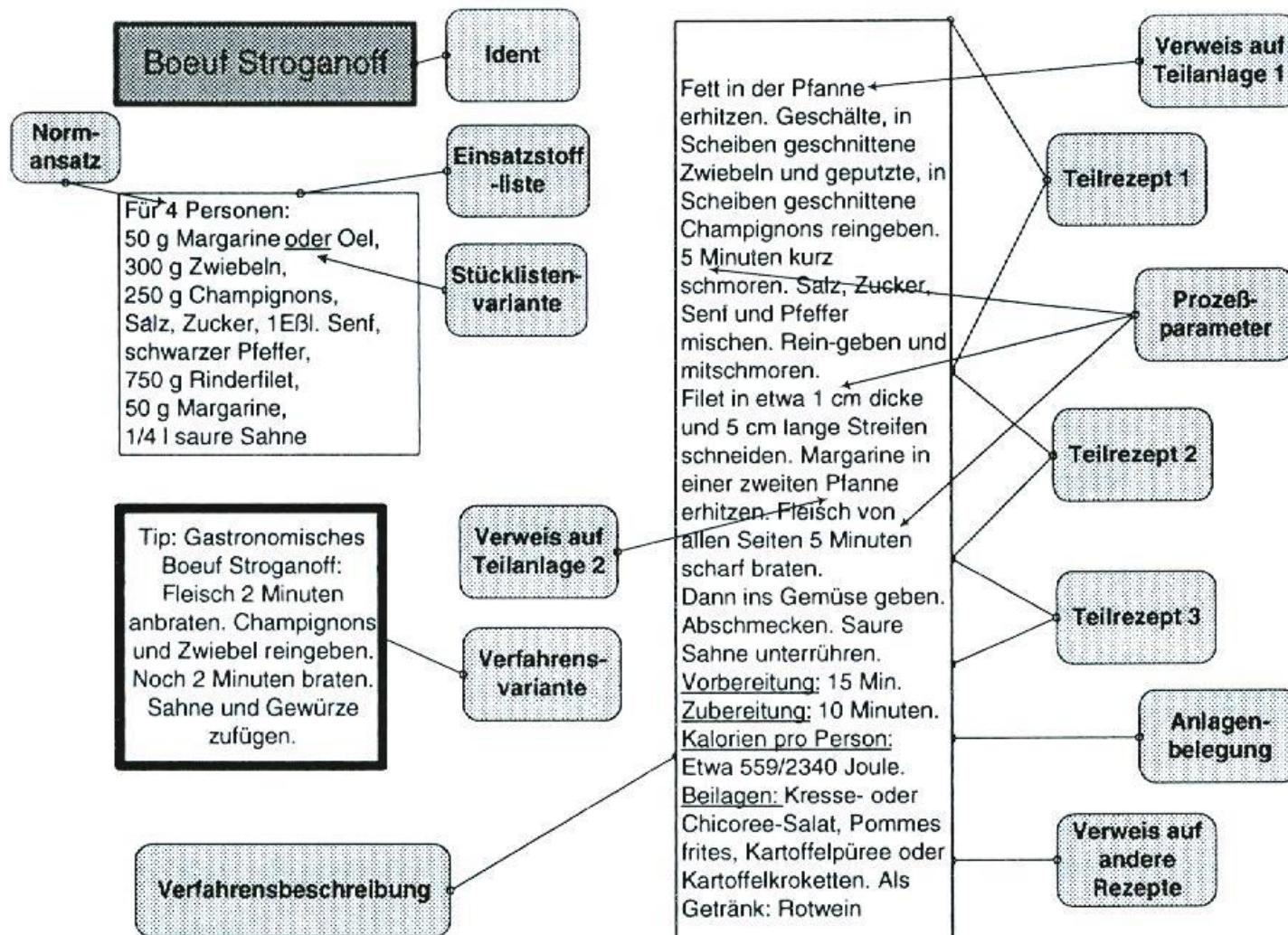
Rezepturen

Rezeptausprägungen

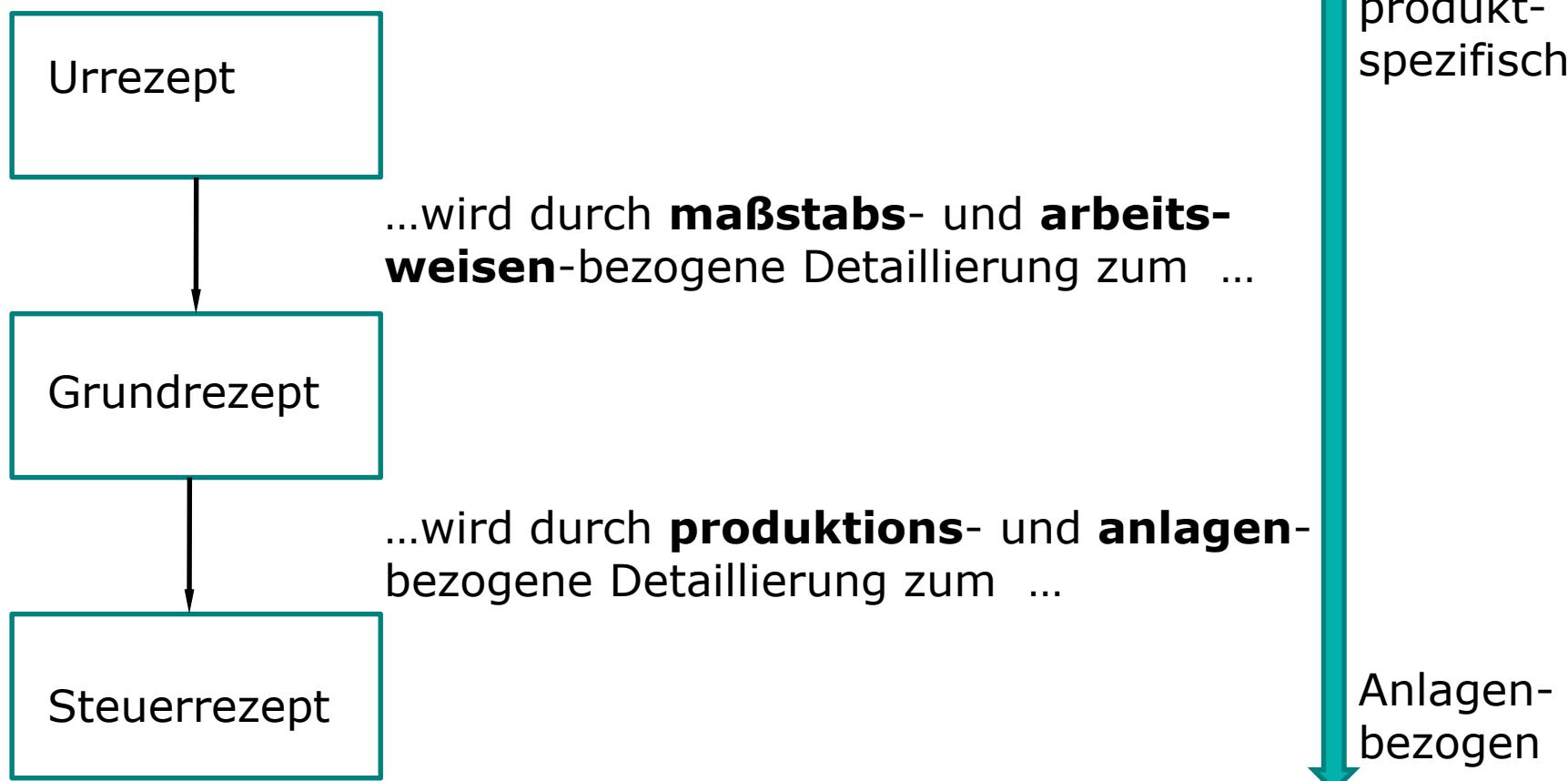


Rezepturen

Rezeptausprägungen, Beispiel



Rezepturen Rezept hierarchien (NAMUR NE 33)



Rezepturen Rezept hierarchien (NAMUR NE 33)

Urrezept (Produktidee, Labor/Forschung)

- > Einsatzstoffe, Zwischenprodukte, Erzeugnisse
- > Verfahrensabschnitte (Grundoperationen)
- > Verknüpfung der Verfahrensabschnitte durch Bilanzen (Energie, Massen)

Das Urrezept ist produkt spezifisch, nicht anlagen orientiert!

Grundrezept (Scale-Up, Technikum, betriebliche Mengen u. Apparate, Instrumentierung)

Präzisierung des Urrezepts durch

- > Handlungsanweisungen
- > Ergänzende Angaben zu Anlagen(-typen)
- > Energieformen und -bedarf
- > Los- /Chargengröße bzw. Konti-/Batchbetrieb
- > Instrumente

Das Grundrezept ist noch nicht für die Produktion gedacht!

Rezepturen

Rezept hierarchien (NAMUR NE 33)

Steuerrezept (Konkretisierung für die Produktion)

Präzisierung des Grundrezepts durch

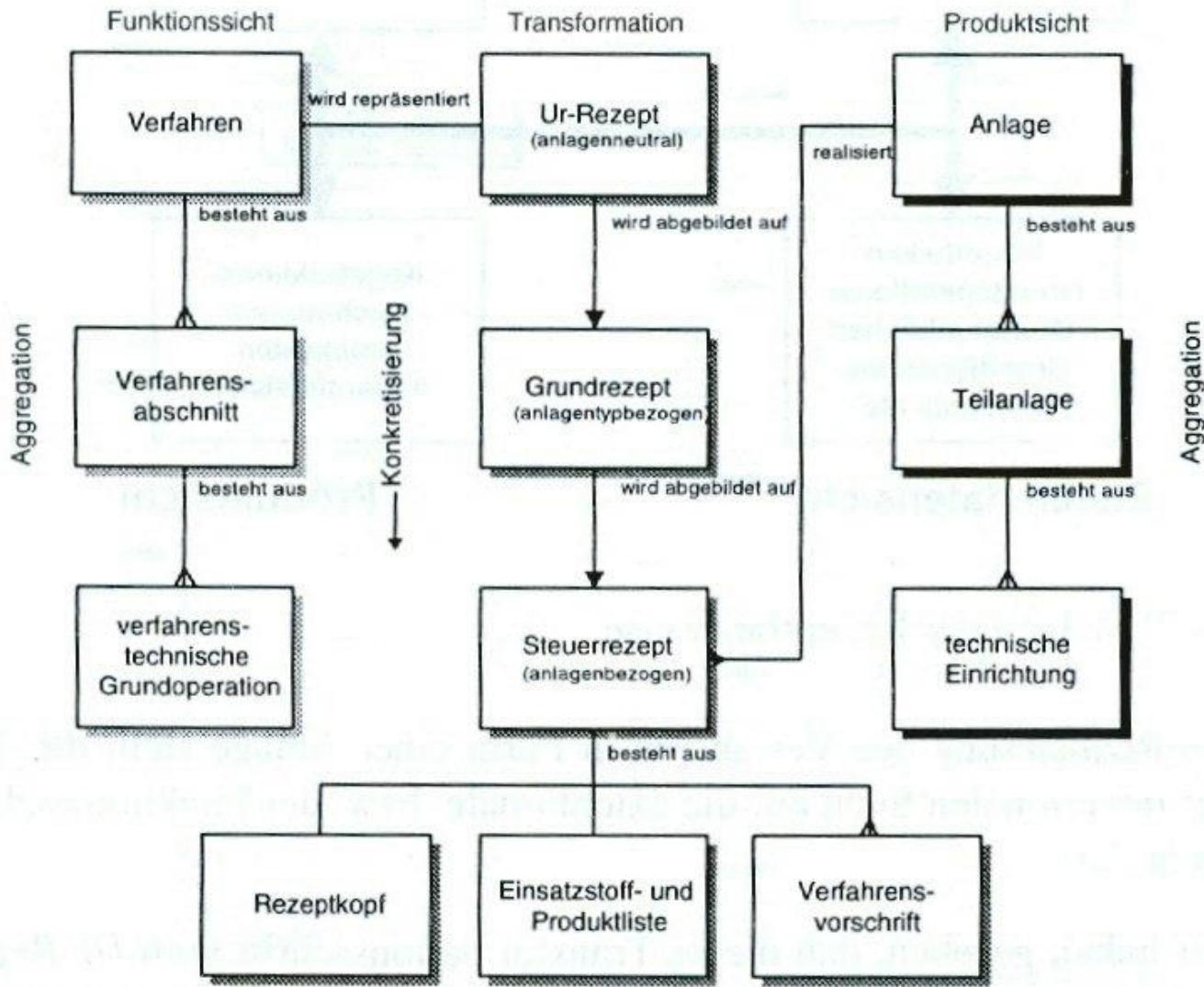
- > Laufzeiten, Steueranweisungen
- > konkrete Teilanlagen (z.B. Festbett-Reaktor R011, Mischer MX_022)
- > Fahrweise (An-/Abfahren, Vorgehen im Prod.-Betrieb, Sollwerte)
- > **Los-/Chargen-ID**

Das Steuerrezept ist maßgebend für den Herstellvorgang der Produkte!

Weitere Rezeptausprägungen und Hierarchien: DIN EN 61 512-1

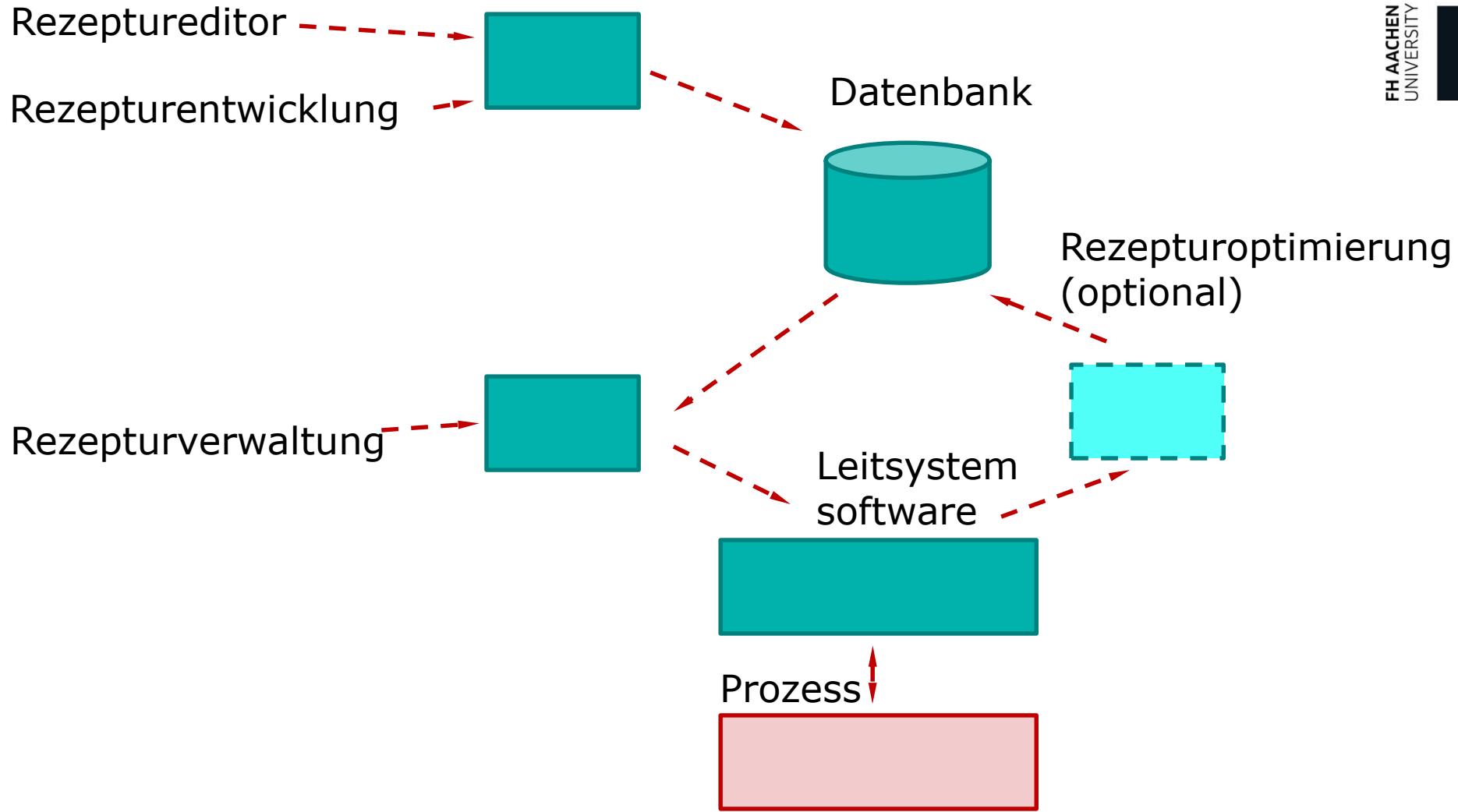
Zusätzlich „Werks-Rezept“, sonst vergleichbar mit NAMUR NE 33

Rezepturen Zusammenfassung



Rezepturen

Rechnergestützte Rezepturbearbeitung



Montageprozess

Produktanalyse I

Bezeichnung:		Zchnq.-Nr.:			Sachbearbeiter:		Datum:	
Grundsatzfragen	Einflußfaktoren	Bewertung *						
		Grundsatzfragen			Einflußfaktoren			
1. Preis des Einzelteiles	Stoffauswahl	1	2	3	1	2	3	
Herstellkosten	Herstellkosten							
Losgröße	Losgröße							
2. Auslieferungs-zustand	Schüttgut							
	Geordnet verpackt							
	Magaziniert							
	Langteil -Fließgut-							
	Gegurtet							
3. Handhabungs-fähigkeit	Ordnungsfähigkeit							
	Weitergeben							
4. Fügerichtung	Fügerichtung							
Fügefähigkeit	Fügefähigkeit							
	Fügeräume							
	Stabilität							
5. Fügeverfahren	Zusammenlegen							
	An- und Eindrücken (Schrauben)							
	Urformen							
	Umformen							
	Stoffvereinigen							
6. Qualität	Teilequalität (AQL)							
	Fremdkörperanteil							
	Sauberkeit							
	Störbetriebskosten							
Sonstiges								

- > für Basis- und jedes Montageteil
- > Produktstruktur

Handhabungs-, Fügefähigkeit [LOTTER]

Montageprozess

Produktanalyse II

Teil-Nr.	Bezeichnung	Anzahl pro Produkt	Anlieferungs-zustand	Handhabungs-eigenschaften	Qualitäts-anforderung	Bemerkung
1	Sockel	1	lagegeordnet verpackt	●	○	manuelle Ersthandschaltung
2	Vierkantmutter	1	Schüttgut	○	○	
3	Wurmschraube	1	Schüttgut	○	○	
4	Sechskantmutter	5	Schüttgut	○	○	manuelle Handhabung
5	Schnappelement	1	Schüttgut	●	○	manuelle Eingabe
6	Schraube	1	Schüttgut	○	○	
7	Gegenkontakt	1	Schüttgut	○	○	
8	Klemme	4	Schüttgut	○	○	
9	Schraube	4	Schüttgut	○	○	
10	Schraube	1	Schüttgut	○	○	
11	Anschlußfahne	3	Schüttgut	●	●	
12	Stein	1	Schüttgut	○	○	
13	Deckel	1	Schüttgut	○	○	
14	Niet	3	Schüttgut	○	○	Sauberkeit
15	Niet	2	Schüttgut	○	○	Sauberkeit Fremdkörperan.
16	Haube	1	Schüttgut	○	○	
17	Bimetall	1	Schüttgut	○	○	Nachsortierung
18	Stift	1	Schüttgut	○	○	
19	Justierschraube	1	Schüttgut	○	○	
20	Schraube	3	Schüttgut	○	○	
21	Sicherungslack	1	Tube	○	○	

> ggf.
montagegerechte
Umgestaltung

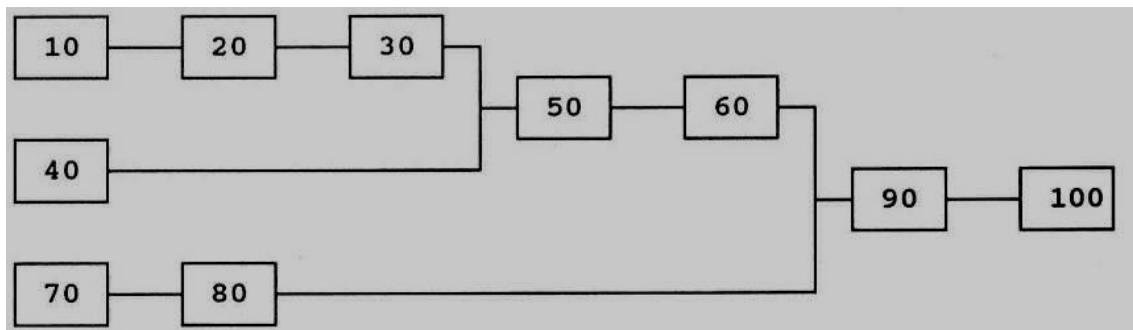
Handhabungs-, Fügefähigkeit [LOTTER]

Montageprozess

Ermittlung der Montagestationen

Motorbaugruppe						
Montageplan: _____		Agf	K Arbeitsgangbeschreibung	wird	te	tr
10	V	Motoren bereitlegen	A1	.	.	
20	F	Dichtung abziehen, oben auf Motor kleb.	A2	.	.	
30	F	Motor auf Dichtung unten aufdrücken	A3	.	.	
40	V	Motorträger bereitlegen	B1	.	.	
50	F	Motor aufsetzen	A4	.	.	
60	F	Auskleidung einsetzen	A5	.	.	
70	V	Motorabdeckung bereitlegen	C1	.	.	
80	F	2 Schaumstoffstücke in Motorabd. einkl.	C2	.	.	
90	F	Motorabd. aufsetzen, Litzen durchführen	A6	.	.	
100	F	Motorabd. mit Motorträger 4x verschr.	A7	.	.	

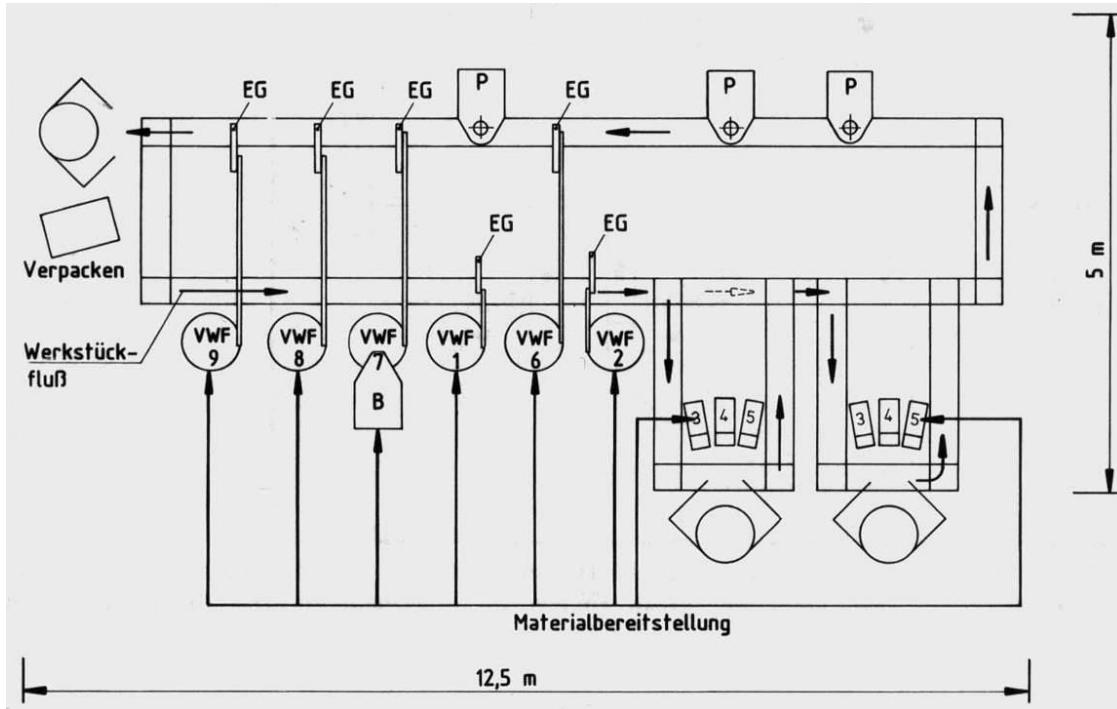
- > Auflistung der Fügevorgänge
- > Ermittlung der benötigten Montagestationen



Fügerangfolge [Scharf, P.]

Montageprozess

Layout der Montageanlage I



Layout Montageanlage 1 [Lotter]

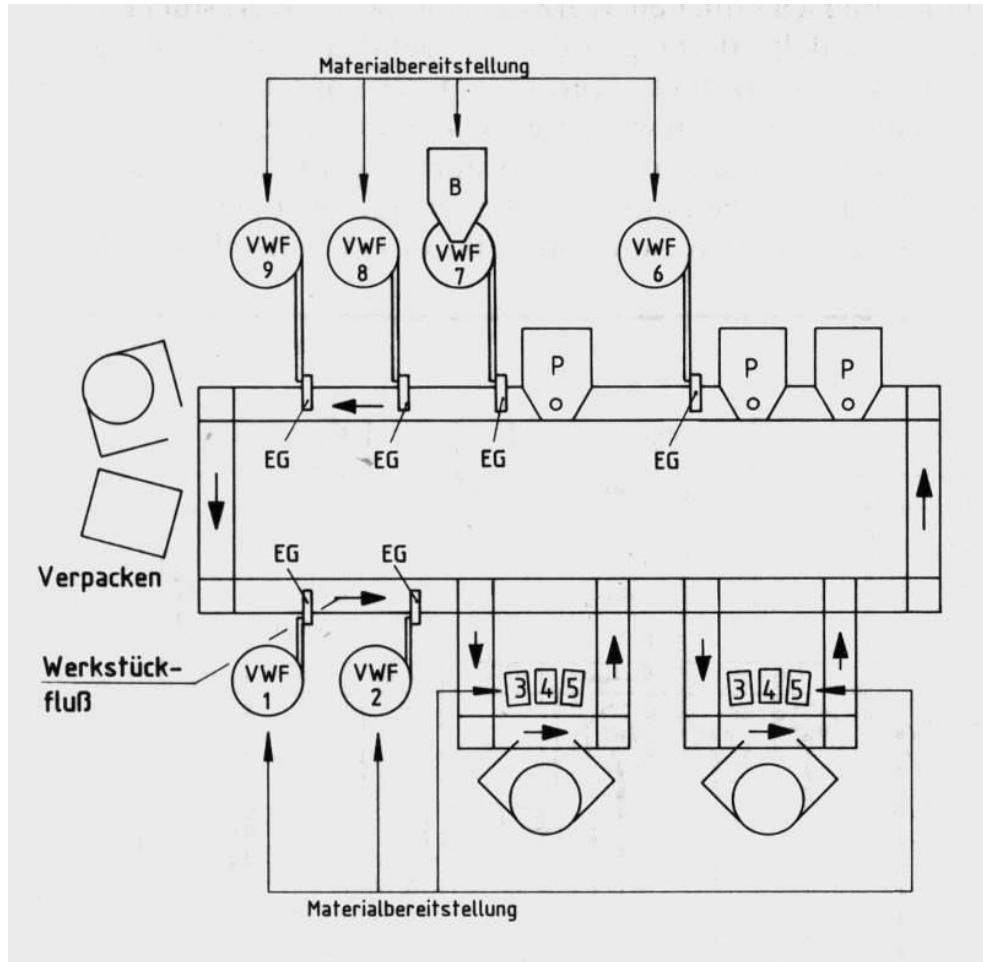
EG = Einlegegerät, VWF = Vibrationswendelförderer, P = Presse,

B = Bunker, 1 bis 9 = Teilenummern

- > **Zuführprozesse**
Magazin, VWF
- > **Fügeprozesse** Art,
manuell/automatisch
- > **Aufteilung auf
Montagestationen**
Fügerangfolge, Taktzeit,
Zugänglichkeit,
Fügerichtung,
Qualitätssicherung
- > **Puffer, Verzweigungen**
- > **Verkettung** Taktung,
Rund-, Längstransfer, WT
- > **Alternativen**

Montageprozess

Layout der Montageanlage II



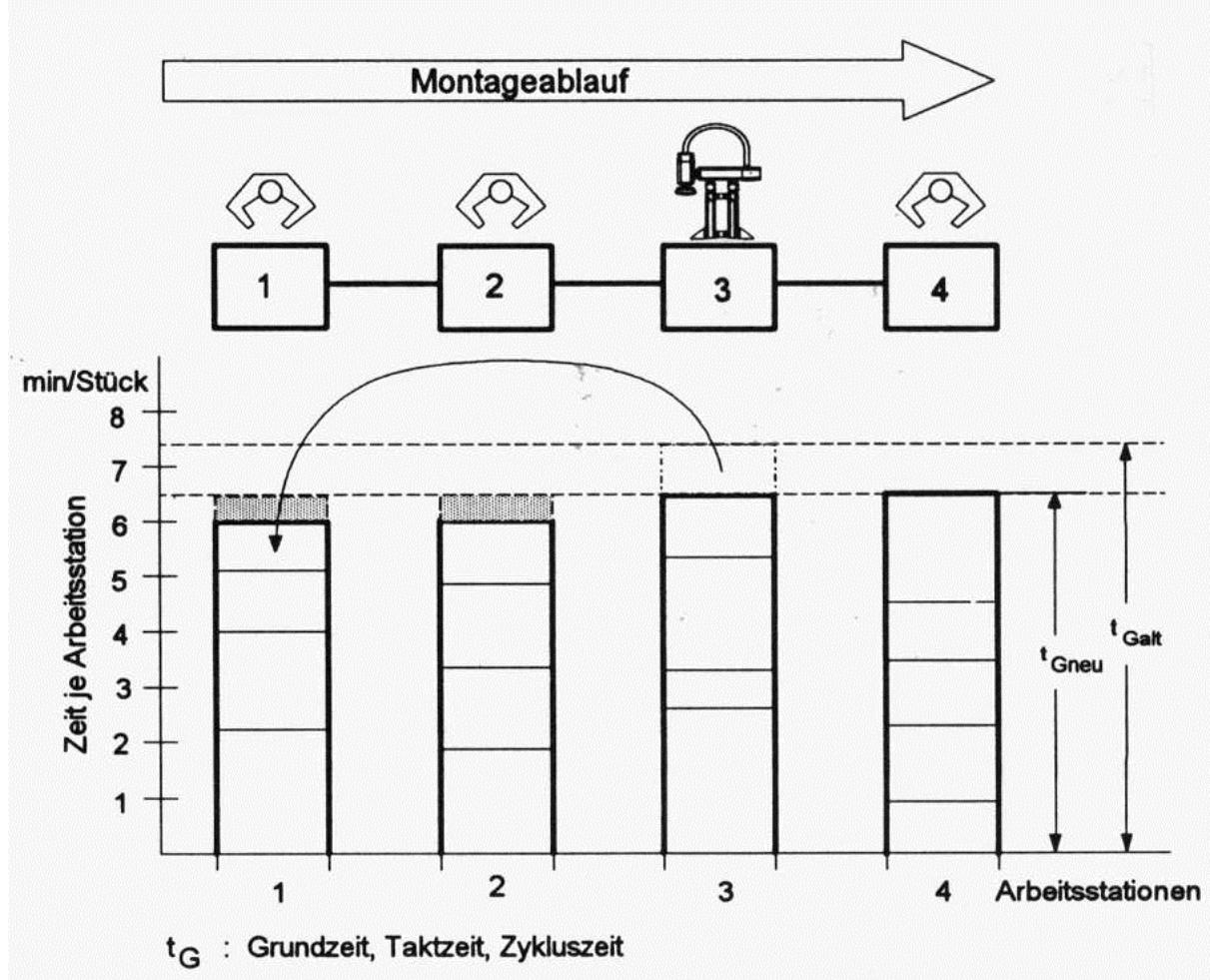
Layout Montageanlage 2 [Lotter]

Wichtig:
Sekundär-Montagezeiten wie Handhaben, Prüfen und Transportieren sind zu minimieren!

Nur Fügen ist Primärmontage und damit wertschöpfend!

Montageprozess

Layout der Montageanlage III



Auch mögliche unterschiedliche Taktzeiten haben Einfluss auf das Layout.

Anlagenplanung und Optimierung

Planungssoftware I

Anlagenplanung und Optimierung

Planungssoftware II



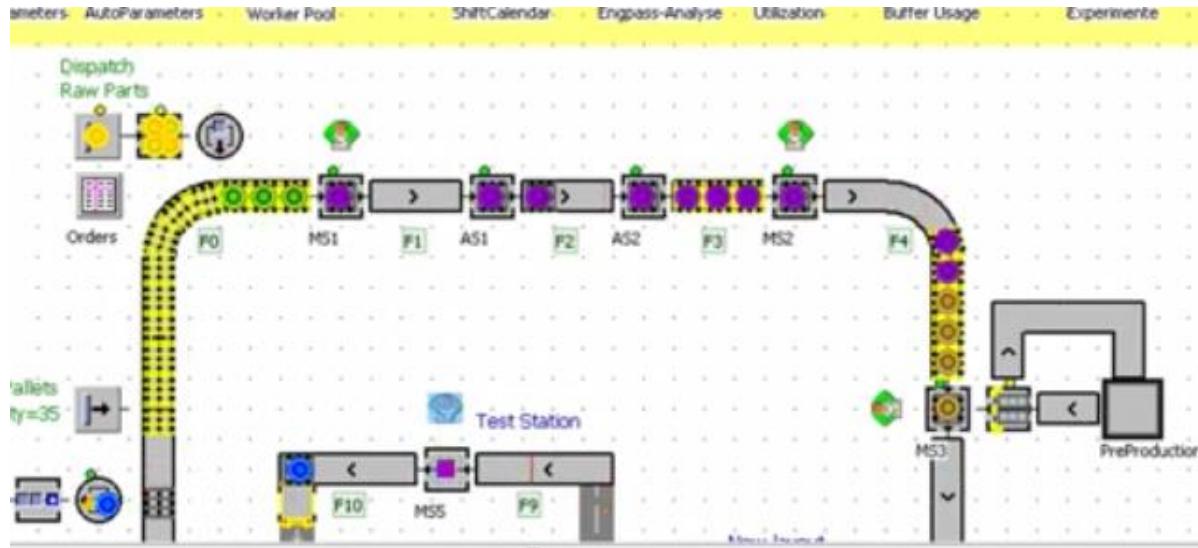
Virtual Desktop MCAD News, Video Edition: August 20, 2012

**In This Issue: Simulating plant operations with
Siemens PLM Software's Plant Simulation**

Kenneth Wong, Senior Editor
DESKTOP ENGINEERING
www.deskeng.com

Anlagenplanung und Optimierung

Planungssoftware III



- > Layout-Generierung
- > Minimierung von Puffergrößen, Anzahl der Werkstückträger
- > Optimierung des Materialfluss
- > Überprüfung der Taktzeit und Durchlaufzeit unter Berücksichtigung von zufälligen Einflüssen, Abschätzung des Datenvolumens,
- > Digitale Fabrik z.B. mit

Tecnomatix

[Siemens]

CIROS Studio

[Festo]

FMSSoft/MTPro

[Bosch Rexroth]