Kurs OMSI
im WiSe 2012/13

Objektorientierte Simulation mit ODEMx

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

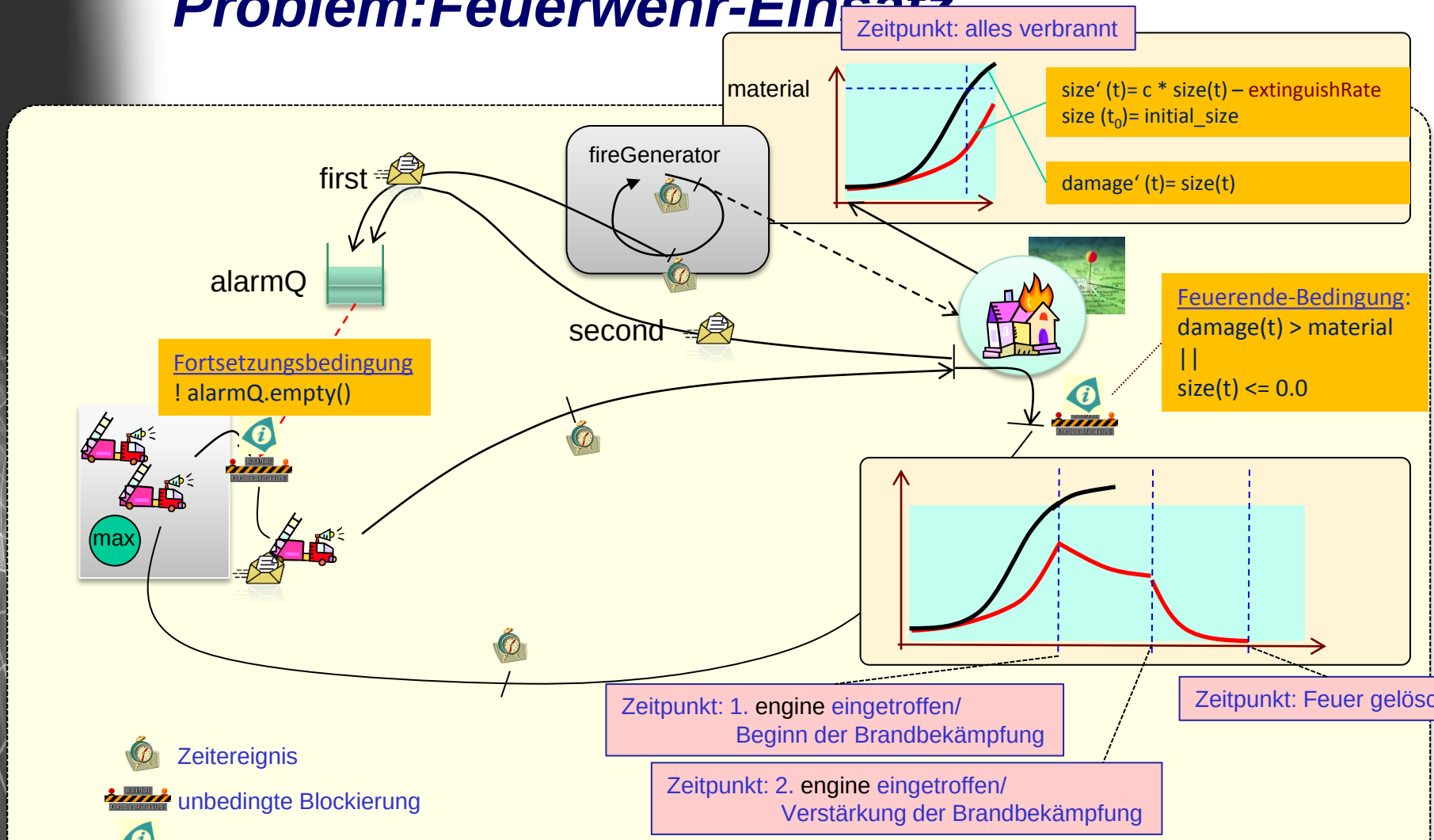
fischer|ahrens|eveslage@informatik.hu-berlin.de

8. Ausblick:

Behandlung zeitkontinuierlicher Zustandsänderungen

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

Problem:Feuerwehr-Einsatz



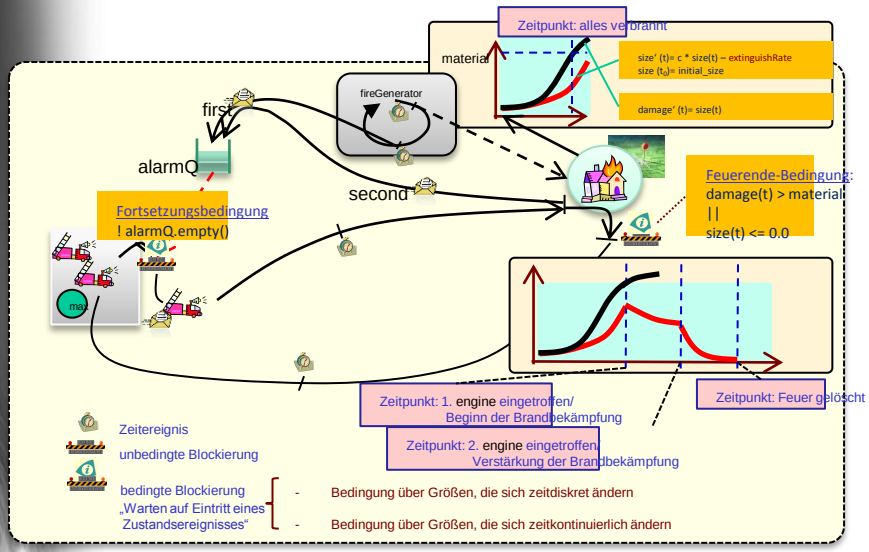
Zeitereignis

unbedingte Blockierung

bedingte Blockierung
„Warten auf Eintritt eines Zustandsereignisses“

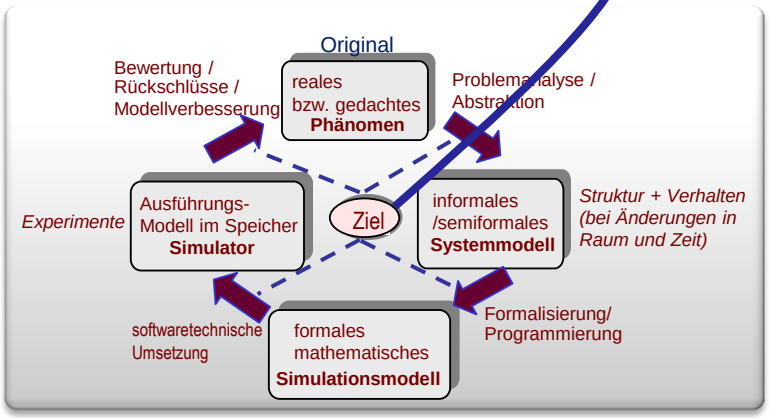
- Bedingung über Größen, die sich zeitdiskret ändern
- Bedingung über Größen, die sich zeitkontinuierlich ändern

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

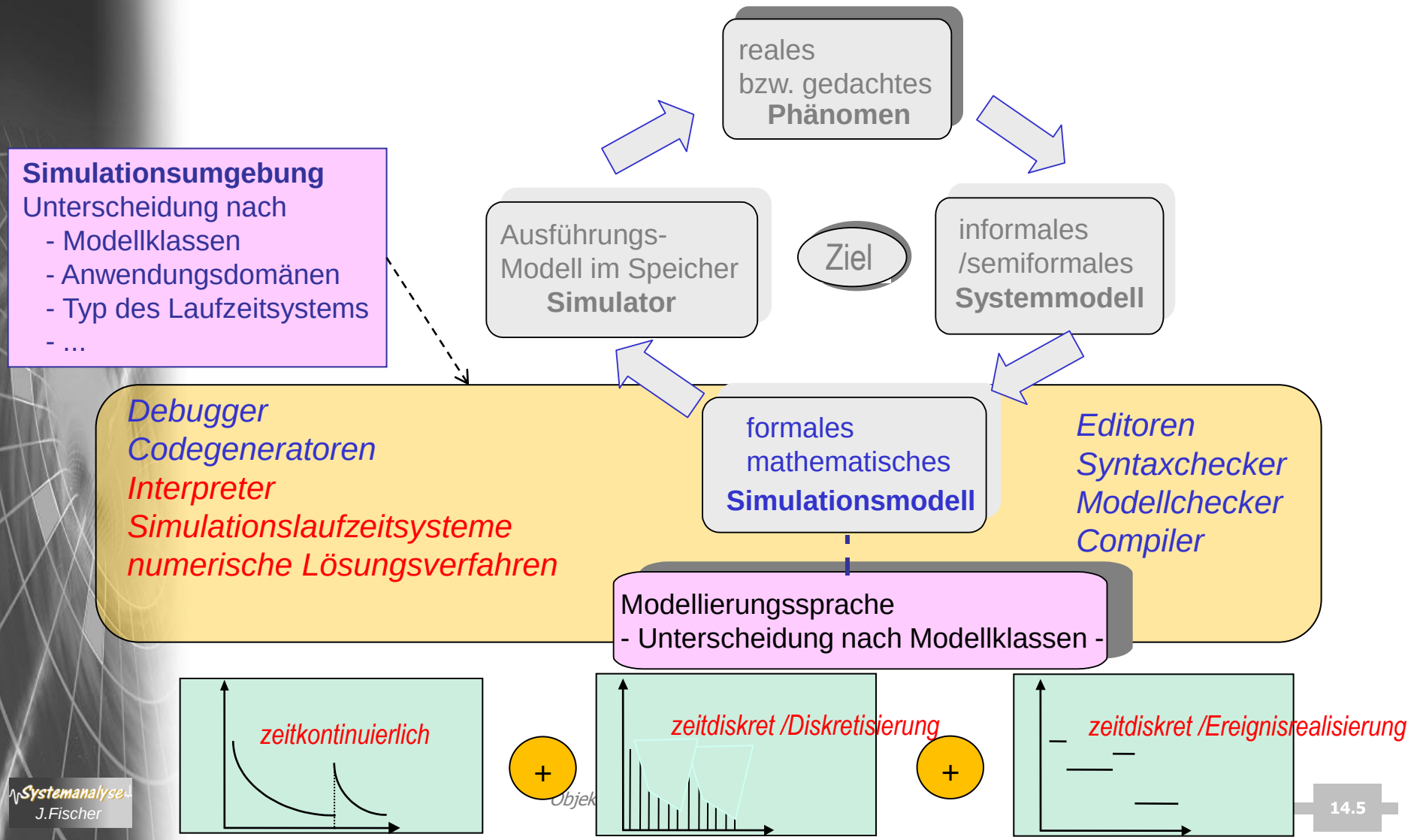
Struktur
Modellierung

Verhaltensmodellierung
Modellierung

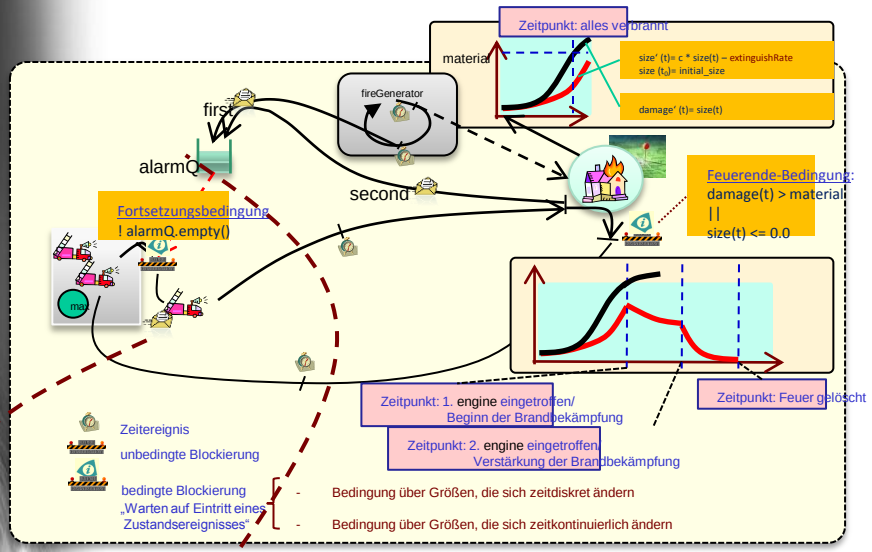
Math. Modellklasse
Verhaltensbeschreibung

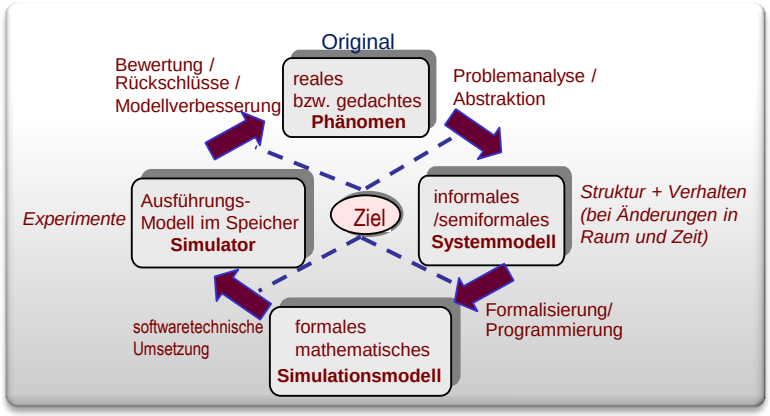
Modellierungssprachen und Simulationsumgebung

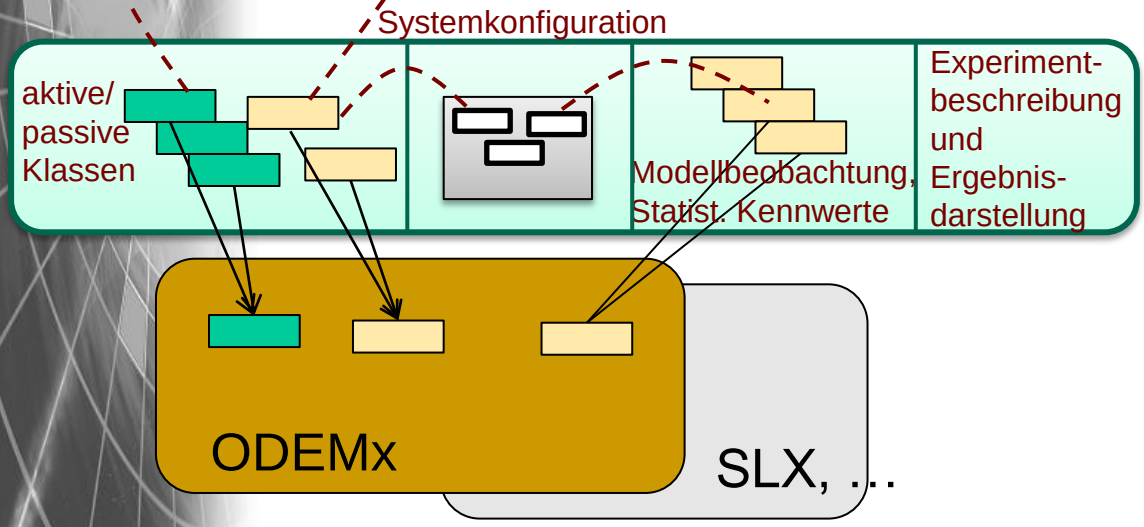
Zustandsänderungen kontinuierlich oder/und diskret in Raum und Zeit



Benötigte Konzepte







benötigte Konzepte

- Prozesse (diskrete)
  Zustandsereignisse
- Prozesse (kontinuierlich)
- Port-Synchronisation (Alarm-Engine)
- CondQ-Synchronisation (Rückfahrt von Engine)
- Zufallsgrößen
  - Hausbrand: (Zeitpunkt der Enstehung,
    Position, Anfangsgröße,
    Materialwert, Materialkoeffizient
    Zeitpunkt der Entdeckung)
  - Fahrtzeiten der Feuerwehr
- Tally/Count

8. Ausblick:

Behandlung zeitkontinuierlicher Zustandsänderungen

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

8. Ausblick:

Behandlung zeitkontinuierlicher Zustandsänderungen

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

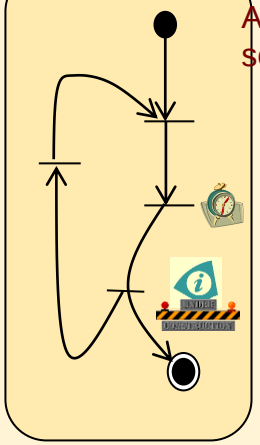
Sequentieller Prozess

mit Zustandsgrößen, die sich zeitdiskret ändern

generelles Problem
Synchronisation nebenläufiger abhängiger Prozesse

ODEMx-Lösung:
NextEventSimulation und Process-Scheduling

Attribute



Aktionsfolge sequentiell

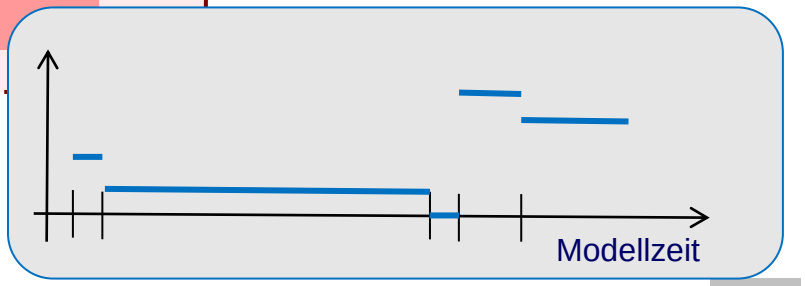
ändert Zustand des Systems, d.h.
eigene
fremde
Attribute
erzeugt/vernichtet/aktiviert Objekte
mit
eventl. Zeitverbrauch
Zustandsbedingten Blockierungen
unbedingten Blockierungen

Umgebung (Prozesse, Objekte passiver Klassen globale Größen)

Attribute

Attribute

zeitdiskrete Änderung
einer einzelnen Zustandsgröße



Sequentieller Prozess

mit Zustandsgrößen, die sich zeitkontinuierlich ändern

elementar: nur double
oder in Strukturen verpackt

Attribute

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

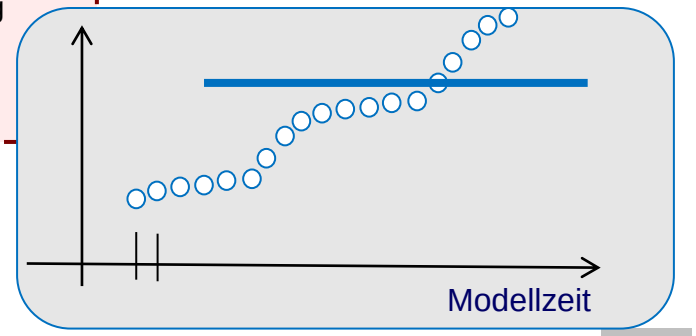
ODEMx-Lösung: Kontinuierliche Prozesse als Ableitung diskreter Prozesse

zeitliche Verzögerung um Schrittweite **h**

Integrationsverfahren
als Ein-Schritt-Verfahren

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

Abbruch
- nach Erreichen einer bestimmten Zeit
- nach Eintritt einer Zustandsbedingung



zeitdiskrete Approximation
der Änderung einer einzelnen
zeitkontinuierlichen Zustandsgröße

Grundsätzliche Einteilung von Systemen und Modellen (Erinnerung)

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

      modelliert als System
      von n Differentialgleichungen 1. Ordnung

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

kombinierte Systeme (Beispiel: Niedrigtemperaturofen)

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

- **Prädiktor-Schritt:**
  Berechnung der neuen Werte zum Zeitpunkt t+h

  x(t+h)= x(t) + h* x‘(t) // r1(t)= (size‘, damage‘) (t)

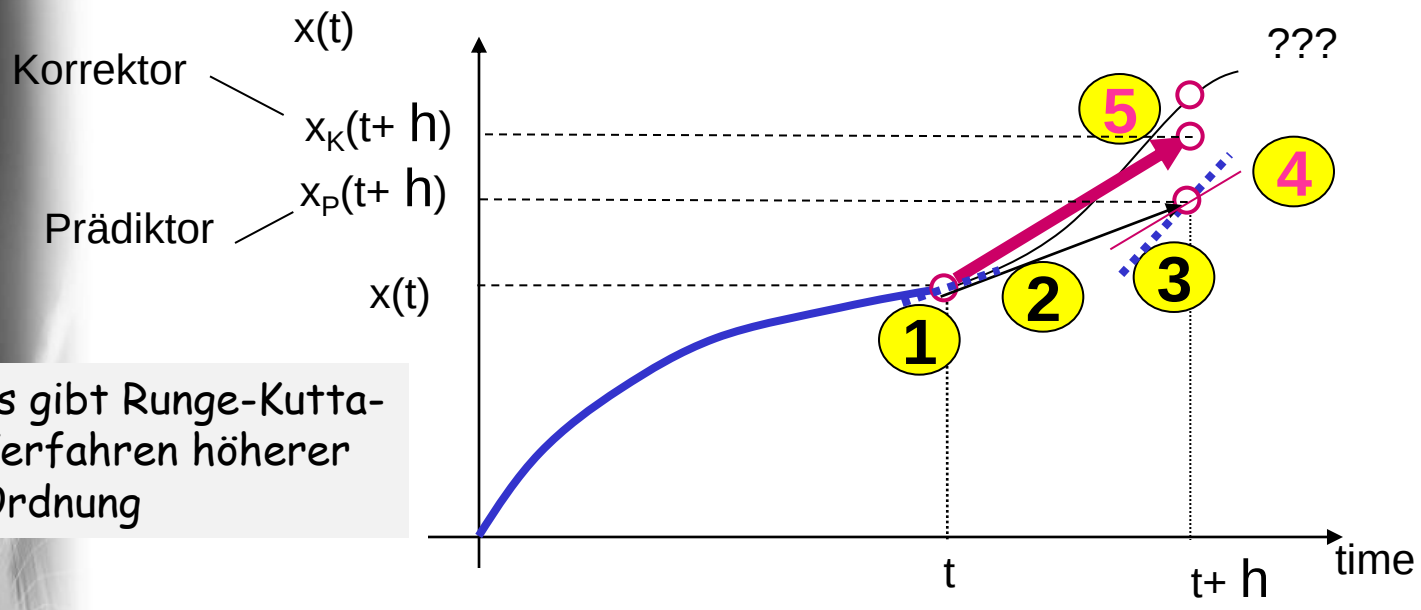
  Berechnung der Änderungen zum Zeitpunkt t+h
  hier (size‘, damage‘) (t+h) , nach Anwendung der DGLs

  // 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)

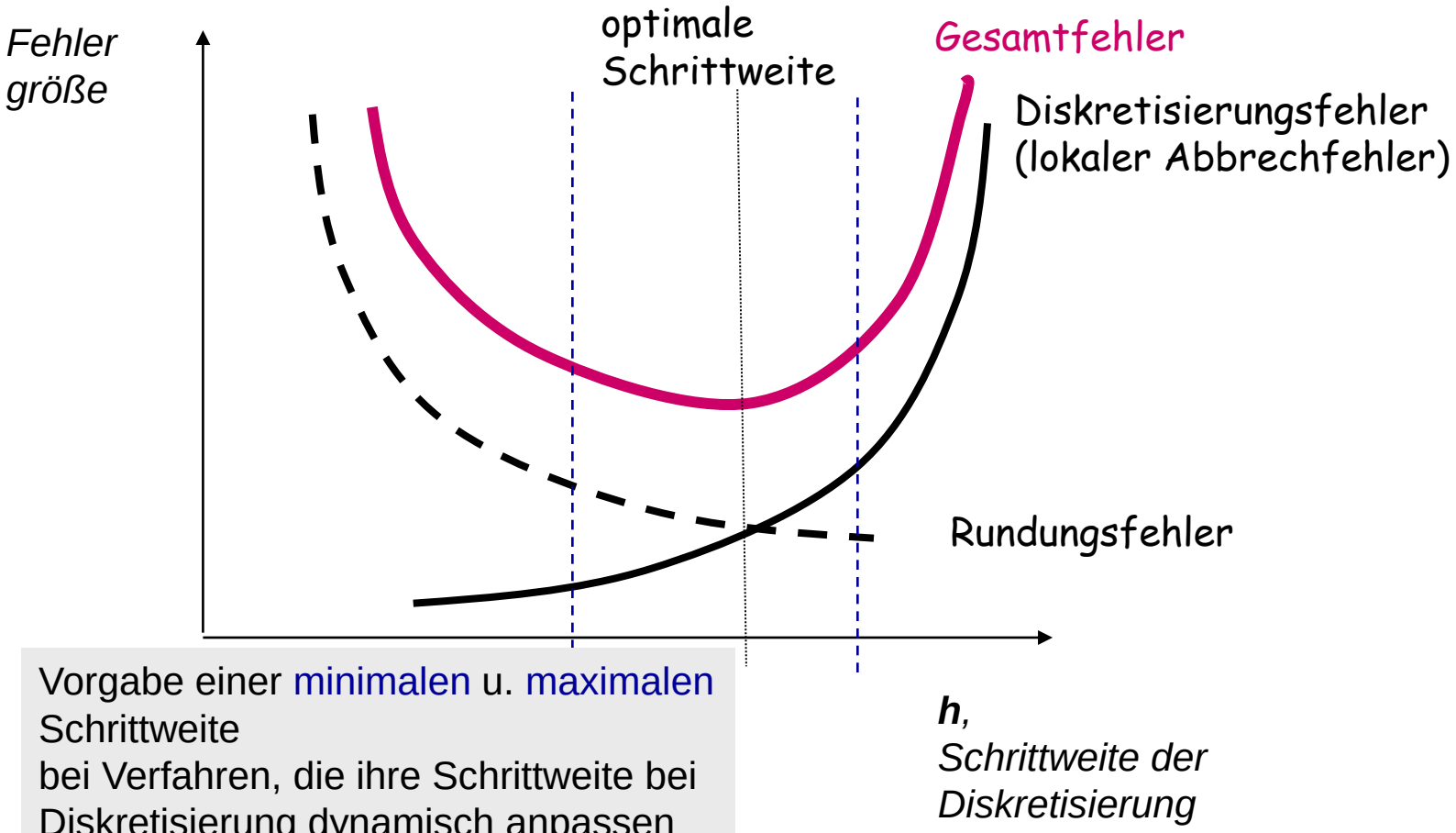


es gibt Runge-Kutta-Verfahren höherer Ordnung

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)

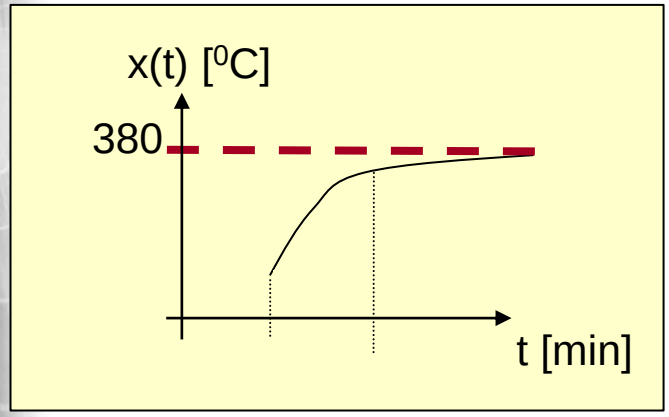
  ihre Werte zum Zeitpunkt t+h ermittelt und
  sich mit holdFor(h) verzögert.

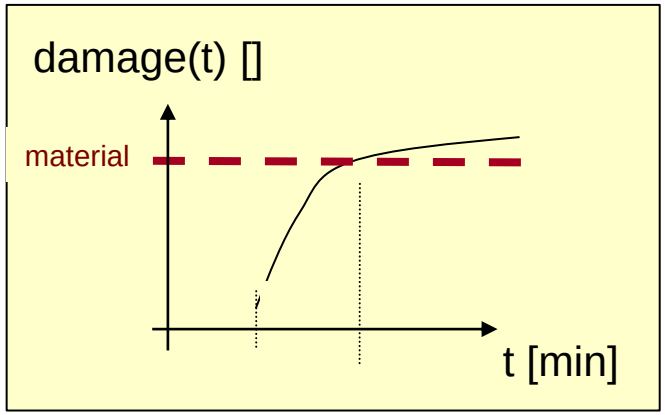
- … bei Bewältigung einer Reihe von Grundproblemen
  - numerische Genaugikeit/ Berechnungsgeschwindigkeit
    (~numerische Schrittweite und von Wahl des Integrationsverfahrens
    (dynamisch änderbar)
  - Synchronisation mit anderen zeitdiskreten und zeitkontinuierlichen Prozessen
  - Genauigkeit bei 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
  - Intervallschachtelung bei Halbierung der Integrationsschrittweite und Abbruch,
    sobald man eine untere vorgegebene Schranke für die Schrittweite erreicht hat



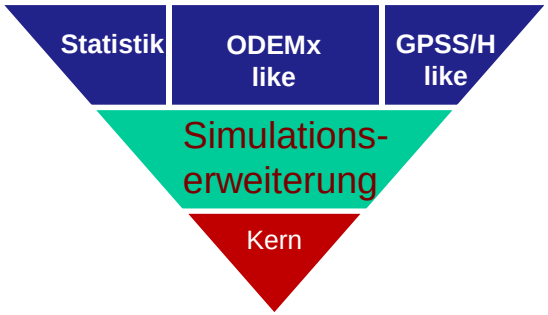


8. Ausblick:

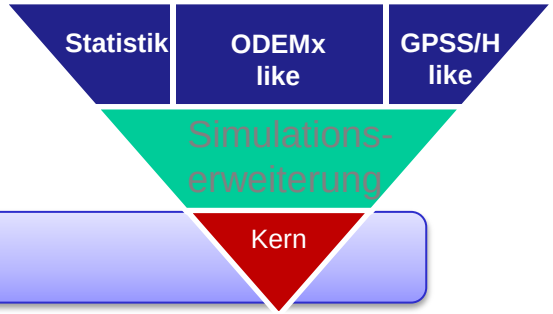
Alternative Simulationssprachen (SLX, SDL, Simulink)

Hintergrund: in Vorbereitung befindliche Projekte

- SLX- Deutsche Bahn AG
- SLX- VW
- Simulink-SLX (Windpark)

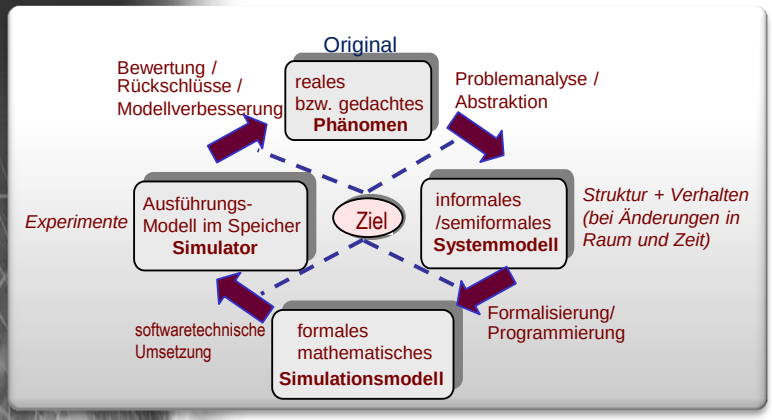


Wir suchen Interessenten für …
(mit Programmiererfahrung JAVA, C, …)



1. Motivation
2. Charakterisierung und Einordnung der Sprachkonzepte
3. Einige ausgewählte SLX-Sprachkonzepte
   - Datentypen, Felder, Klassen (nur kurz)
   - Aktive Klassen und Pucks
   - Interne Parallelität

Ausbildungsschwerpunkte: OMSI, IWF



Problemlösung Modellierung/ Simulation

Objektorientierte Modellierung

UML, SysML

Klassendiagramm, Zustandsdiagramm, Sequenzdiagramm

Workflows (industrielle Workflows)

Traditionelle Prinzip-Beispiele: Fähre, Hafen, Tiefofen-Barren, …

C++

ODEMx

Allgemeine Modellierungskonzepte

GPSS

SLX

Zufalls-größen

Ereignis Prozes

Prozess-synchronisation

Event/Proze Scheduling

Zustandsereignis (D, K)

Master-Slave (D)

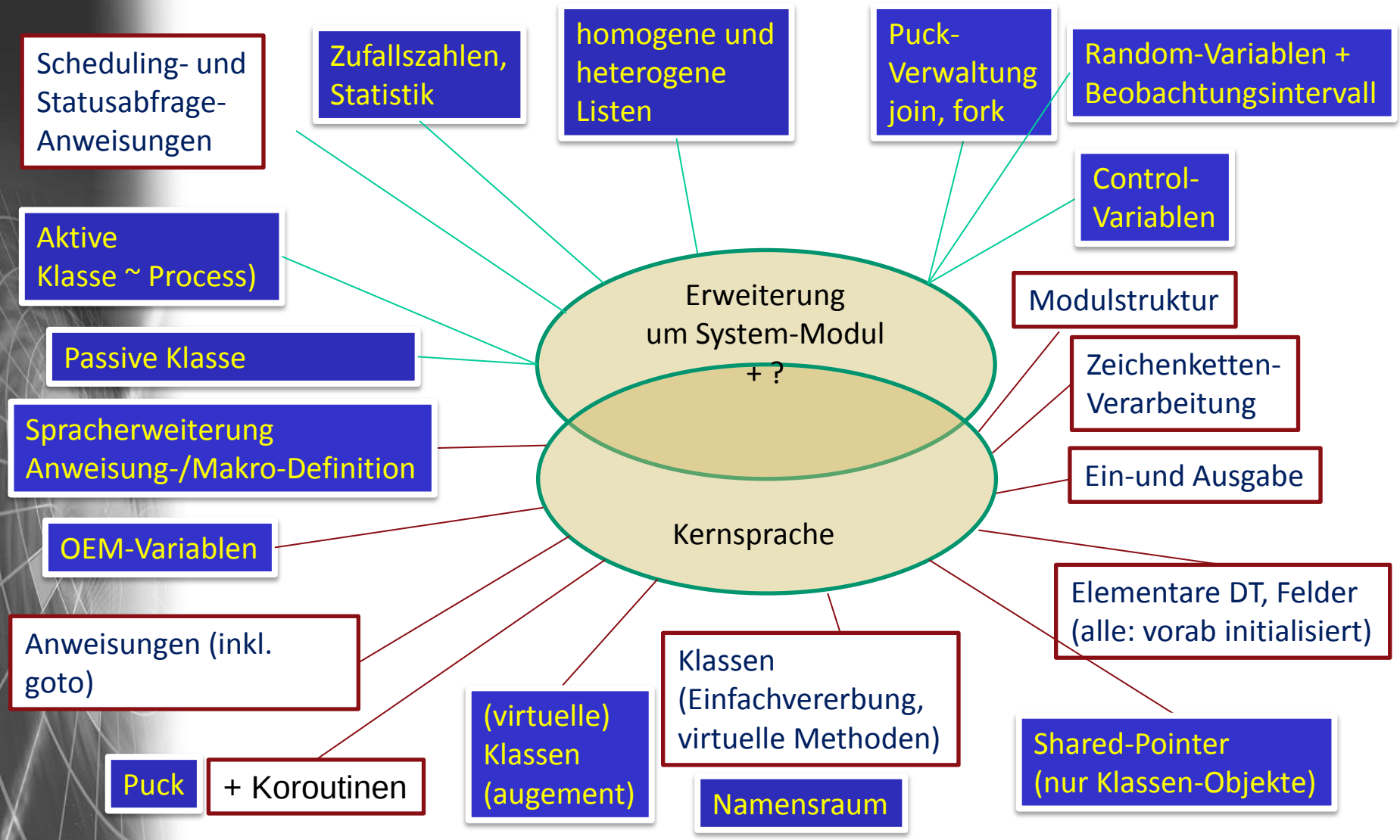
Interrupt von Bedienungs- und Warteaktivitäten

Master-Slave (K) Monitor, Variable, Continuous

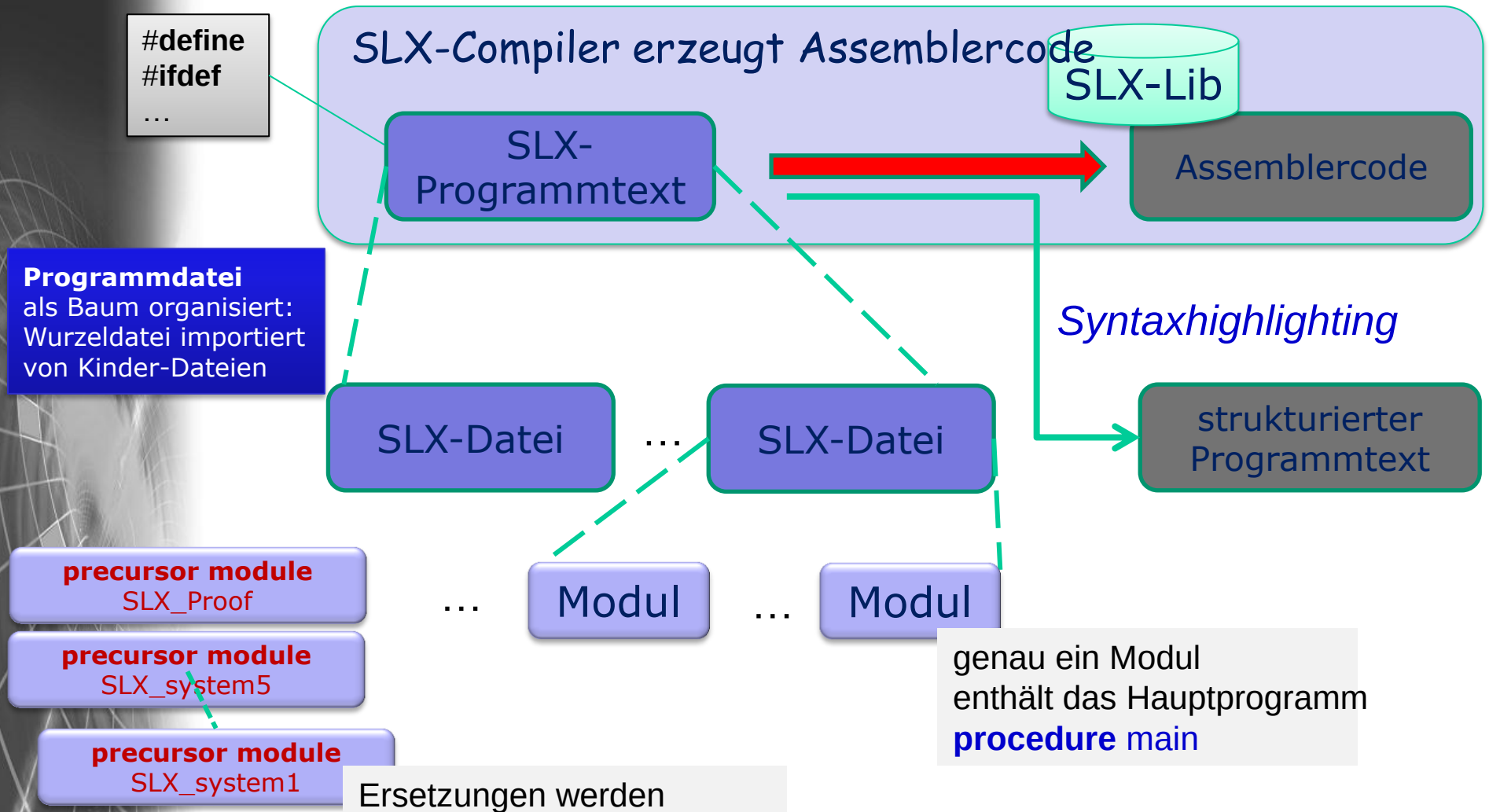
Statistik von Zufallsgrößen

Experimentauswertung

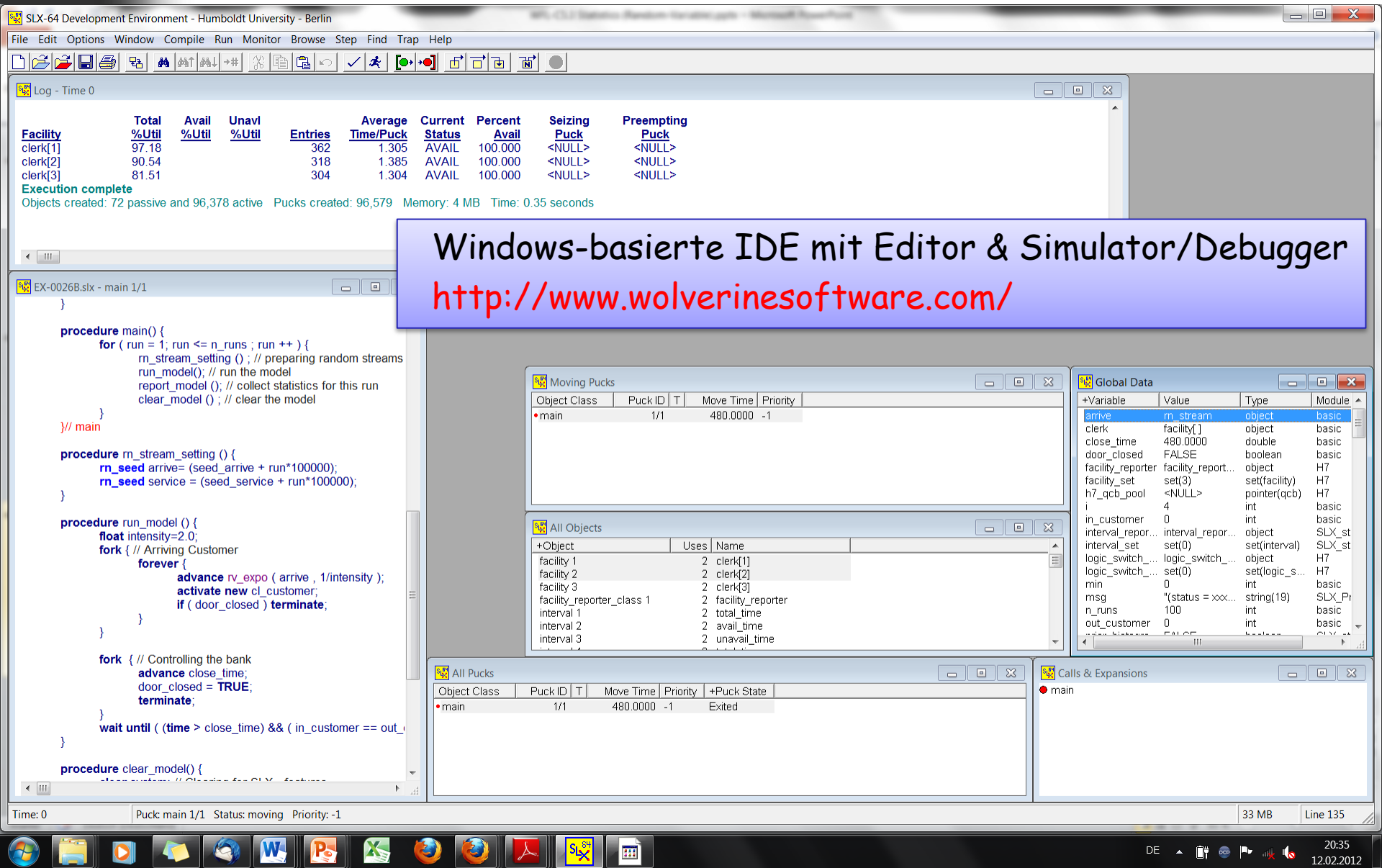
SLX-Konzept-Charakterisierung



SLX-Programmstruktur



SLX-Entwicklungsumgebung



Externe Schnittstellen

Einbindung von SLX in andere Systeme

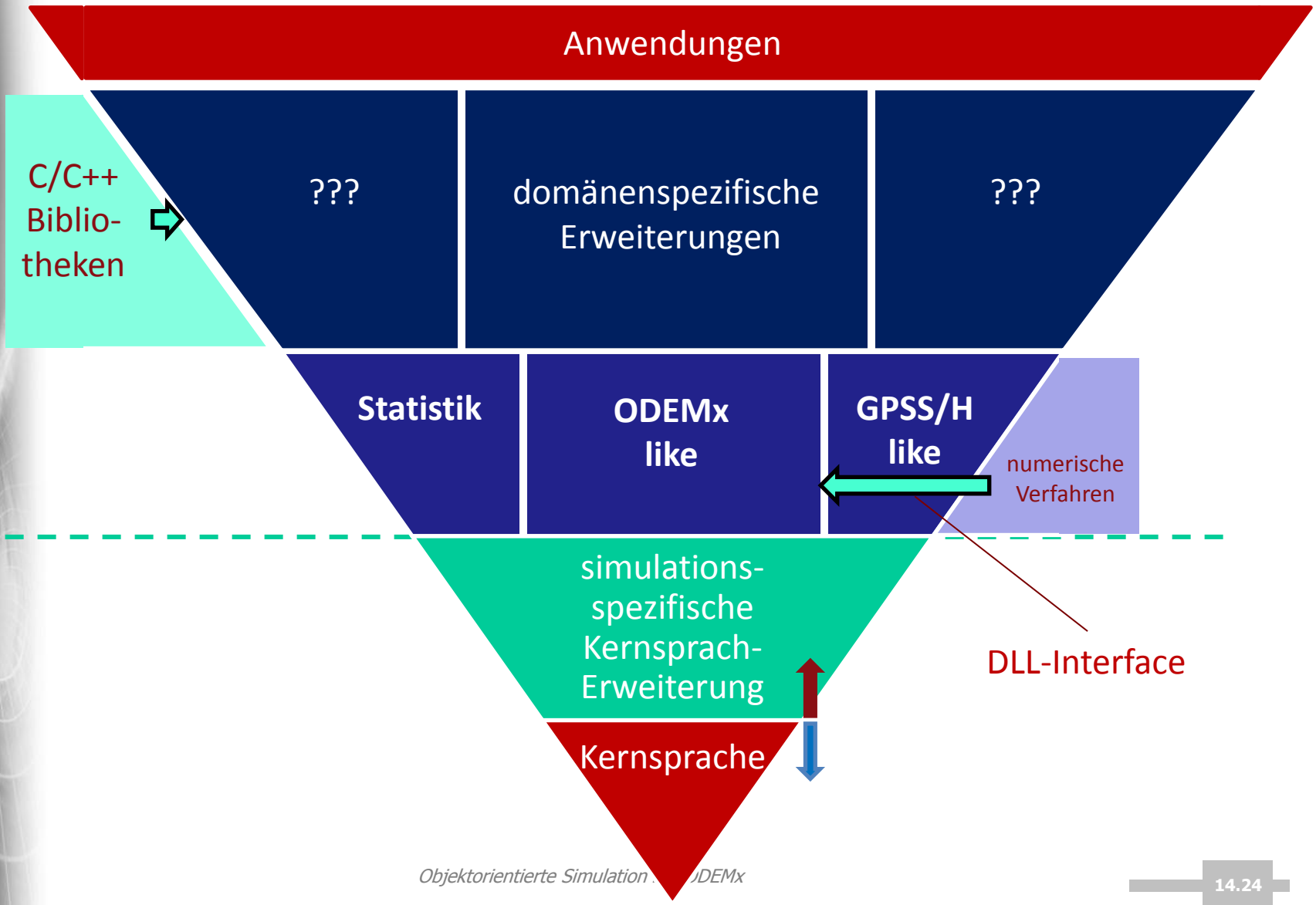
- universeller Datenaustausch über **formatierte Textdateien**

```
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)

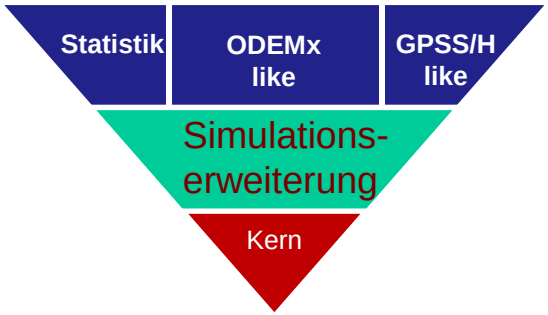
Die SLX-Modul-Pyramide



Viele Aha-Effekte bei der Prüfungsvorbereitung und hervorragende Resultate

Hintergrund: in Vorbereitung befindliche Projekte

- SLX- Deutsche Bahn AG
- SLX- VW
- Simulink-SLX (Windpark)



Wir suchen Interessenten für …
(mit Programmiererfahrung JAVA, C, …)

Begriffe (Prüfung)

- System, Systemumgebung, Systemelement, Kopplungsrelation, Rückkopplung, Zustandsgröße, Modellgröße, Zufallsgröße, Bewertungsgröße, Ein- und Ausgabegröße, Zustandsüberführungsfunktion, Ausgabefunktion, zeitdiskrete Systeme, zeitkontinuierliche Systeme, Teilsystem

- OO-Konzepte: Klasse, Objekt, Struktur, Verhalten, Beziehung: Objekt-Systemelement, Beziehung (Objekt, Prozess), Beziehung: Prozess-Aktivität-Ereignis

- OO-Abstraktion: Spezialisierung/Verallgemeinerung, Klassifikation/Exemplifikation, Zustandsautomat, Zustand, Zustandsübergang, asynchrone (Automaten-) Kommunikation, Trigger
(in UML-Semantik)

- Statistische Kennwerte (inkl. Erfassung und prinzipielle Berechnung)

- Original, Modell, Simulationsmodell, Simulator, Realzeit, Modellzeit, Simulationszeit

- Ereignis (Zeitereignis, Zustandsereignis), Next-Event-Simulation, Zusammenhang (Zufall-Pseudozufall), Verteilungsfunktion (empirische, theoretisch), antithetische Zufallsgrößen, Zusammenhang der Verteilungsfunktionen

Begriffe und Anwendungsbeispiele

- Modellklassen, die von ODEMx unterstützt werden (zeitdiskret, …)
- Scheduling-Mechanismen, bedingte und unbedingte Blockierungen
- Spezialisierungen: Sched- Event, Sched- Process, Sched-Process-Continuous
- Synchronsationskonzepte: WaitQ, CondQ, Bin, Res (und Template-Varianten) Memory (PortHead, PortTail, Cond, Timer)
- Unterbrechung von Warte- und Aktionsphasen eines Prozesses

Beispiele

- Fähre
- Tanker-Hafen-Tank-Raffenerie
- Tanker-Schlepper-Anlegeplatz-Gezeiten
- Bagger-Fahrzeuge-Beladung
- (Recycling-Hof)