Kurs OMSI
im WiSe 2014/15

Objektorientierte Simulation mit ODEMx

Prof. Dr. Joachim Fischer
Dr. Klaus Ahrens
Dr. Markus Scheidgen
Dipl.-Inf. Ingmar Eveslage

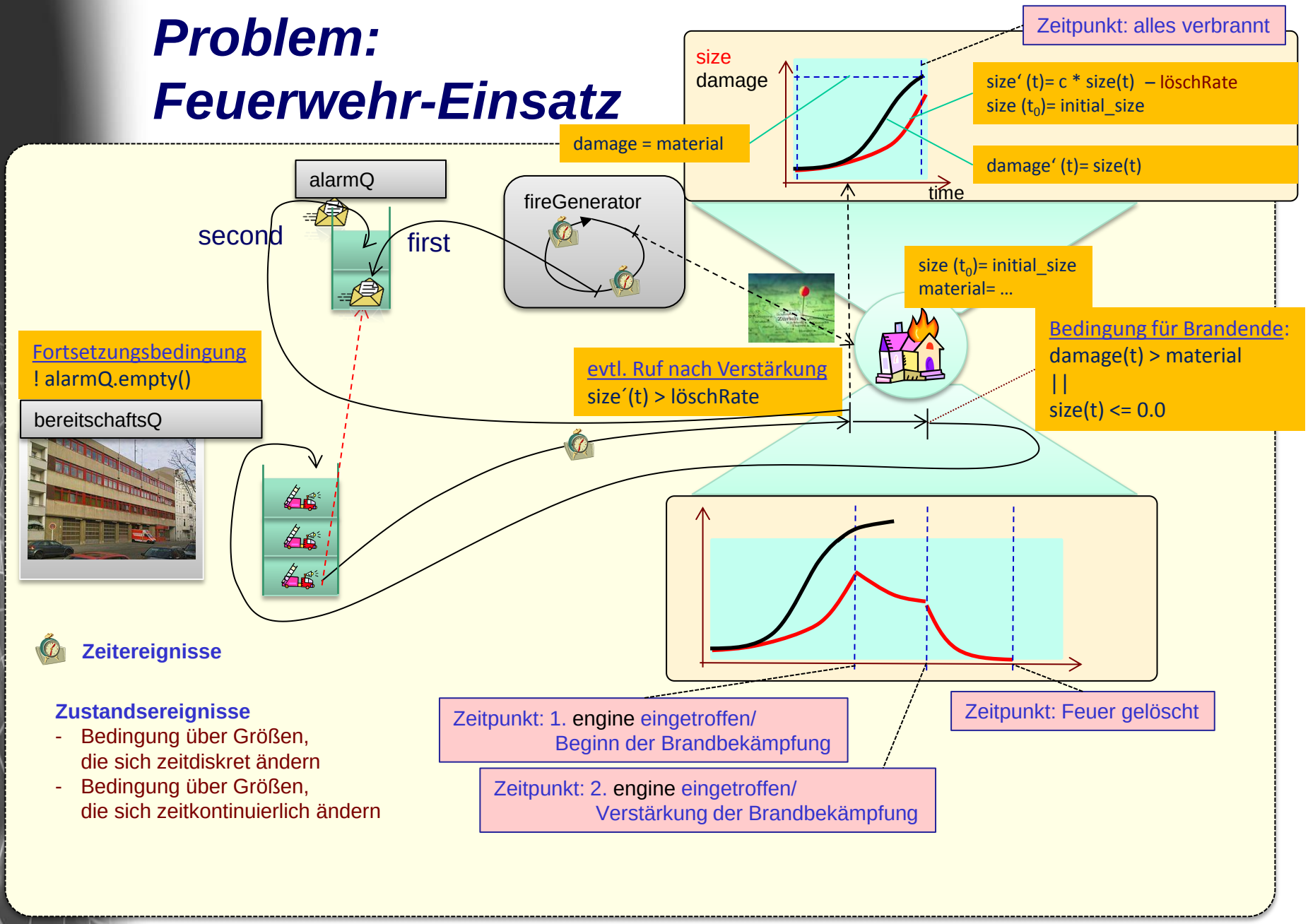
fischer|ahrens|eveslage@informatik.hu-berlin.de

8. Ausblick:

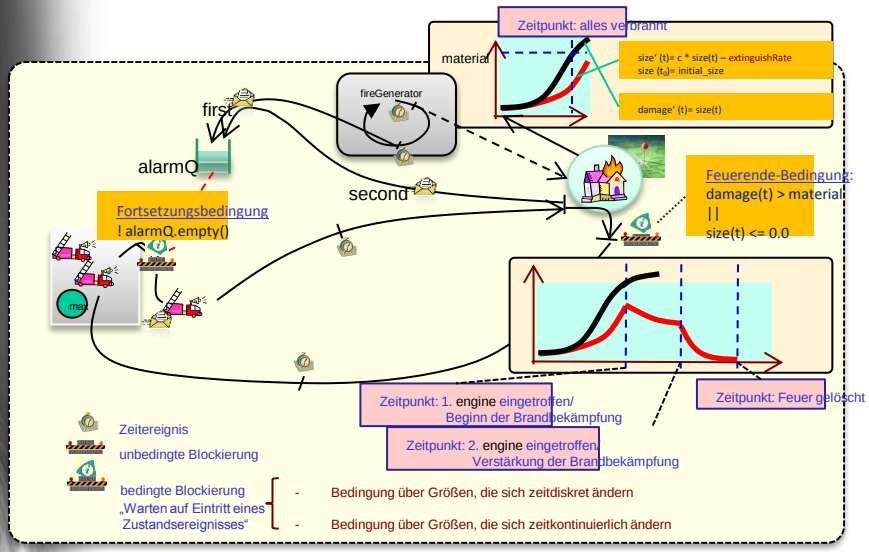
Behandlung zeitkontinuierlicher Zustandsänderungen

- Beispiel: Feuerwehreinsatz
- Beispiel: Chemische Reaktoren
- Konzept für die zeitkontinuierliche Simulation
- Chaotische Systeme
- Weitere Beispiele

Problem: Feuerwehr-Einsatz

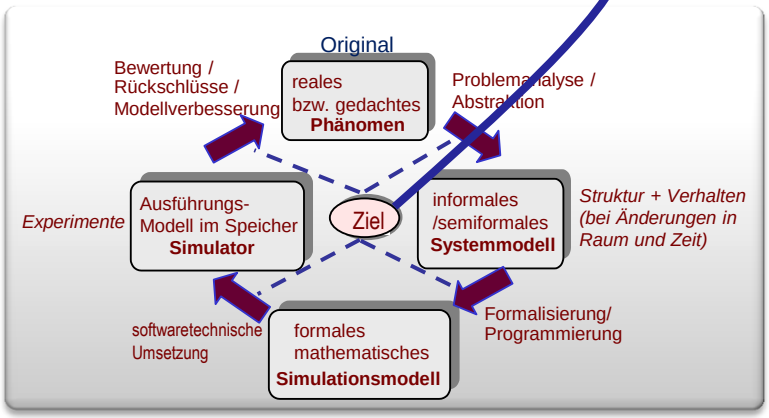


Informales → Formales System-Modell



mögliche Zielstellung ~ Untersuchung

- zur Anzahl und Ausstattung der Feuerwehrstationen
- zum Strategie-Vergleich
  (1 oder 2 Fahrzeuge als Ersteinsatz)
- Einsatz von WasserTanks in der Stadt



OO- Modell
- Vorzüge
- Abstraktionsparadigmen

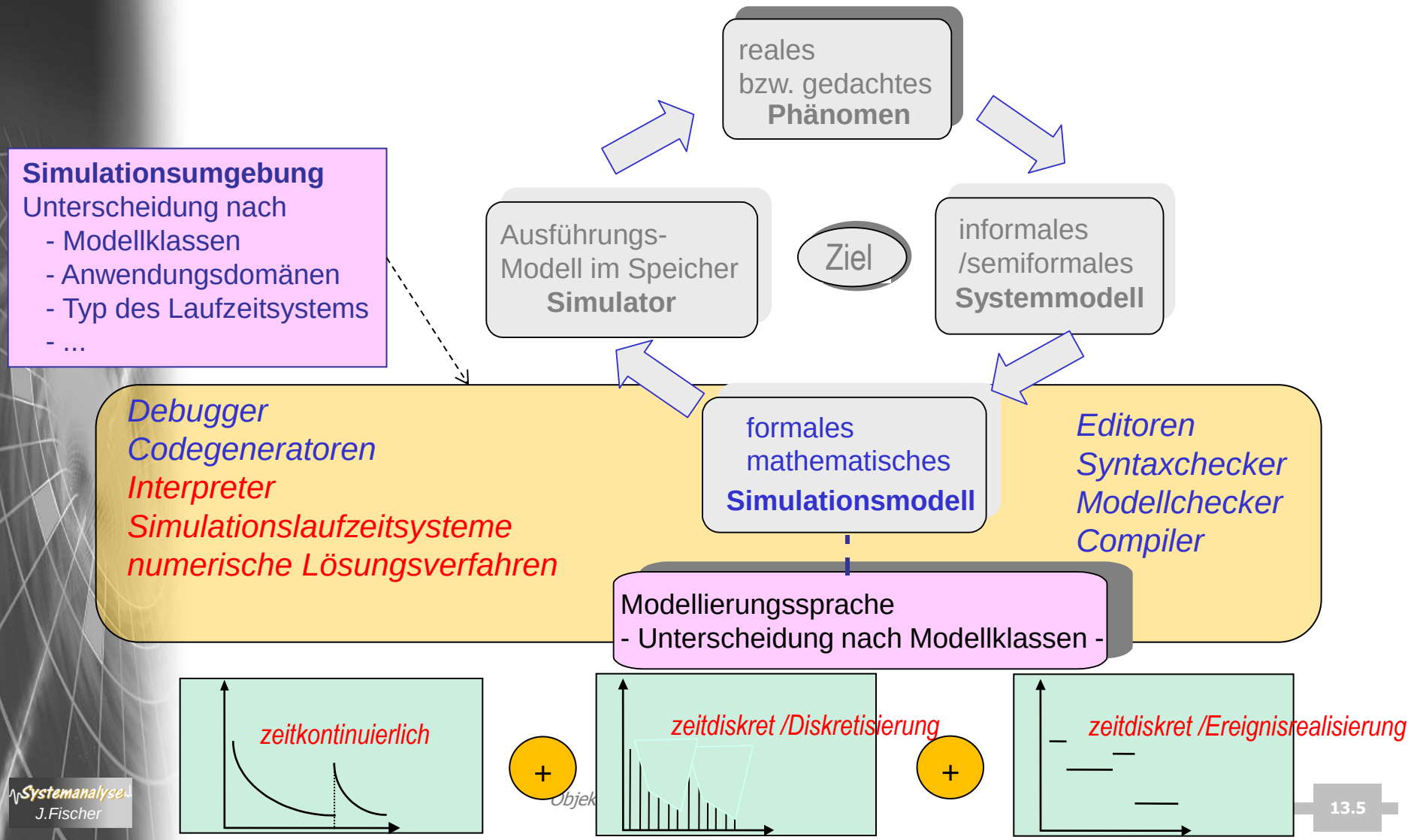
Struktur
Modellierung

Verhaltensmodellierung
Modellierung

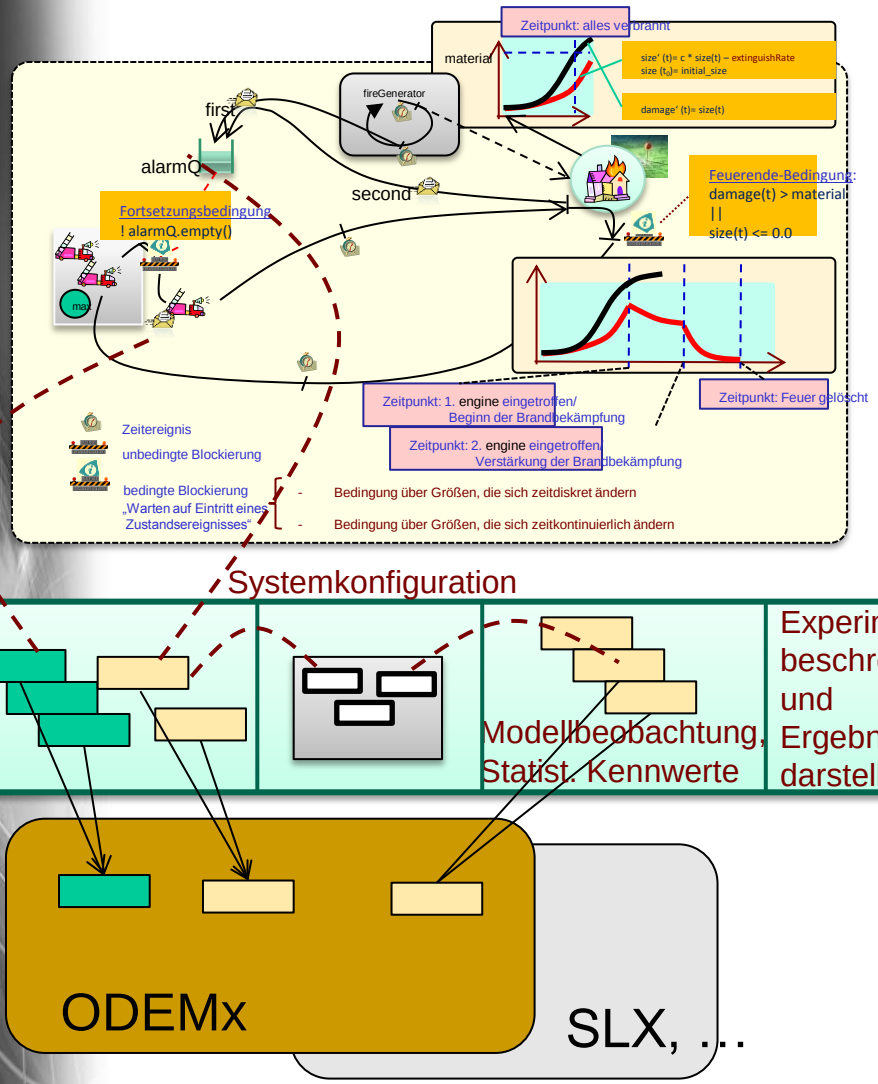
Math. Modellklasse
Verhaltensbeschreibung

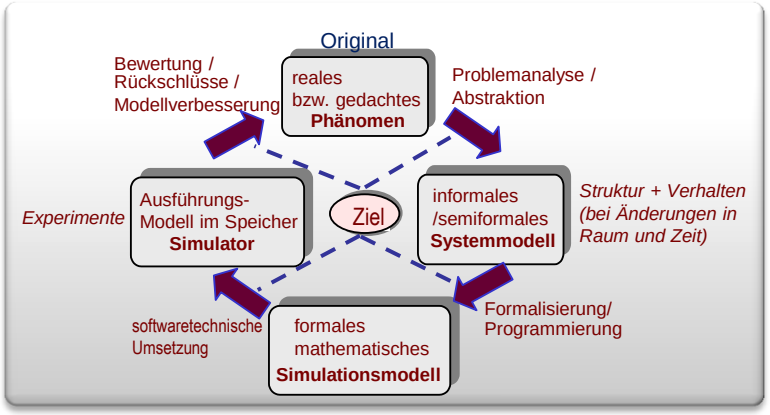
Modellierungssprachen und Simulationsumgebung

Zustandsänderungen kontinuierlich oder/und diskret in Raum und Zeit



Benötigte Konzepte





benötigte Konzepte

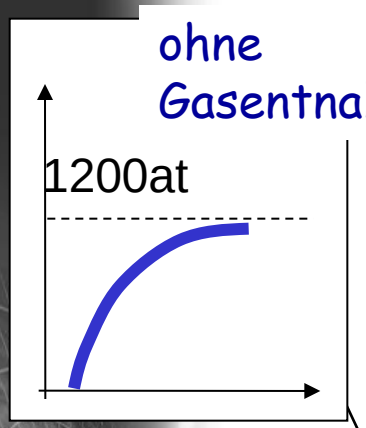
- diskrete Prozesse, Zustandsereignisse
- Kontinuierliche Prozesse, Zustandsereignisse
- Port-Synchronisation (Alarm und Engine)
- CondQ-Synchronisation (Rückfahrt von Engine)
  (**aber**: wer gibt das Signal?)
- Zufallsgrößen
  - Hausbrand: (Zeitpunkt der Enstehung,
    Position, Anfangsgröße,
    Materialwert, Materialkoeffizient
    Zeitpunkt der Entdeckung)
  - Fahrtzeiten der Feuerwehr
- Tally/Count

8. Ausblick:

Behandlung zeitkontinuierlicher Zustandsänderungen

- Beispiel: Feuerwehreinsatz
- Beispiel: Chemische Reaktoren
- Konzept für die zeitkontinuierliche Simulation
- Chaotische Systeme
- Weitere Beispiele

Wortmodell: Simulation chemischer Reaktoren



Ausgangsstoff + Gas → Endprodukt + Ausgangsreststoff

Umsetzung erfolgt optimal nur bei bestimmten Druck- und Temperaturverhältnissen

Chemische Reaktion:
- Ausgangsstoff und
- Endprodukt im chem. Gleichgewicht

Ausgangsstoff
(initiale Befüllung)

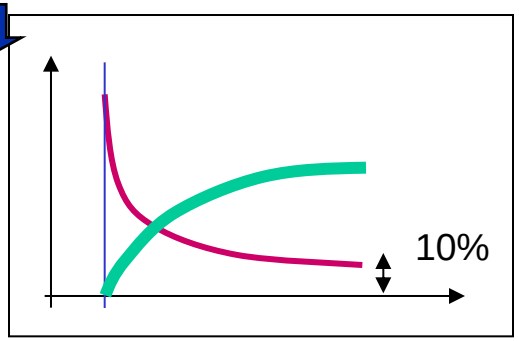
Gas

Kompressor

95at- 1200at

Gastank

100at



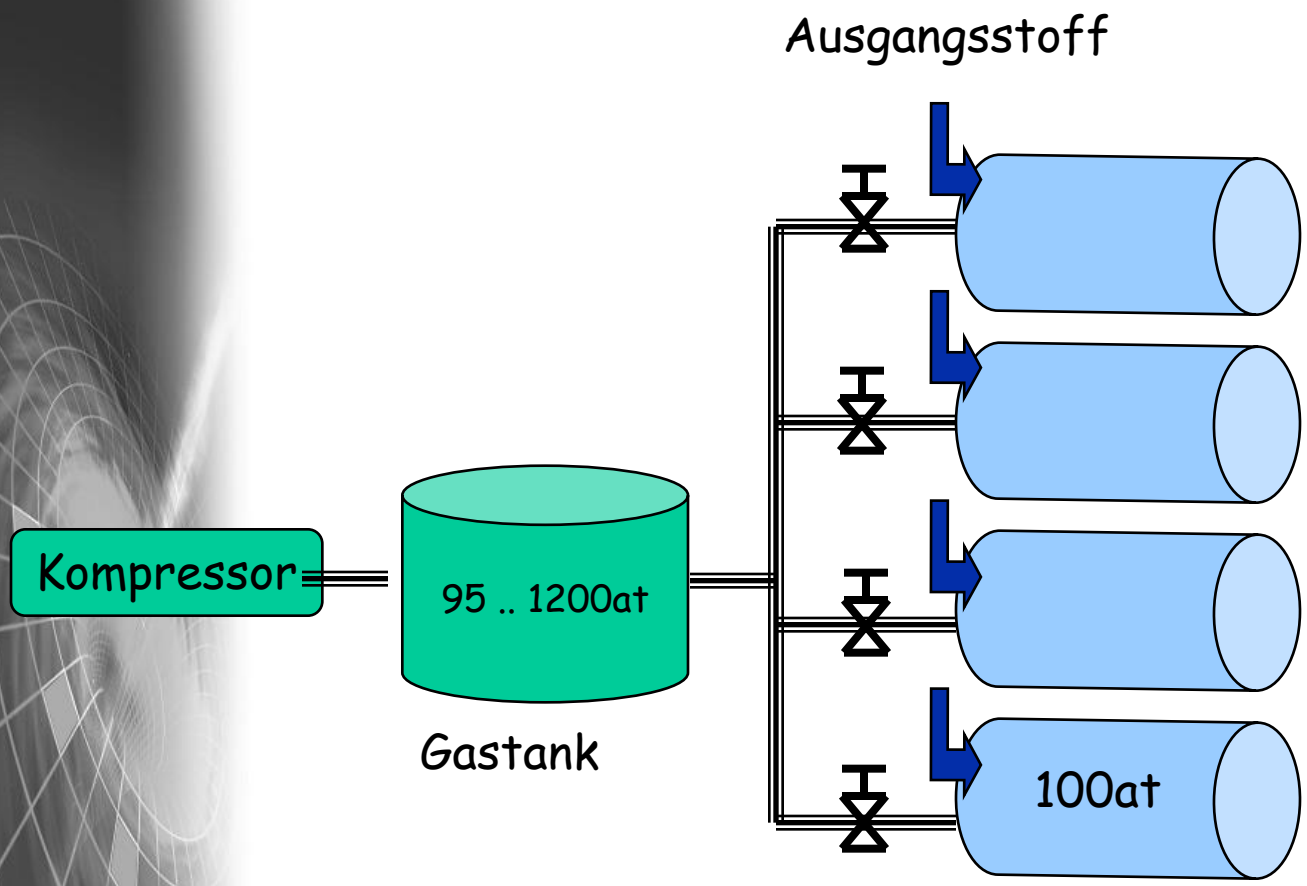
Reaktor

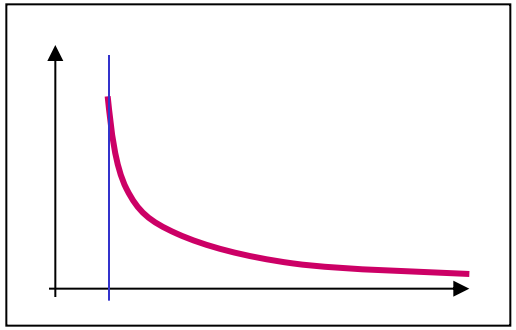
Endprodukt

Ventile drosseln den Druck

100 at konstant: wäre optimal
tatsächlich kann er aber kleiner werden, was verhindert werden soll

Wortmodell: Simulation chemischer Reaktoren



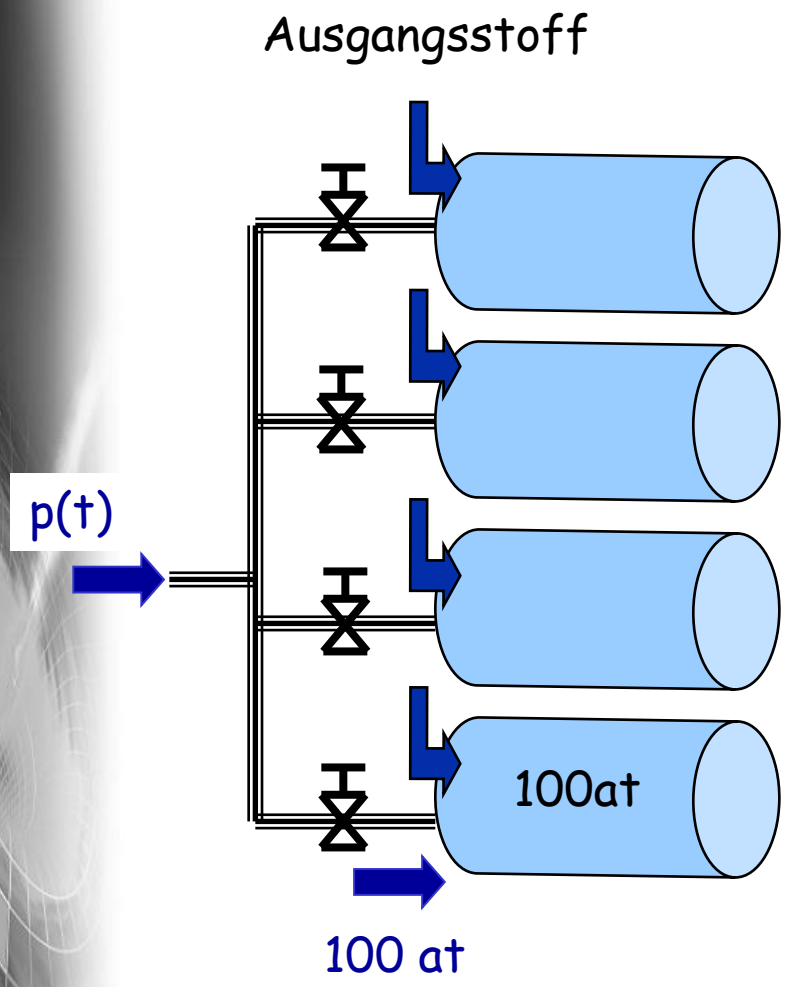


Gasverbrauch nimmt im Laufe der chemischen Reaktion exponentiell ab

Kompressor kann Druck im Tank wieder aufbauen

gleichzeitiger Start aller Reaktoren sollte vermieden werden
➔ zeitversetzte Inbetriebnahme (Öffnen des Ventils)

Wortmodell: Simulation chemischer Reaktoren



zeitversetzter zyklischer Ablauf je Reaktor:

(1) Beschickung mit Ausgangsstoff

(2) Öffnung des Ventils (100at):
Start der chemischen Reaktion

(3) Sperrung des Ventils:
Stopp der chem. Reaktion
sobald sich ein chem. Gleichgewicht einstellt (ist der Fall, wenn nur 10% des Ausgangsstoffes messbar ist)

(4) Leerung des Reaktors

(5) Säuberung des Reaktors

Zustandsereignis
sollte der Gastankdruck p(t) unter 100 at sinken, soll immer der zuletzt in Betrieb genommene Reaktor abgeschaltet werden

Ziel: Arbeitsweise nachbilden, Arbeitsplanerstellung für Bediener

Mathematisches Verhaltensmodell

Änderung der Konzentration c
des Ausgangsproduktes im i-ten Reaktor:

$$c_i'(t) = -k_i * c_i(t) * peff(t)$$

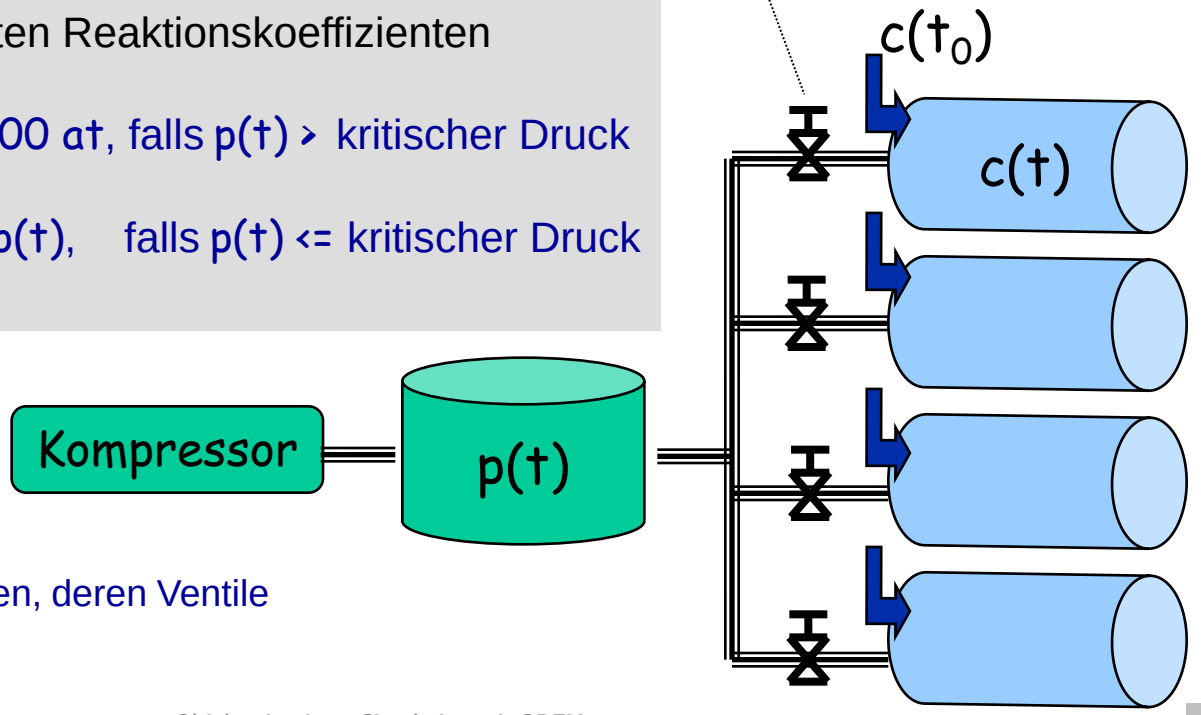
$$c_i(t_0) = C_i \quad \text{für } i = 1,2,3,4$$

mit
k_i als konstanten Reaktionskoeffizienten
und

$$peff(t) = \begin{cases} 100\ at, & \text{falls } p(t) > \text{kritischer Druck} \\ p(t), & \text{falls } p(t) <= \text{kritischer Druck} \end{cases}$$

für alle Reaktoren, deren Ventile
geöffnet sind

Ventil drosselt den Druck
100 at: wäre optimal
real kann er kleiner sein



Mathematisches Verhaltensmodell

Änderung des Gasdruckes im Tank:

$$p'(t) = 107.03 * (vomKompressor - zuReaktoren(t))$$
$$p(0) = 500$$

mit

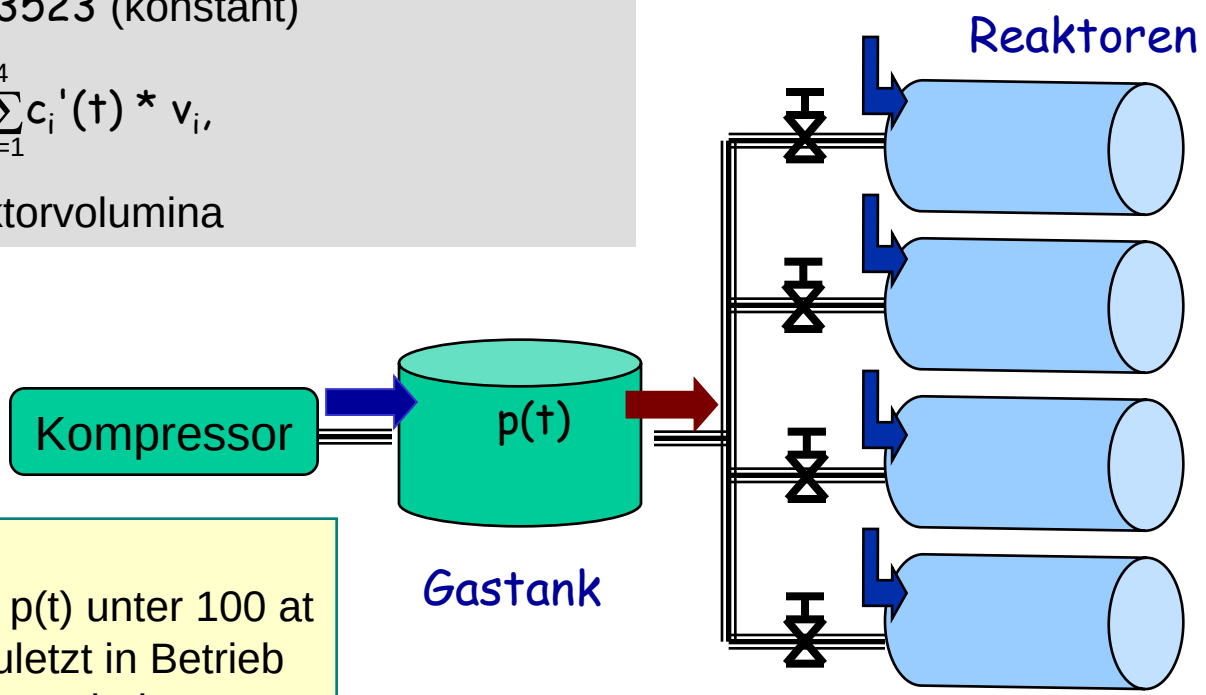
$$vomKompressor = 4.3523 \text{ (konstant)}$$

$$zuReaktoren(t) = \sum_{i=1}^{4} c_i'(t) * v_i,$$

und v_i als konstante Reaktorvolumina

p und $c_1, c_2, \ldots, c_4$

sind rückgekoppelte Zustandsgrößen



Zustandsereignis
Sollte der Gastankdruck p(t) unter 100 at sinken, soll immer der zuletzt in Betrieb genommene Reaktor abgeschaltet werden

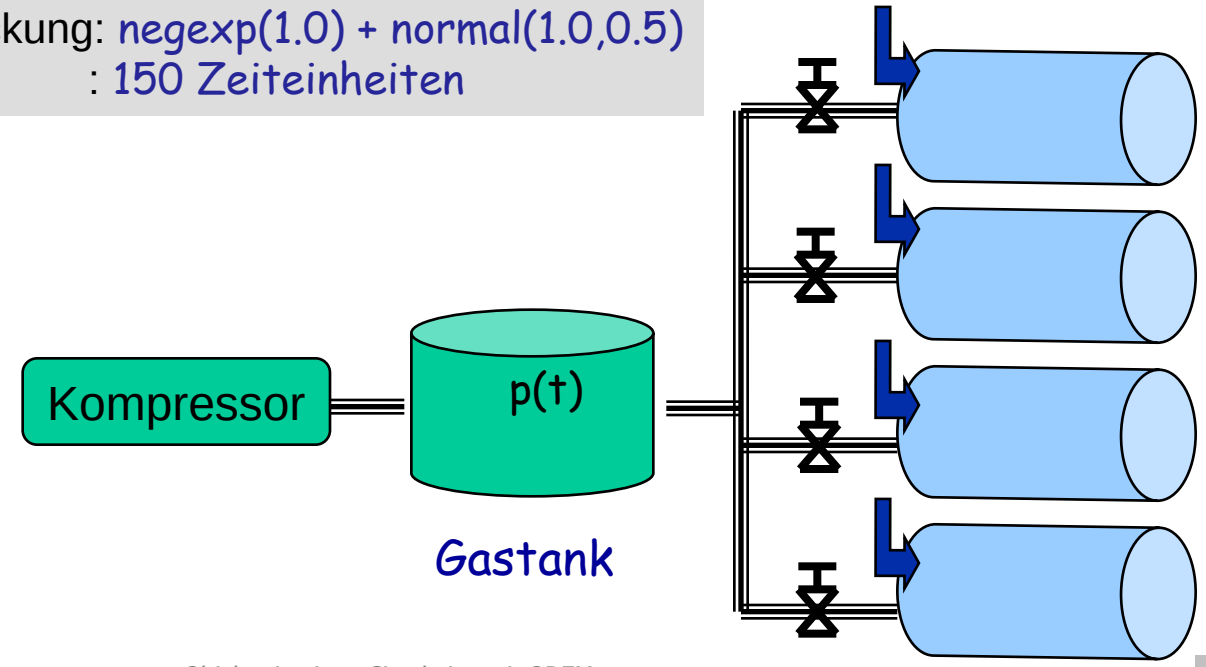
Parametrisierung des Modells

Systemkonstanten:

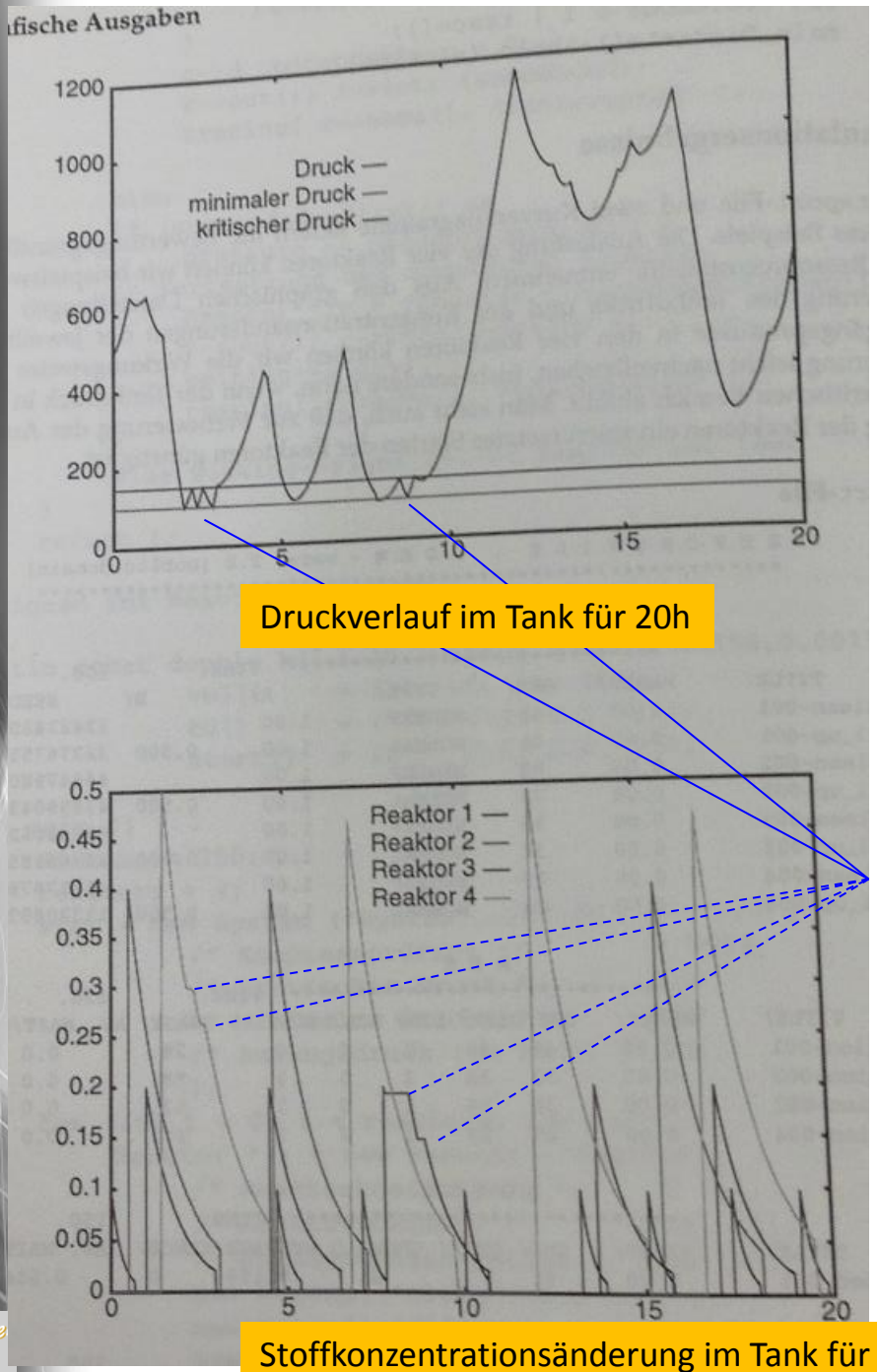
	k_i	v_i	c_i	erster Startzeitpunkt
Reaktor 1	0.03466	10	0.1	0.0
Reaktor 2	0.00866	15	0.4	0.5
Reaktor 3	0.01155	20	0.2	1.0
Reaktor 4	0.00770	25	0.5	1.5

Zeitdauer einer Neubeschickung: negexp(1.0) + normal(1.0,0.5)
Zeitdauer der Simulation : 150 Zeiteinheiten

1 ZE= 1h



Typisches Zeitverhalten



Druckverlauf im Tank für 20h

Stoffkonzentrationsänderung im Tank für 20h

Erweitertes
Ziel der simulativen Untersuchung

Bestimmung einer
optimalen Fahrweise der Reaktoren:

- Vermeidung von Abschaltungen während der Reaktion
- mit maximalem Output

Abschaltphasen

8. Ausblick:

Behandlung zeitkontinuierlicher Zustandsänderungen

- Beispiel: Feuerwehreinsatz
- Konzept für die zeitkontinuierliche Simulation
- Chaotische Systeme
- Weitere Beispiele

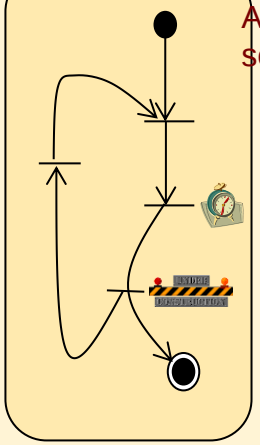
Bisher: Sequentieller Prozess

mit Zustandsgrößen, die sich zeitdiskret ändern

generelles Problem
Synchronisation nebenläufiger abhängiger Prozesse

ODEMx-Lösung:
NextEventSimulation und Process-Scheduling

Attribute



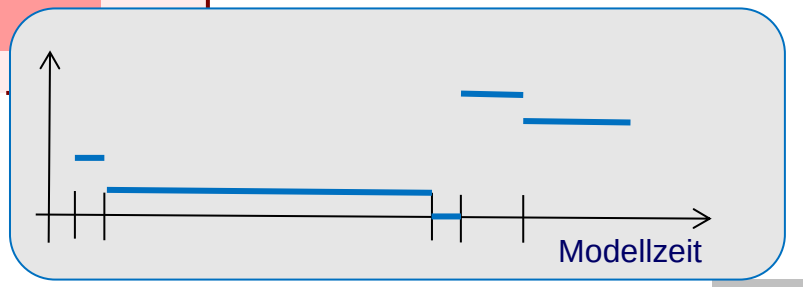
Aktionsfolge sequentiell

ändert Zustand des Systems, d.h.
eigene
fremde
Attribute
erzeugt/vernichtet/aktiviert Objekte
mit
- eventl. Zeitverbrauch,
- zustandsbedingten Blockierungen,
- unbedingten Blockierungen

Umgebung (Prozesse, Objekte passiver Klassen globale Größen)

Attribute

Attribute



zeitdiskrete Änderung
einer einzelnen Zustandsgröße

Jetzt zusätzlich: Sequentieller Prozess

mit Zustandsgrößen, die sich zeitkontinuierlich ändern

elementare Attribute: nur **double**
meist in Strukturen/Feldern verpackt

ODEMx-Lösung:
Kontinuierliche Prozesse als Ableitung diskreter Prozesse

Attribute

ändert Zustand
- *eigene*
- *evtl. auch fremde Attribute (zeitkontinuierlicher Prozesse)*

Integrationsverfahren
als Ein-Schritt-Verfahren

sequentielle Aktionen,
die einen Integrationsschritt
mit Schrittweite **h** ausführen

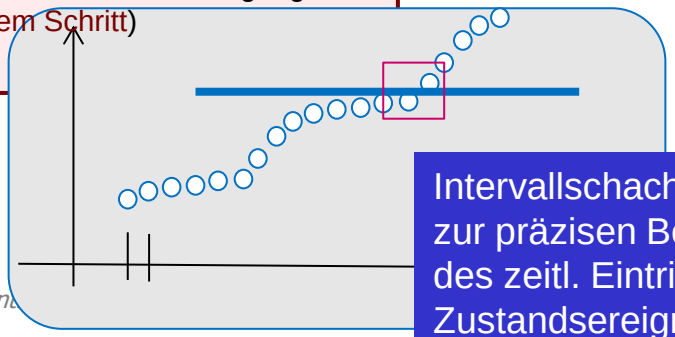
zeitliche Verzögerung um Schrittweite **h**

10^{-5}

condQ.Signal()

Attribute

Abbruch/Unterbrechung
- nach Erreichen einer bestimmten Zeit
- nach Eintritt einer Zustandsbedingung
  (Test nach jedem Schritt)



zeitdiskrete Approximation
der Änderung einer einzelnen
zeitkontinuierlichen Zustandsgröße

Intervallschachtelung
zur präzisen Bestimmung
des zeitl. Eintritts des
Zustandsereignisses

Grundsätzliche Einteilung von Systemen und Modellen (Erinnerung)

- Einteilung von (Teil-) Systemen
  - **zeitdiskrete** Prozesse
    - kennen wir bereits
  - **zeitkontinuierliche** Prozesse
    - nach höchster Ableitung
    - nach Anzahl der Zustandsgrößen
    - System n-ter Ordnung hat n Zustandsgrößen

kombinierte Systeme
Beispiele:
- Niedrigtemperaturofen,
- Feuerwehr,
- Tanker-Tank
- Reaktor, Tank

modelliert als System
von n Differentialgleichungen 1. Ordnung

(math. äquivalent zu einer Differentialgleichung n-ter Ordnung)

size‘ (t)= c * size(t)
damage‘ (t)= size(t)

size (0.0)= c *
damage (0.0)= size(t)

Hausbrand ~ kontinuierlicher Prozess mit zwei Zustandsgrößen

- Gewöhnliche DGL
- als Anfangswertaufgabe
- (Teil-)system 1.Ordnung der Dimension 2
- Lineare DGL

Integrationsverfahren

Betrachten hier sehr einfaches Verfahren (Euler-Heun-Verfahren)

size‘ (t)= c * size(t) – extinguishRate

damage‘ (t)= size(t)

- Vektoren x(t) und x‘(t) gegeben
  hier: x(t)= (size, damage) (t)
  x‘(t)= (size‘, damage‘) (t)

- Berechnung der Änderungen zum Zeitpunkt t
  hier x‘(t) berechnet sich entsprechend der beschreibenden DGLs
  Vor.: man kennt x zu diesem Zeitpunkt t
  (Anfangswert muss gegeben sein)

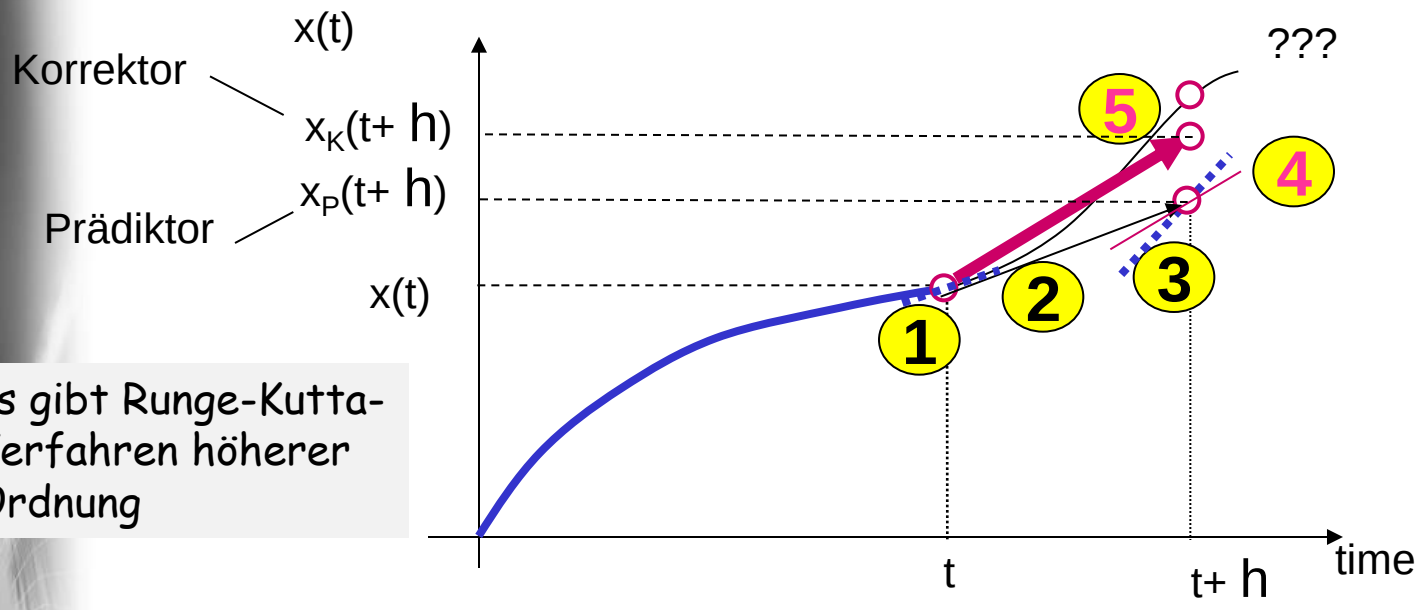
size (t_0) = initial_size
damage (t_0) = initial_size

- **Prädiktor-Schritt:**
  Berechnung der neuen Werte zum Zeitpunkt t+h (Schrittweite h)
  x(t+h)= x(t) + h* x‘(t) // x´(t)= (size‘, damage‘) (t) ➔ r1(t)

  Berechnung der Änderungen zum Zeitpunkt t+h
  hier (size‘, damage‘) (t+h) , nach Anwendung der DGLs
  // r2(t)= (size‘, damage‘) (t+h)

- **Korrektor-Schritt:**
  abermalige Berechnung der neuen Werte zum Zeitpunkt t+h
  x(t+h)= x(t) + h/2* (r1(t) + r2(t))

Runge-Kutta- Integrationsverfahren (2. Ordnung)

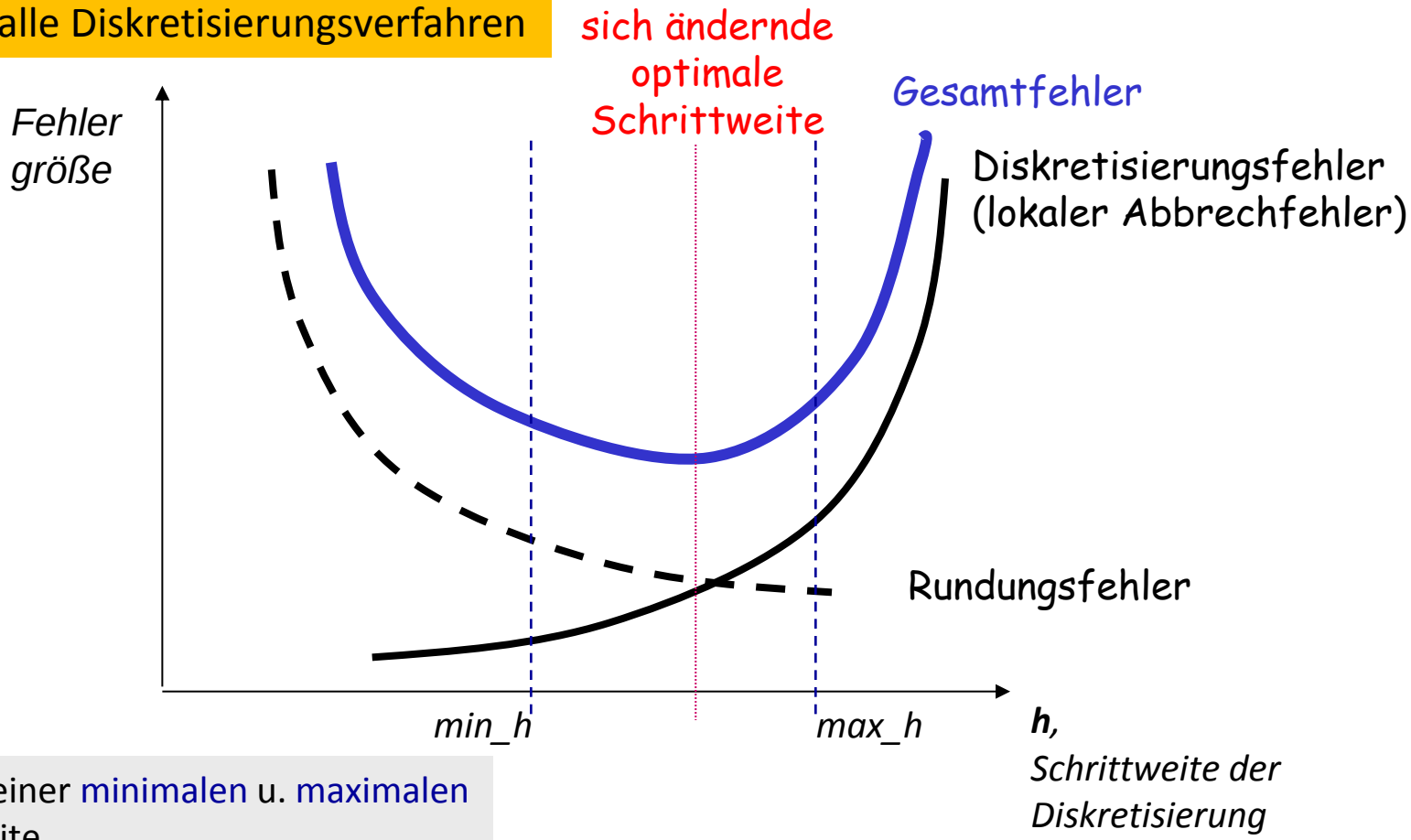


es gibt Runge-Kutta-Verfahren höherer Ordnung

1. berechne x'(t), d.h. f(x(t), t)
2. berechne x(t+h) nach Euler-Vorwärts mit x'(t)
3. berechne x'(t+h), d.h. f(x(t+h))
4. **Prädiktorschritt**: bilde den Mittelwert von x'(t) und x'(t+h)
5. **Korrektorschritt**: wiederhole Berechnung von x(t+h) nach Euler-Vorwärts, diesmal aber mit dem Mittelwert der beiden Ableitungen

Fehlerüberlagerung

... gilt für alle Diskretisierungsverfahren



Vorgabe einer minimalen u. maximalen Schrittweite
bei Verfahren, die ihre Schrittweite bei Diskretisierung dynamisch anpassen

Zeitkontinuierliche *Zustandsänderungen in ODEMx*

Einführung einer aktiven Klasse Continuous

- verwaltet „kontinuierliche“ **Variablen** anderer Prozesse (Referenzen auf Attribute)

  hier: die Variablen size, damage
  eines jeden Hausbrand-Objektes

  - verwaltet „beschreibende“ **DGLs**
    hier: die Berechnungsvorschriften für jede Brandentwicklung
    (als Zeiger auf eine Member-Funktion)

  - numerisches **Integrationsverfahren**,
    das zu einem Zeitpunkt t mit
    einer vorgegebenen Schrittweite h aus den aktuellen Werten der kontinuierlichen Variablen (unter Nutzung der DGLs)

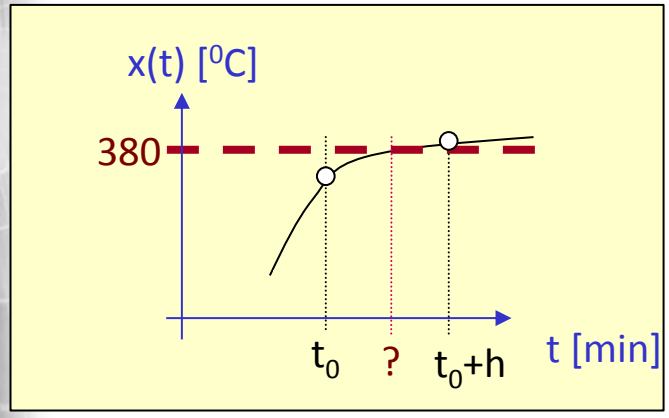
    ihre Werte zum Zeitpunkt t+h ermittelt und
    sich mit holdFor(h) verzögert.

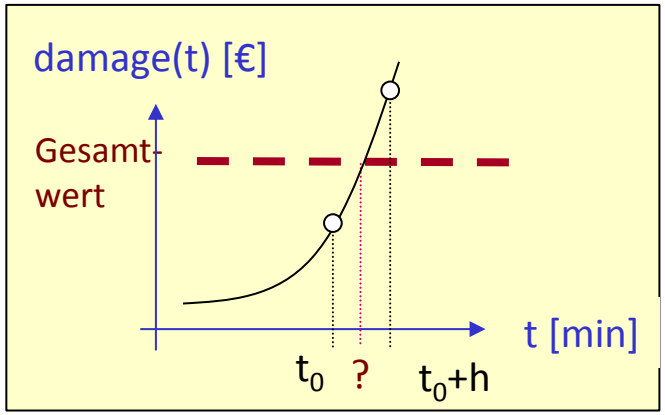
- … bei Bewältigung einer Reihe von Grundproblemen
  - numerische Genaugikeit/ Berechnungsgeschwindigkeit
    (~numerische Schrittweite und von Wahl des Integrationsverfahrens
    (dynamisch änderbar)
  - Synchronisation mit anderen zeitdiskreten und zeitkontinuierlichen Prozessen
  - Genauigkeit bei der Bestimmung von Zustandsereignissen
  - dynamische Änderung der Verhaltensbeschreibung

Zustandsereignisse zeitkontinuierlicher Abläufe

- Überwachung jedes vollzogenen (numerisch akzeptierten) Integrationsschrittes, bei Überprüfung einer zugeordneten **Zustandsbedingung**
  (per Zeiger einer zugeordneten Memberfunktion)

- Bei Eintritt der Bedingung wird versucht, den Zeitpunkt des Eintritts der Bedingung genauer zu fassen:
  - **Interpolation** (nur begrenzt anwendbar)
  - **Intervallschachtelung** bei Halbierung der Integrationsschrittweite und Abbruch der numerischen Integration, sobald man eine untere vorgegeb Schranke für die Schrittweite (**min_h**) erreicht hat





8. Ausblick:

Behandlung zeitkontinuierlicher Zustandsänderungen

- Beispiel: Feuerwehreinsatz
- Konzept für die zeitkontinuierliche Simulation
- Chaotische Systeme
- Weitere Beispiele

Chaotische Systeme

- bei den meisten realen Systemen bleiben benachbarte Zustandsbahnen im Laufe der Zeit nahe beieinander.

➔ Systeme sind deshalb vorhersagbar

- aus den Anfangswerten lässt sich die zukünftige Entwicklung ermitteln
- Entwicklung ist gegenüber kleinen Messfehlern der Anfangswerte unempfindlich

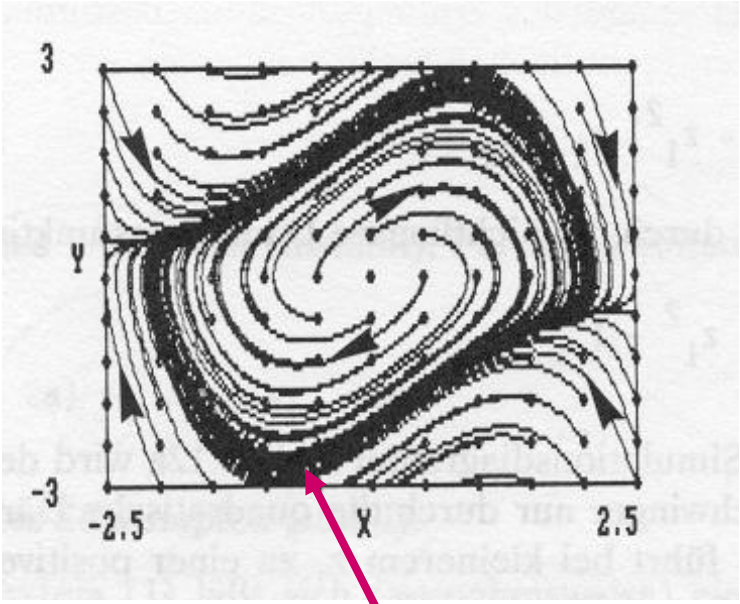
- Verhaltensbereich chaotischer Systeme zerfällt zwar in Regionen, in denen sich der Systemzustand nach einer gewissen Zeit befinden muss, ohne dass jedoch sein exakter Ort mit Sicherheit vorhergesagt werden kann
  - benachbarte Zustandsbahnen streben exponentiell auseinander, verlassen aber ihre Region nicht
  - Endzustand liegt irgendwo auf dem Regionen-Rand

➔ chaotische Systeme sind deshalb nicht vorhersagbar

Attraktoren als Grenzlinie

Verlauf mit a= -1.0

- teilen den Zustandsraum in Regionen mit unterschiedlichem Verhalten
- **weiteres Beispiel**:
  van der Pol Oszillator



Grenzlinie

$$z'_1 = z_2$$
$$z'_2 = az_1 + (1 - z_1^2)z_2$$

$$z'_1 = z_2$$
$$z'_2 = (1 - z_1^2)z_2 - z_1$$

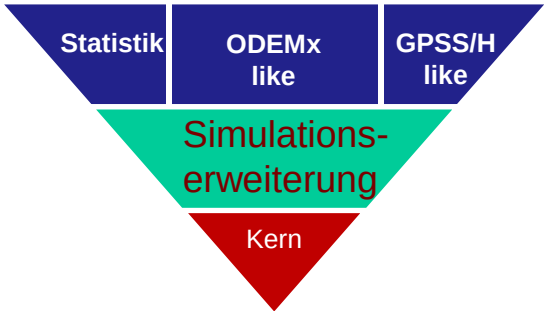
noch kein chaotisches System

8. Ausblick:

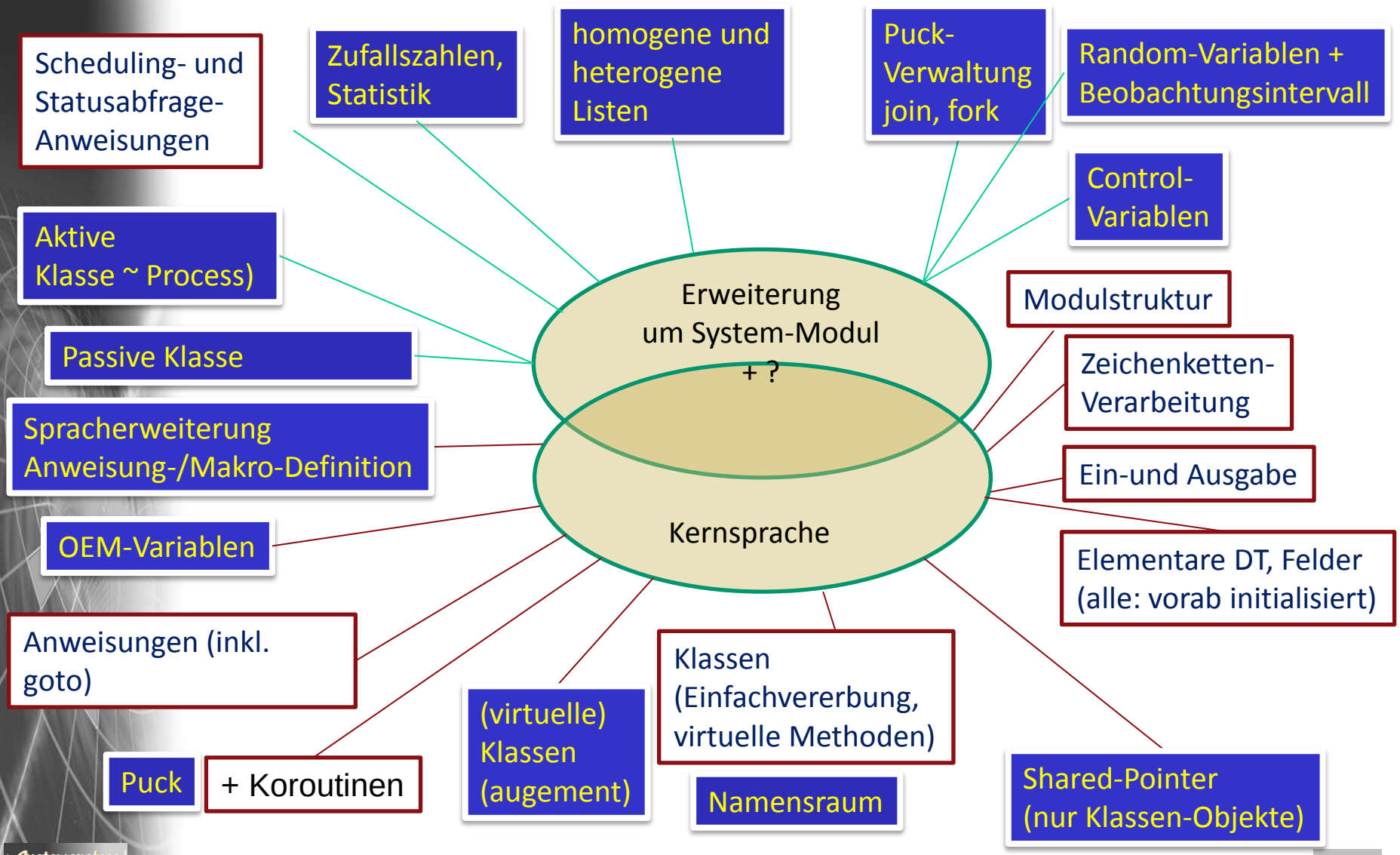
Alternative Simulationssprachen (SLX, SDL, Simulink)

Hintergrund: in Vorbereitung befindliche Projekte

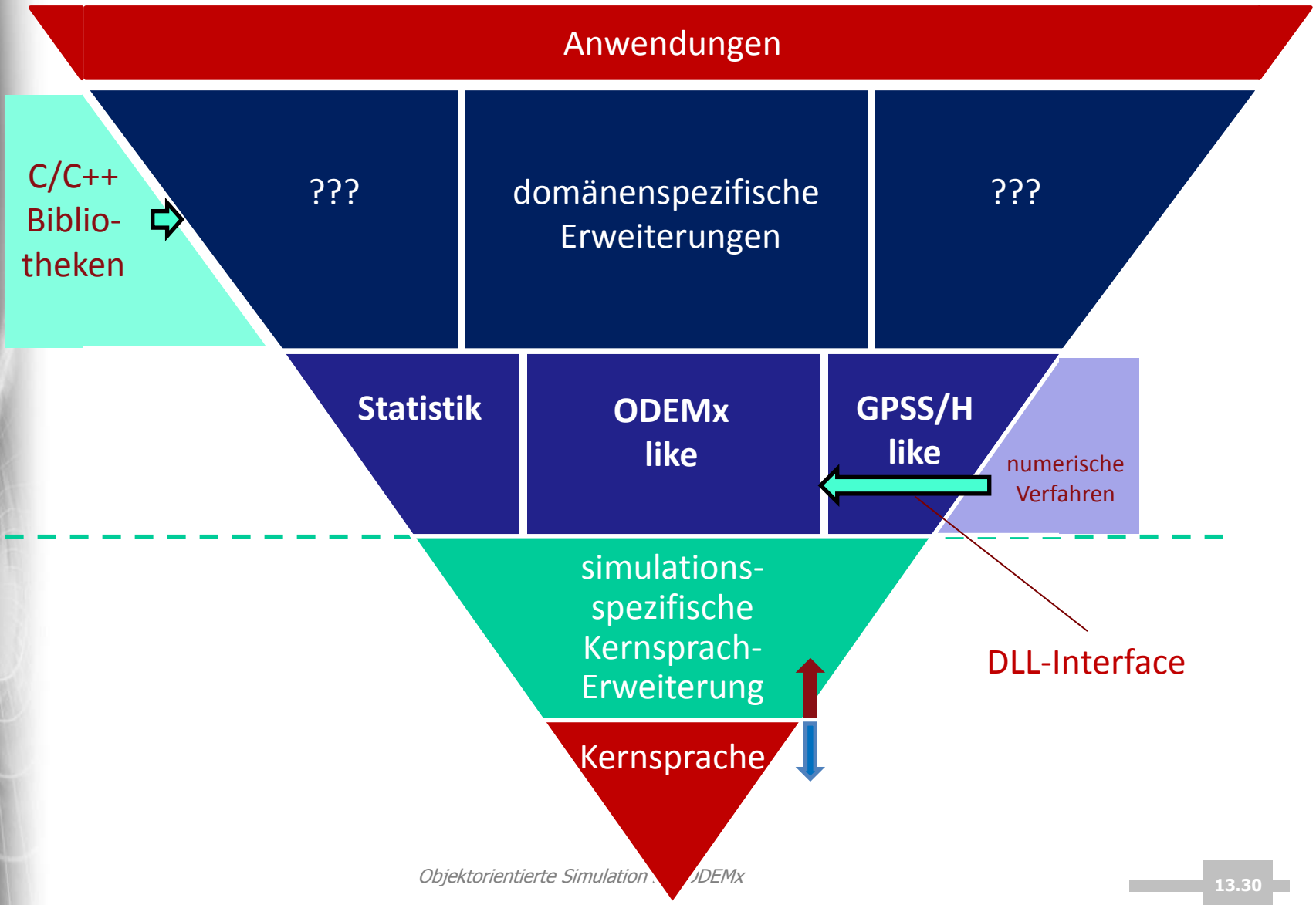
- SLX- Deutsche Bahn AG
- SLX- Workflows, SLX-Logistik
- ODEMx Verbesserung



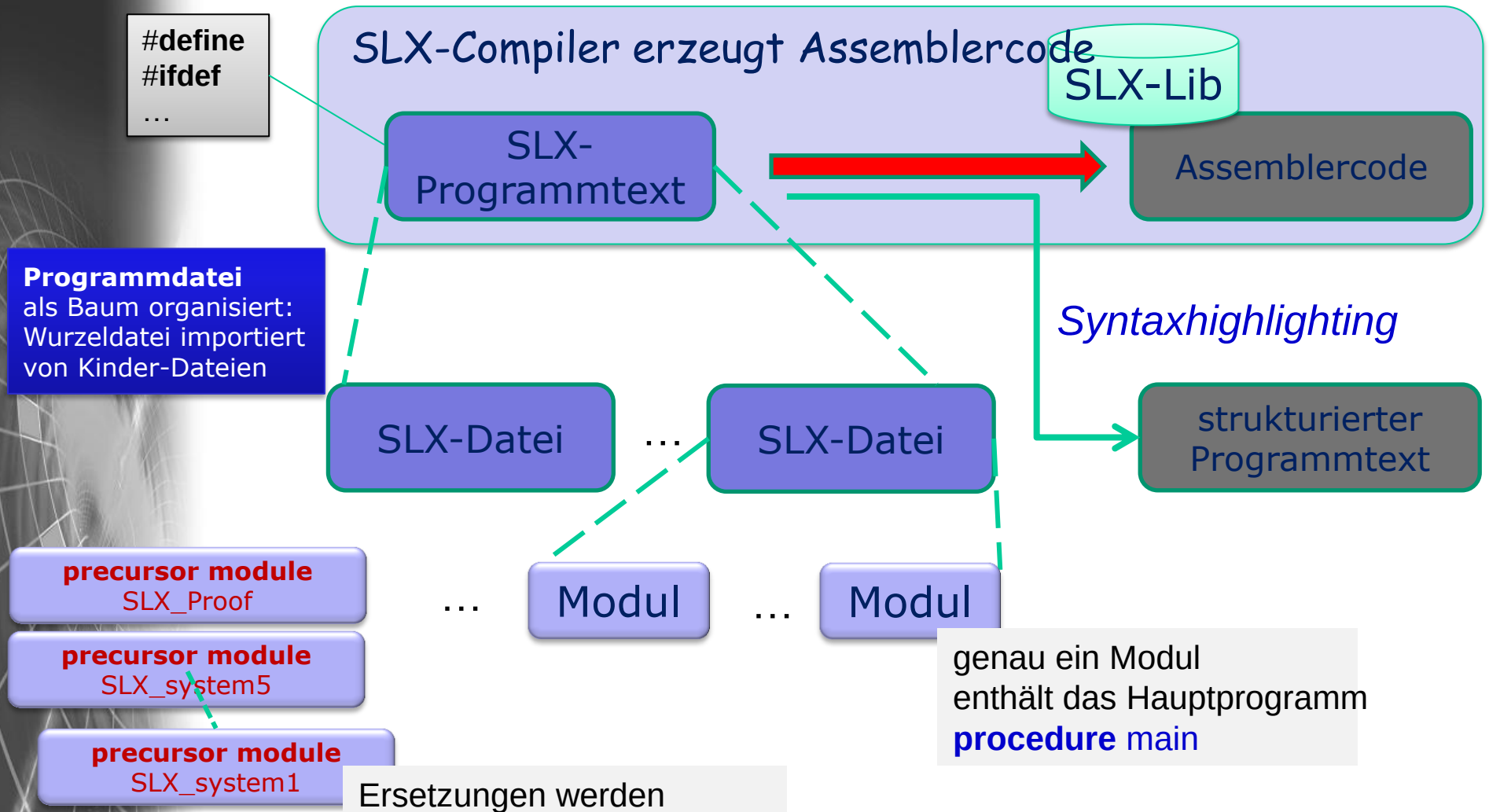
SLX-Konzept-Charakterisierung



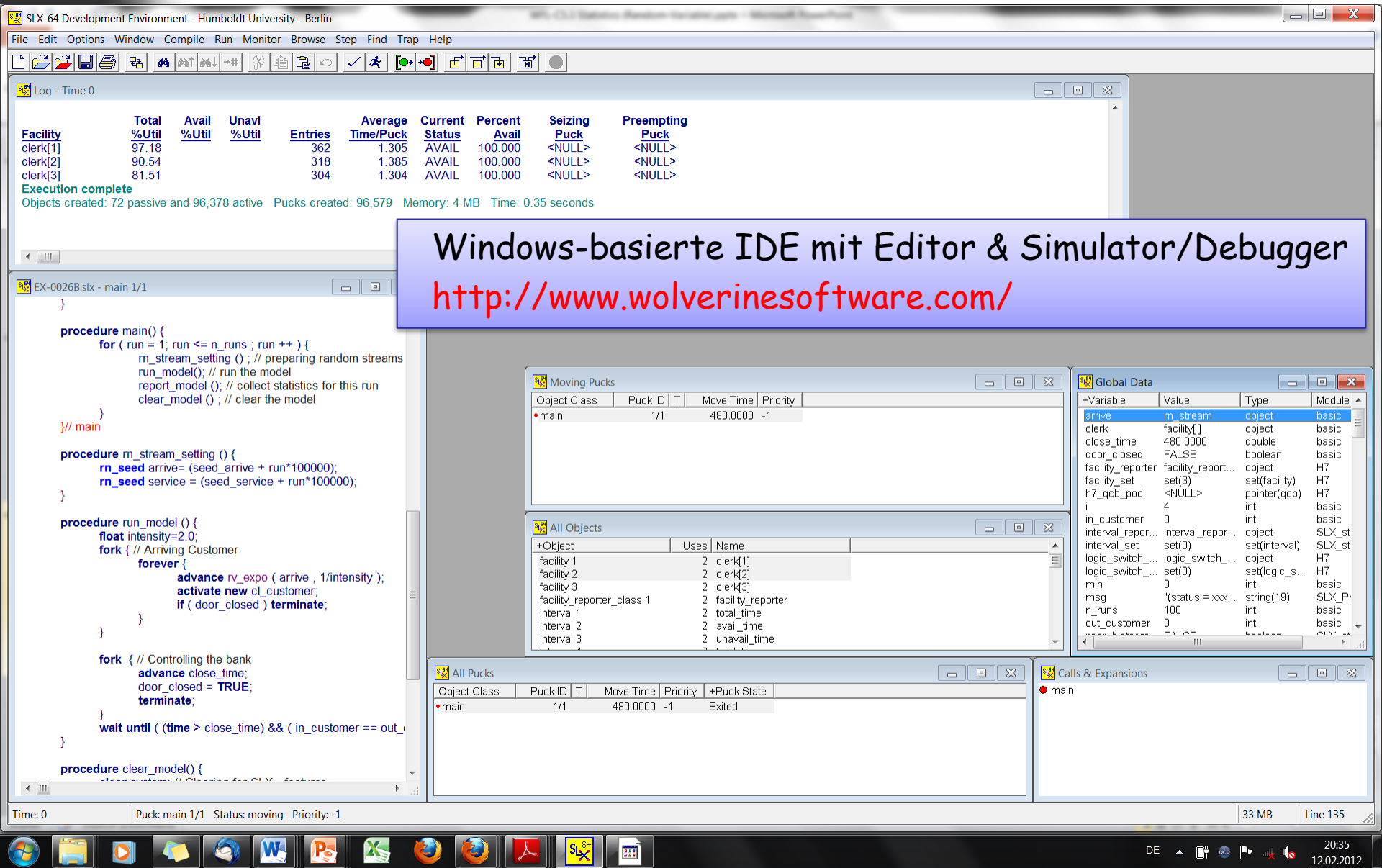
Die SLX-Modul-Pyramide



SLX-Programmstruktur



SLX-Entwicklungsumgebung



Externe Schnittstellen

Einbindung von SLX in andere Systeme

- universeller Datenaustausch über **formatierte Textdateien**

```
filedef MyXML input options=XML name="Myfile.xml";     // XML specified in a file definition
```

- universelles **DLL-Interface** (C/C++ -Programme)

- Kopplung mit
  - **Optimierungssystem** ISSOP (Dresdner Firma)
  - **ODBC-Datenbanken** (Schnittstellen-Modul, TU-Magdeburg)
  - **Animationswerkzeug** PROOF (Wolverine)
  - **HLA** (CORBA-Plattform für verteilte Simulation)
    OMG-Standard (amerikanische Verteidigungsministerium)

Zentrale Konzepte und ihre Interpretation in ODEMx

- System, Systemumgebung, Systemelement, Kopplungsrelation, Rückkopplung, Zustand, Zustandsgröße, Modellgröße, Zufallsgröße, Bewertungsgröße, Ein- und Ausgabegröße, Zustandsüberführungsfunktion, Ausgabefunktion, zeitdiskrete Systeme, zeitkontinuierliche Systeme, kombinierte Systeme
- OO-Konzepte: Klasse, Objekt, Struktur, Verhalten, Beziehung: Objekt-Systemelement, Beziehung: Objekt-Prozess, Beziehung: Prozess-Aktivität-Ereignis
- Modellierung unf OO-Abstraktion: Spezialisierung/Verallgemeinerung, Klassifikation/Exemplifikation, Zustandsautomat, Zustand, Zustandsübergang, asynchrone (Automaten-) Kommunikation, Trigger (in UML-Semantik)
- Simulation als zielgerichtete experimentelle Untersuchungsmethode bei Nutzung ausführbarer Modelle, Paradigmen: (1) “modelliere nicht so genau wie möglich, sondern so genau wie nötig”, (2) “verlieb Dich nicht in Dein Modell”
- Original, Modell, Simulationsmodell, Simulator, Realzeit, Modellzeit, Simulationszeit
- Ereignis (Zeitereignis, Zustandsereignis), Next-Event-Simulation, Zusammenhang (Zufall-Pseudozufall), Verteilungsfunktion (empirische, theoretisch), Zusammenhang der Verteilungsfunktionen

Obj

Reflektion bei Anwendung von ODEMx

Begriffe und Anwendungsbeispiele

- Unterstützt Modellklassen von ODEMx
- Scheduling-Mechanismen zur Umsetzung einer Next-Event-Simulation, bedingte und unbedingte Prozess-Blockierungen
- Spezialisierungen: Sched- Event, Sched- Process, Sched-Process-Continuous
- Synchronsationskonzepte: WaitQ, CondQ, Bin, Res, Memory (PortHead, PortTail, Cond, Timer)
- Unterbrechung von Warte- und Aktionsphasen eines Prozesses

Beispiele

- Fähre (in verschiedenen Variationen: Bin, Port, WaitQ, Res)
- Tanker-Hafen-Tank-Raffenerie
- Tanker-Schlepper-Anlegeplatz-Gezeiten
- Bagger-Fahrzeuge-Beladung
- Recycling-Hof (letzte Vorlesung)
- Feuerwehr/Eisenbarren/Reaktoren

Viel Erfolg in der OMSI-Prüfung !