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

2. Prinzip der Next-Event-Simulation

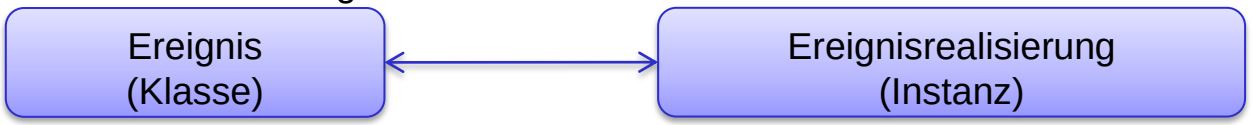
1. Charakterisierung der Next-Event-Simulation
   - Ereignisse, Next-Event-Scheduler
   - Barren-Beispiel
   - Zusammenhang von ereignisbasierter und prozessbasierter Modellbeschreibung

2. Umsetzung des Prinzips in ODEMx
   - Aufbau von ODEMx (erster Blick)
   - Triviales Clock-Beispiel

Diskrete Ereignissimulation (allgemein)

- **Verhaltensrealisierung**/ Verhaltensnachbildung eines Systems
  wird als chronologische Reihenfolge von Ereignissen verstanden

- **Ereignis**: Menge von Aktionen,
  die Systemzustandsänderungen zu einem (diskreten) Ereigniszeitpunkt bewirken,
  wenn denn das Ereignis tatsächlich eintritt



- **Bedingungen** für das Eintreten von Ereignissen
  - explizite Zeitangabe (→ Zeitereignisse)
    - Zeitpunkte determiniert
    - Zeitpunkte stochastisch
  - Zustandsbedingungen (→ Zustandsereignisse), Zeitpunkte ergeben sich implizit

- klassische **Implementierung**: sog. NEXT-EVENT-Methode
  (nicht äquidistante Sprünge von Ereigniszeitpunkt zu Ereigniszeitpunkt)

Basis für **Simulationssprachen** mit

- ereignis-orientierter
- aktivitäts-orientierter
- prozess-orientierter

Modellbeschreibung

Vertreter

- SLAM
- GPSS
- ODEMx, DESMO-J, SLX

Konzepte der diskreten Ereignissimulation

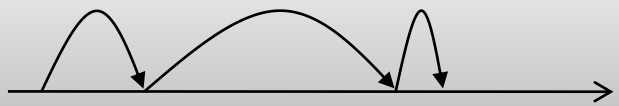
- SystemStruktur (Zustand)
  eine Menge von Variablen
  (ausgezeichnete Menge von Modellbeschreibungsgrößen)
  beschreibt in ihrer Belegung den Systemzustand zum aktuellen Zeitpunkt

- Systembewertungsgrößen
  (z.B. statistische Kenngrößen)

- Uhr
  (Modellzeit, dimensionslos, monoton wachsend)

- EL= Ereignisliste (Kalender)
  - Umgang mit Gleichzeitigkeit von Ereignissen
  - Zeitfortschritt:
    Bestimmung des nächsten Ereignisses im Kalender,
    Setzen der Uhr
    (aktuelle Modellzeit:= Ereigniszeit)

Zustandsänderungen finden immer nur zu diskreten Zeitpunkten statt



Ereignis-Charakteristik

Ereigniszeit

Ereignisroutine (Parameter: Referenzen zu Systemstruktur-komponenten {seq. Code}

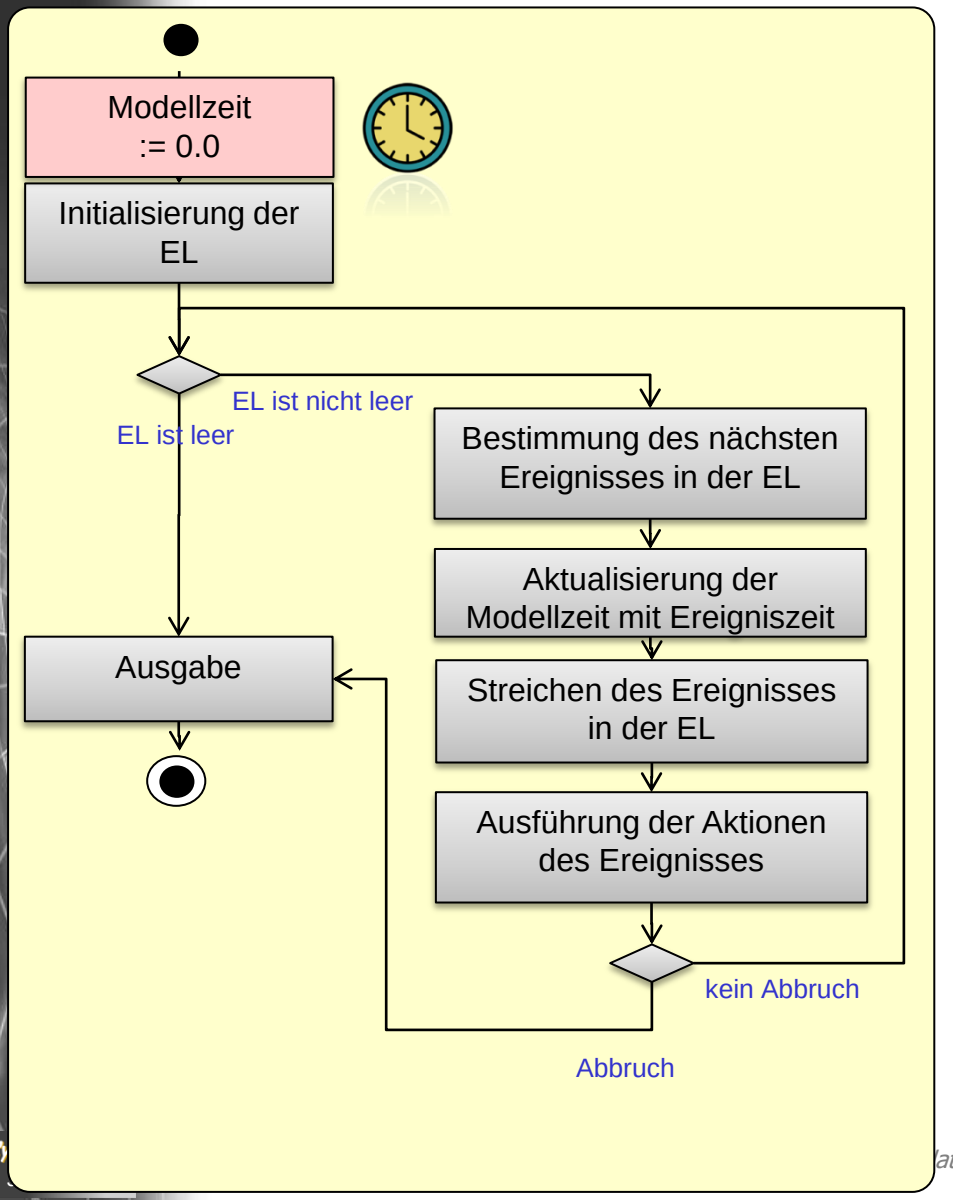
- Zustandsänderungen
- Planung weiterer Ereignisse

(Event) Scheduler

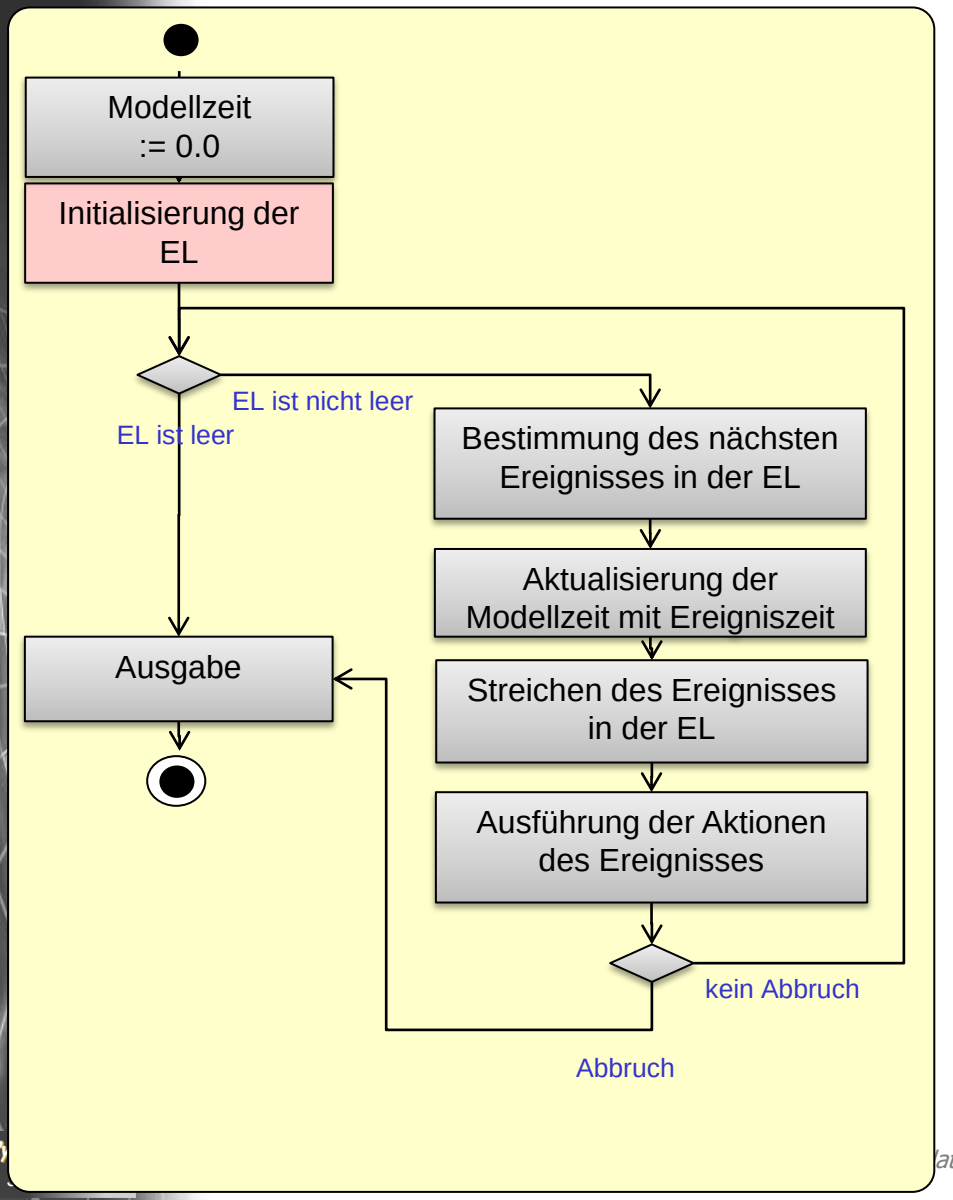
Zeitfortschritts-realisierung

Sortierung/ Realisierung von Ereignissen

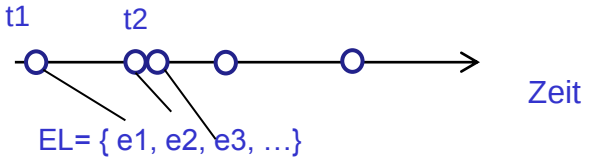
Trivialer Discrete-Event-Scheduler



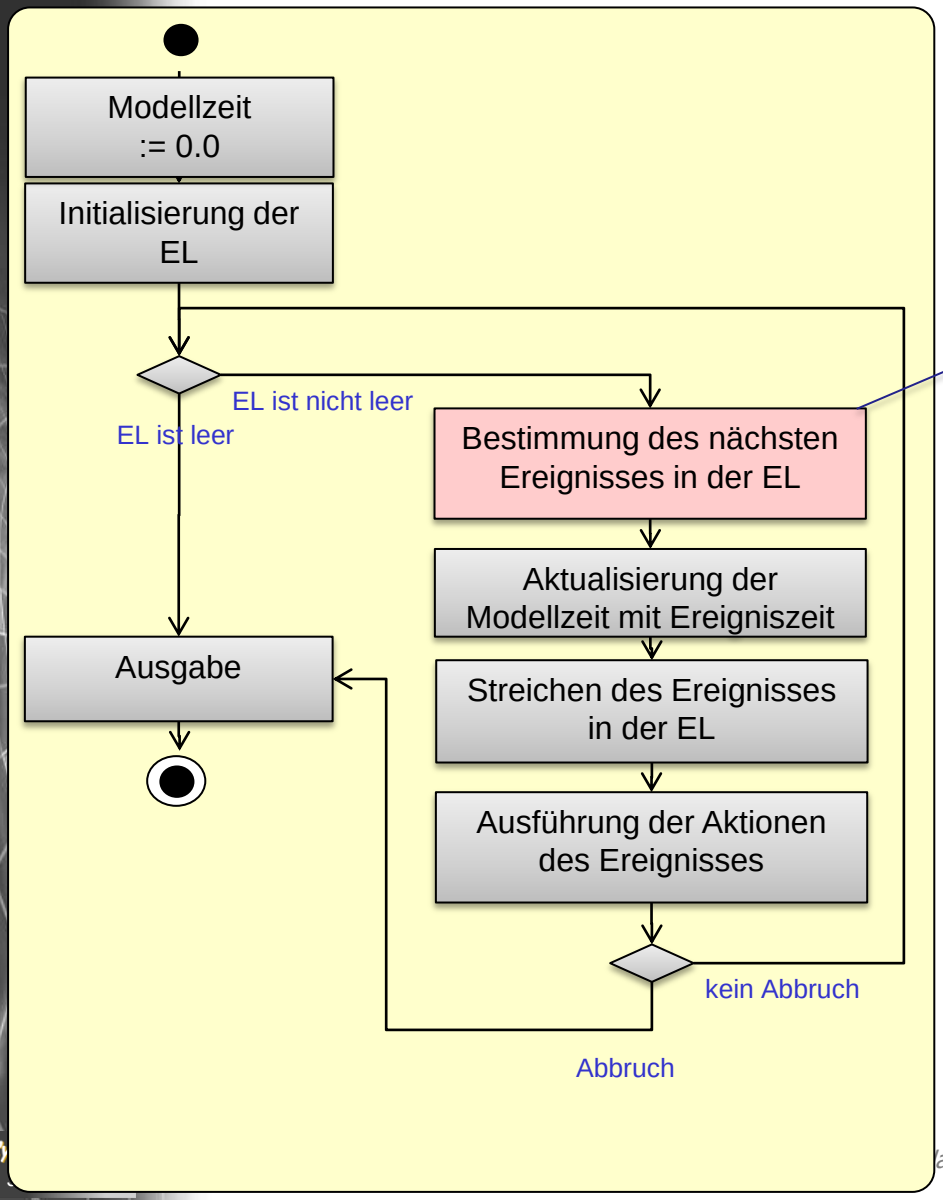
Trivialer Discrete-Event-Scheduler



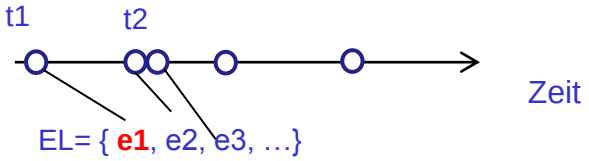
EL= Ereignisliste



Trivialer Discrete-Event-Scheduler

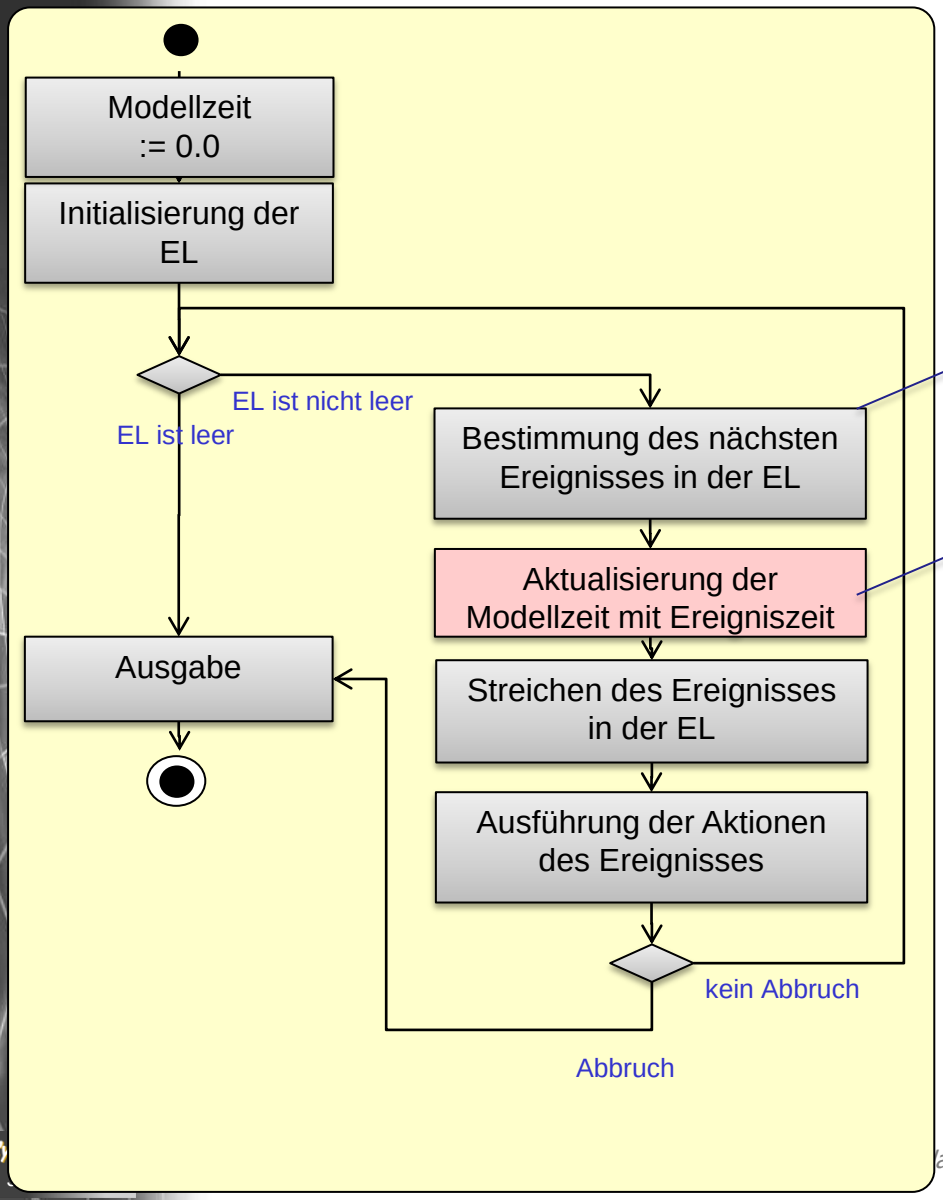


EL= Ereignisliste

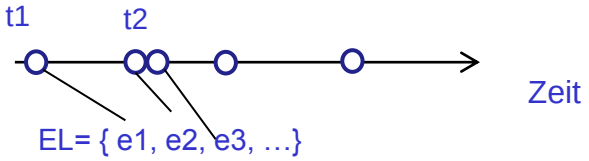


EL ist chronologisch sortiert

Trivialer Discrete-Event-Scheduler



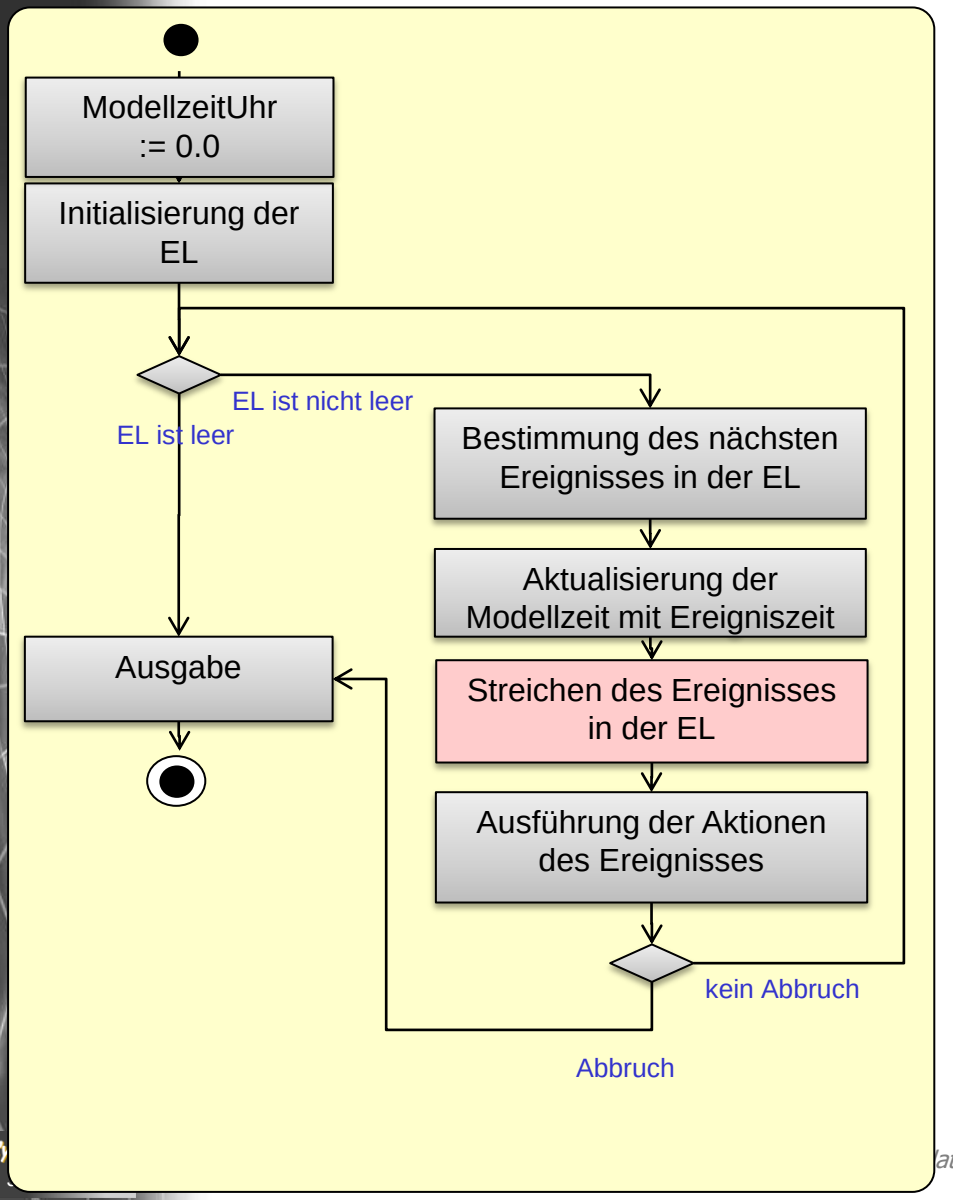
EL= Ereignisliste



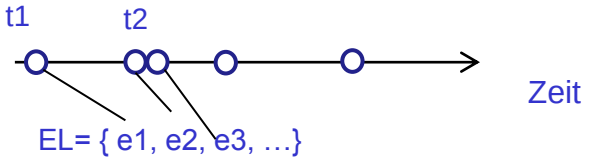
EL ist chronologisch sortiert

evtl. Zeitsprung (zeitdiskret)

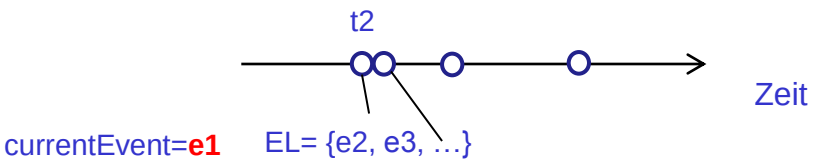
Trivialer Discrete-Event-Scheduler



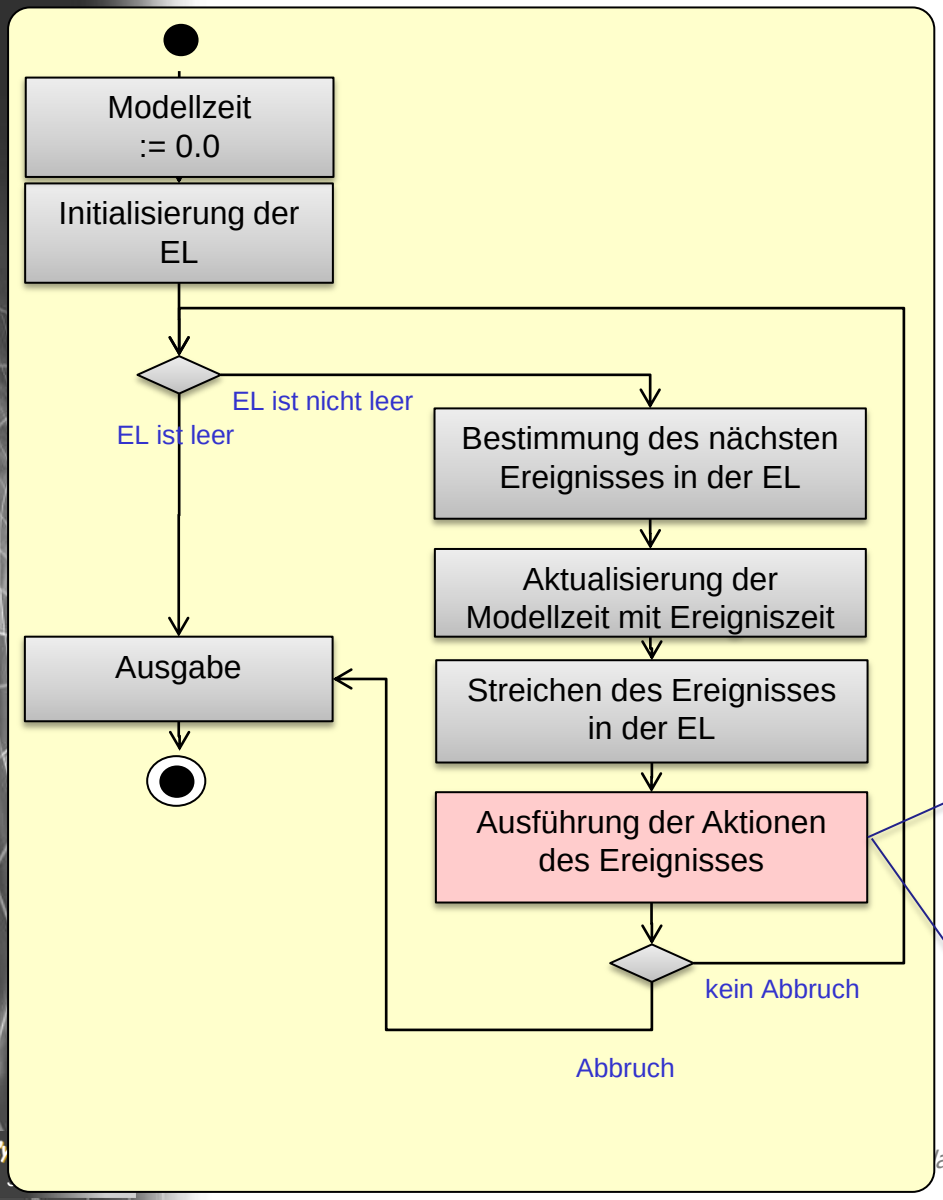
EL= Ereignisliste

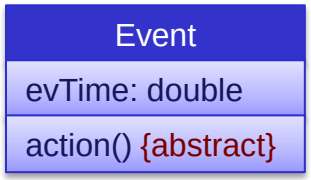


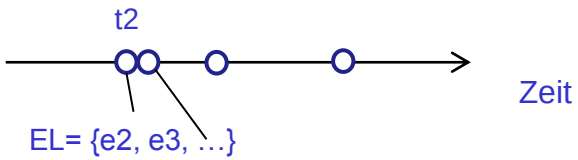
EL= Ereignisliste



Trivialer Discrete-Event-Scheduler







Zustandsänderungen

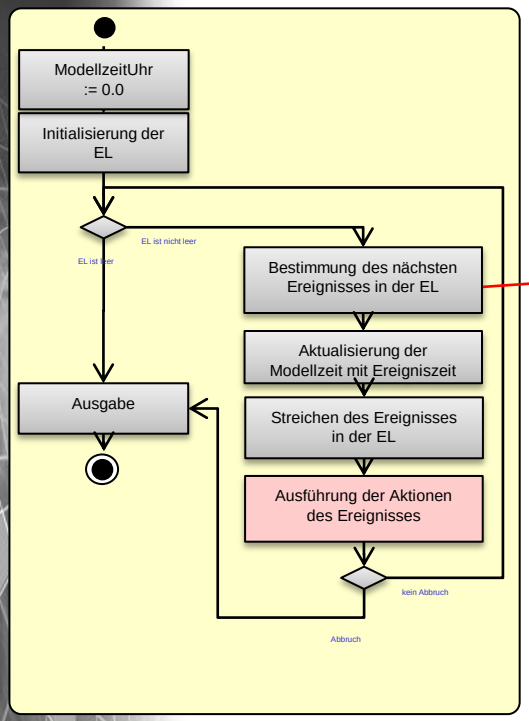
(Klassifizierung von Ereignisklassen, um partielle Zustandsänderungen strukturell zu unterstützen

Planung neuer Ereignisse

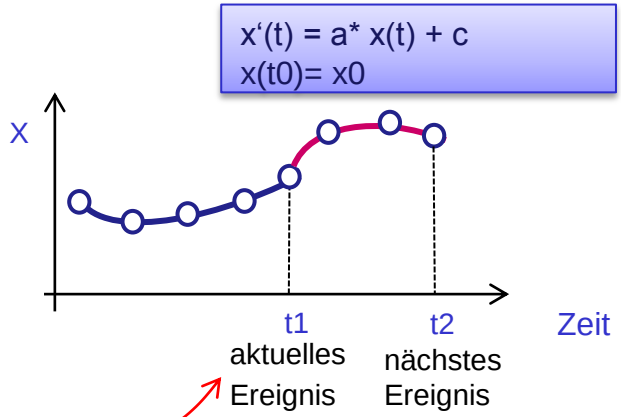
(Problem: Gleichzeitigkeit/ Zustandsereignis)

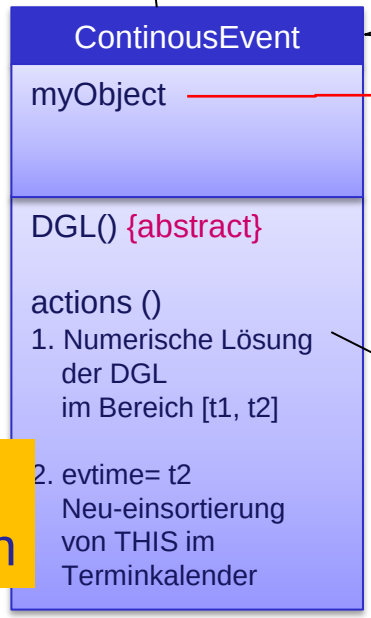
Trivialer Continues-Scheduler

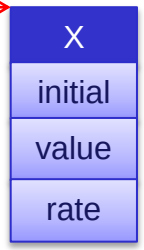
- als Spezialisierung eines Discrete-Event-Schedulers

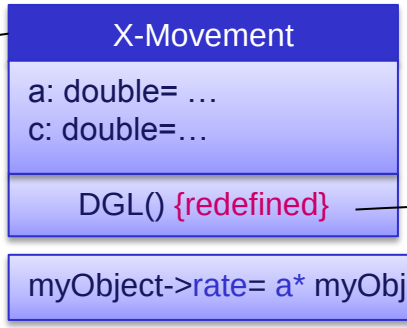


Sonderbehandlung von Continuous-Aktionen









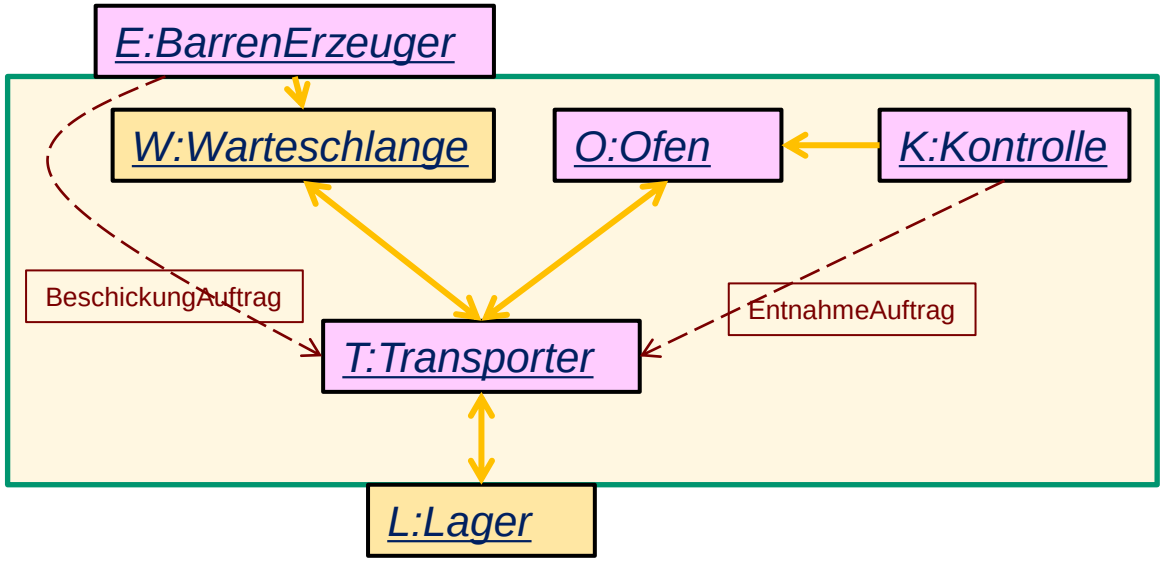
Alg. ist unabhängig von konkteter DGL

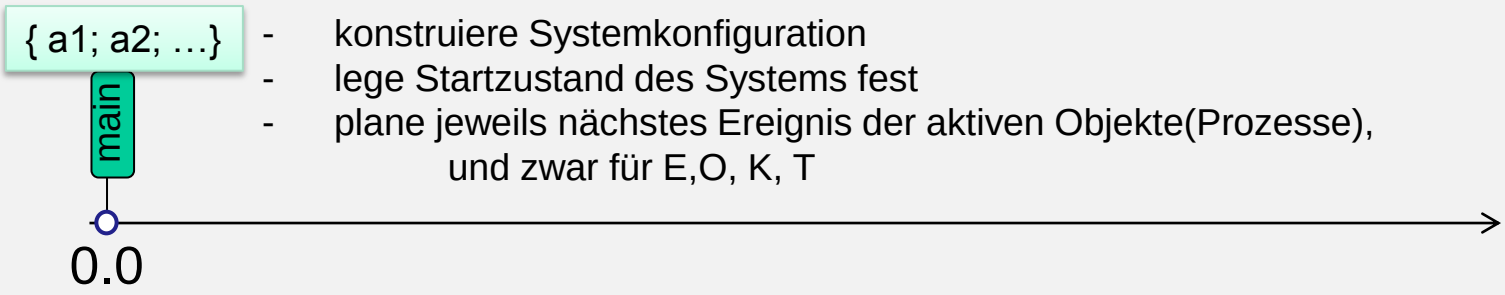
2. Prinzip der Next-Event-Simulation

1. Charakterisierung der Next-Event-Simulation
   - Ereignisse, Next-Event-Scheduler
   - Barren-Beispiel
   - Zusammenhang von ereignisbasierter und prozessbasierter Modellbeschreibung

2. Umsetzung des Prinzips in ODEMx
   - Aufbau von ODEMx (erster Blick)
   - Triviales Clock-Beispiel

Prinzipieller Ablauf am Beispiel (1)

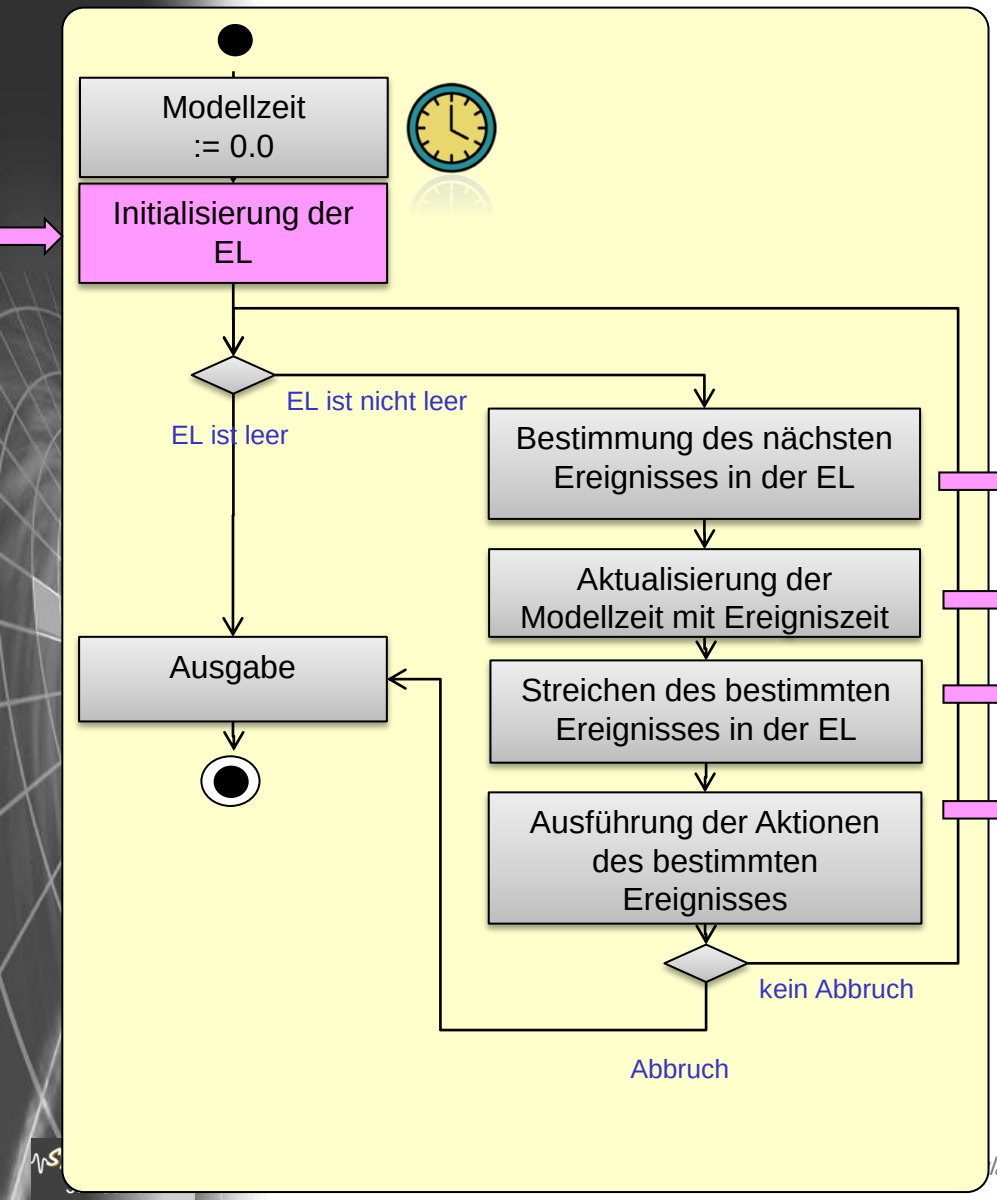




- konstruiere Systemkonfiguration
- lege Startzustand des Systems fest
- plane jeweils nächstes Ereignis der aktiven Objekte(Prozesse), und zwar für E,O, K, T

EL: Liste von Ereignissen (Kalender),
sortiert nach Zeitpunkten in der Modellzeit

Trivialer Discrete-Event-Scheduler (1)



Ur-Ereignis (main)

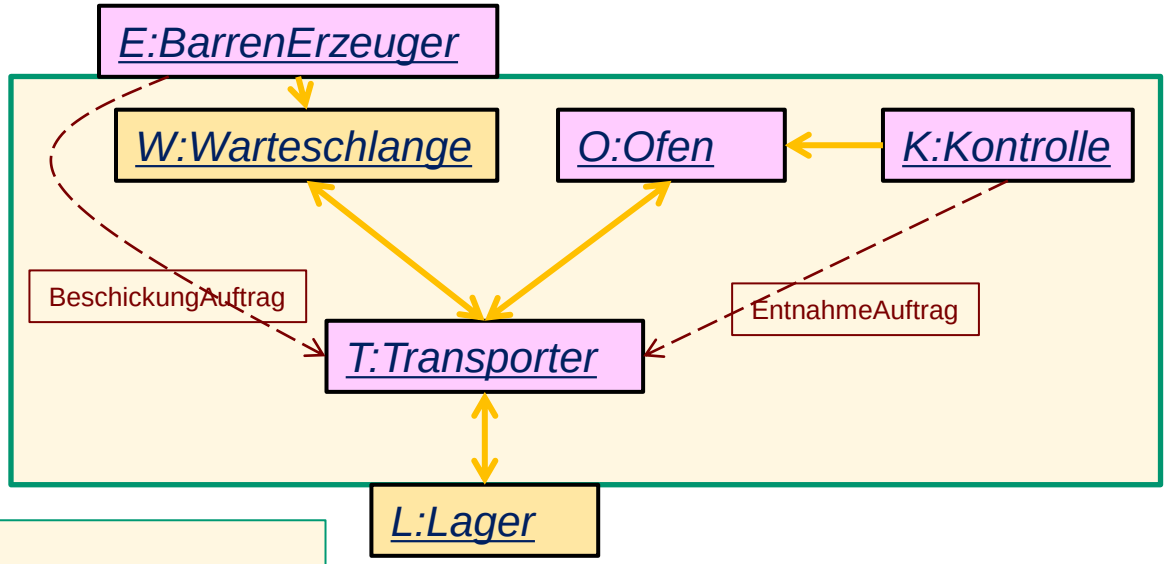
time= 0.0

EL= {} /* leer */

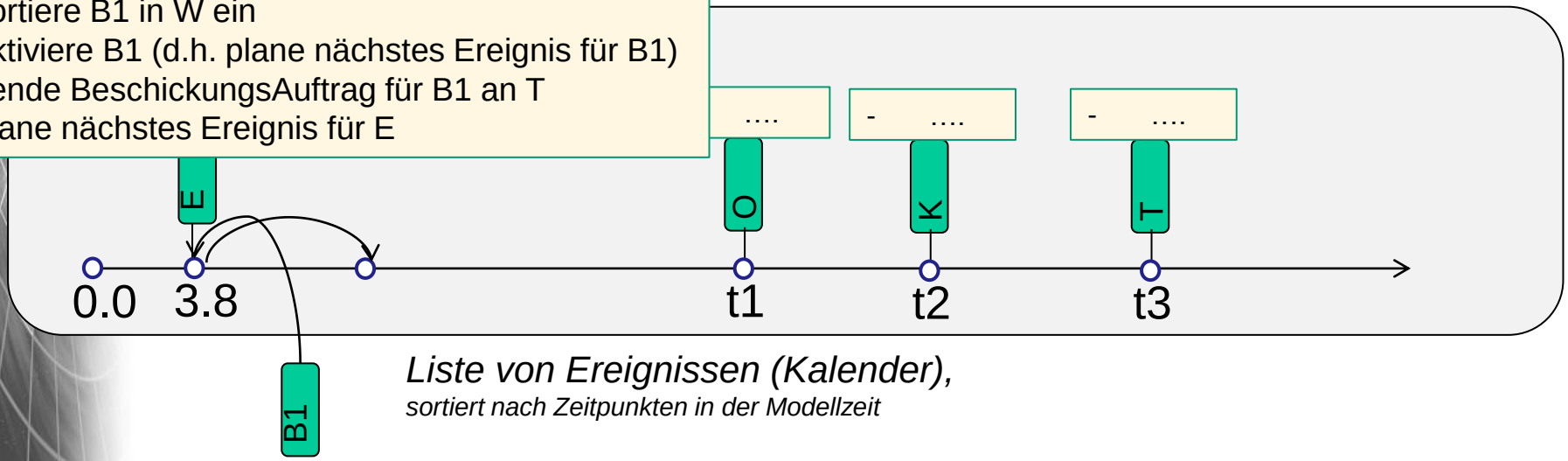
main

- konstruiere Systemkonfiguration
- lege Startzustand des Systems fest
- plane jeweils nächstes Ereignis der aktiven Objekte (Prozesse), und zwar für E,O, K, T

Prinzipieller Ablauf am Beispiel (2)

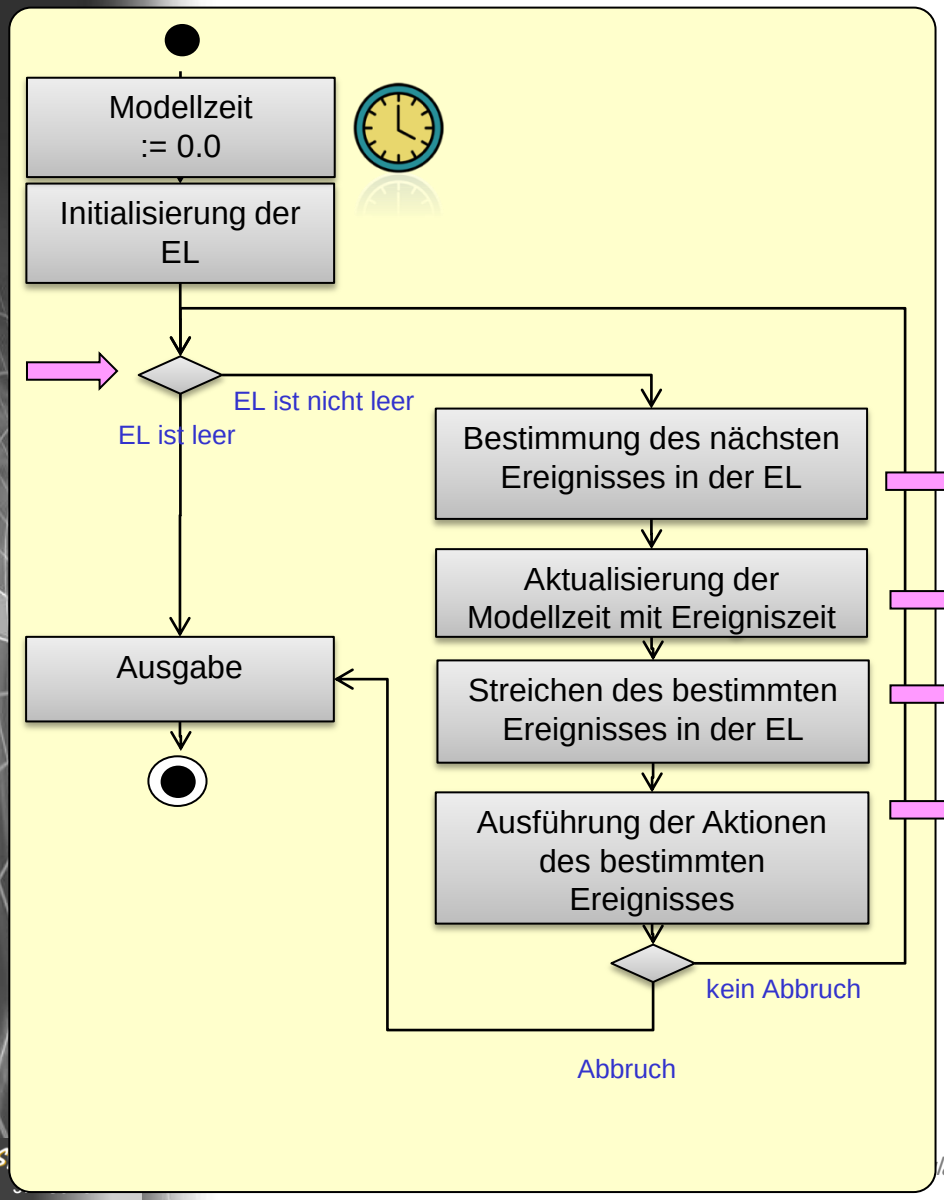


- erzeuge Barren-Objekt: B1
- sortiere B1 in W ein
- aktiviere B1 (d.h. plane nächstes Ereignis für B1)
- sende BeschickungsAuftrag für B1 an T
- plane nächstes Ereignis für E



Liste von Ereignissen (Kalender),
sortiert nach Zeitpunkten in der Modellzeit

Trivialer Discrete-Event-Scheduler (2)



E-Ereignis

Modellzeit= 3.8

EL= { x } /* nicht leer */

E-Ereignis

- erzeuge Barren-O: O1
- sortiere O1 in W ein
- aktiviere O1 (plane nächstes Ereignis für O1)
- sende BeschickungsAuftrag für O1 an T
- plane nächstes Ereignis für E

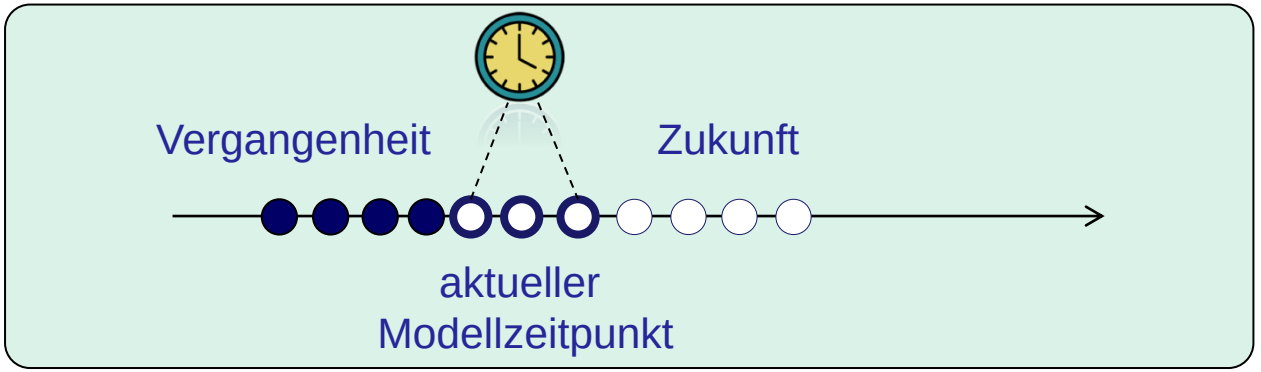
USW.

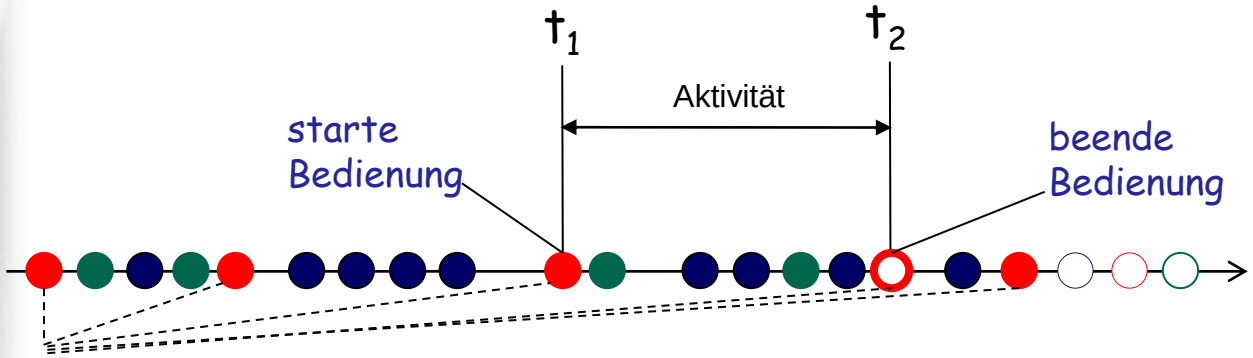
2. Prinzip der Next-Event-Simulation

1. Charakterisierung der Next-Event-Simulation
   - Ereignisse, Next-Event-Scheduler
   - Barren-Beispiel
   - Zusammenhang von ereignisbasierter und prozessbasierter Modellbeschreibung

2. Umsetzung des Prinzips in ODEMx
   - Aufbau von ODEMx (erster Blick)
   - Triviales Clock-Beispiel

Ereignis, Aktivität, Prozess: Zusammenhang





Ereignisfolge für eine konkrete Entity
(Objekt einer aktiven Klasse)

➔ Ereignisfolge wird als Prozessrealisierung aufgefasst

Wir suchen eine Verhaltensbeschreibungen / Prozessbeschreibungen für **Objektklassen**

Ereignisse für andere Objekte

Ereignis, Aktivität, Prozess: Zusammenhang

Ereignis

- Zeitpunkt
  oder
- Zustandsbedingung

- Aktionsfolge

  {A1, A2, …}

- sequentiell
- ohne Zeitverbrauch
- operieren über globalen Systemzustand oder spezielle Objekte
- planen /streichen neue Ereignisse

realisieren zeitdiskrete Zustandsänderungen

Aktivität

- zwei zusammenhängende Ereignisse über demselben Objekt
  (Start, Ende)

- zeitliche Dauer ist explizit oder als Zustandsbedingung gegeben

realisieren zeitdiskrete und zeitkontinuierliche Zustandsänderungen

Prozess

- zusammenhängende Folge von Ereignissen über demselben Objekt

- zwei ausgezeichnete Ereignisse
  - Start und
  - Ende der Lebenszeit des Objektes

- zeitliche Dauer einzelner Aktivitäten ist explizit oder als Zustandsbedingung gegeben
- Synchronisationen zwischen Prozessen bei Kooperationen lassen sich gut beschreiben

realisieren zeitdiskrete u. zeitkontinuierliche Zustandsänderungen

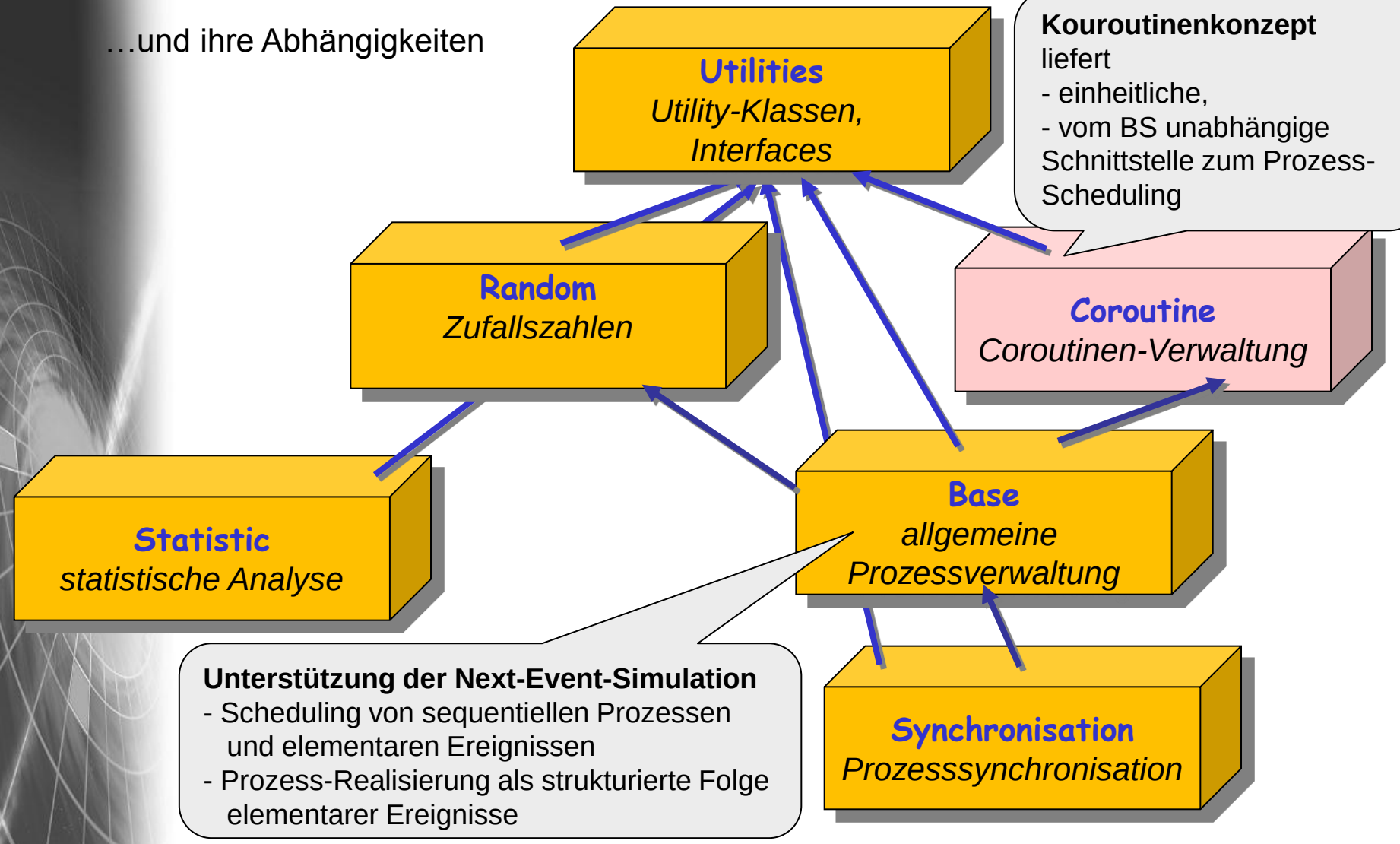
2. Prinzip der Next-Event-Simulation

1. Charakterisierung der Next-Event-Simulation
   - Ereignisse, Next-Event-Scheduler
   - Barren-Beispiel
   - Zusammenhang von ereignisbasierter und prozessbasierter Modellbeschreibung

2. Umsetzung des Prinzips in ODEMx
   - Aufbau von ODEMx (erster Blick)
   - Triviales Clock-Beispiel

Die ODEMx-Module

…und ihre Abhängigkeiten

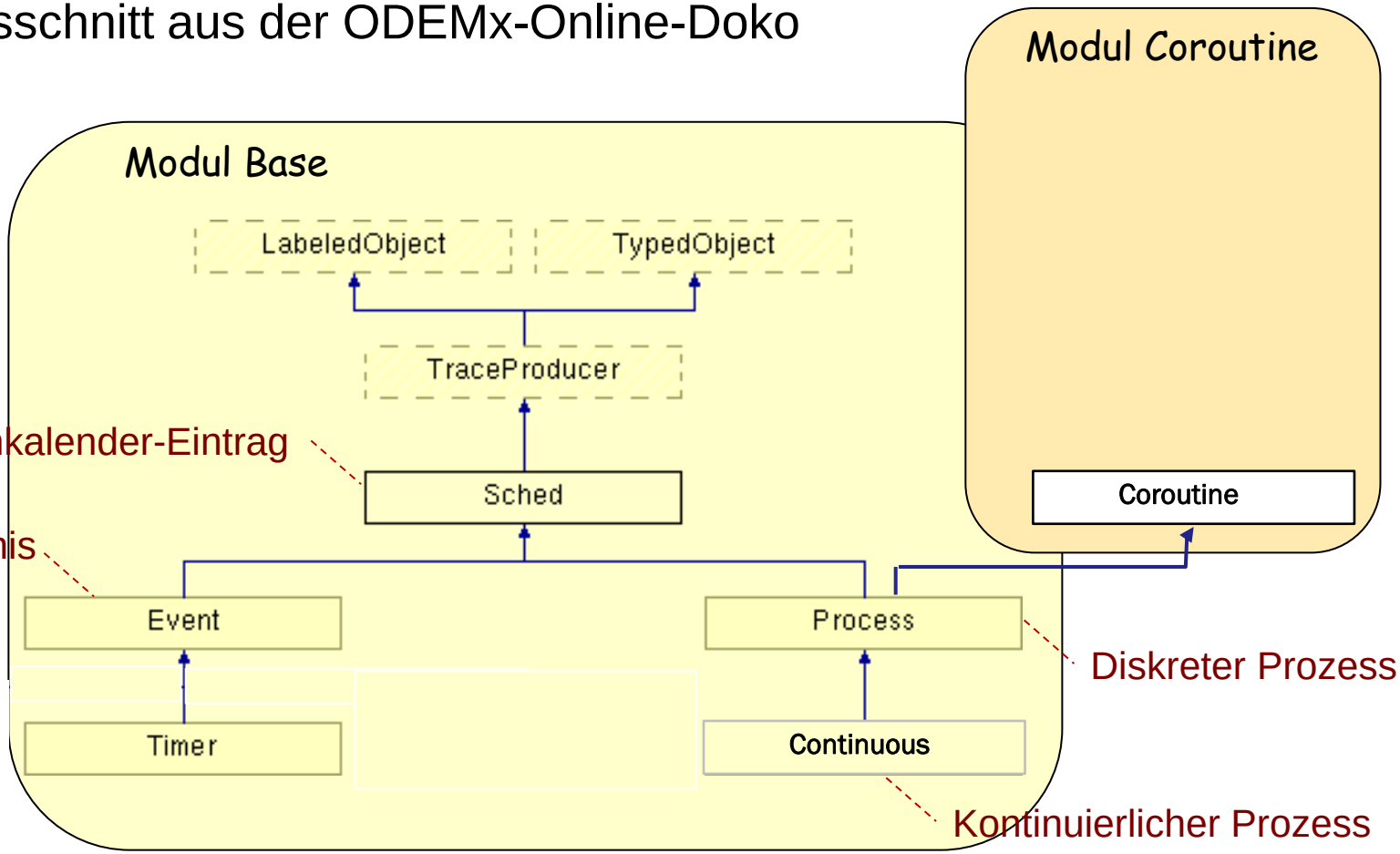


Entwurfsmaxime für die ODEMx- Bibliothek

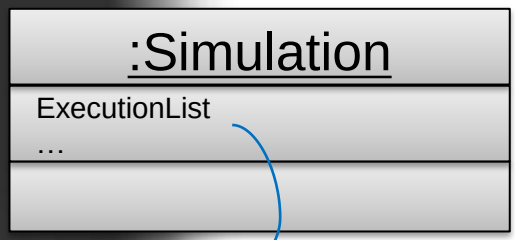
- **Basis**: ältere ODEM-Bibliothek
- (relativ) **leichte Benutzbarkeit**:
  - automatische Beobachtung, Sammlung von Modellwerten und deren statistische Auswertung
  - vordefinierte Synchronisationsmechanismen
- **Prozess-Unterstützung**:
  - ein C++ (Haupt-)Programm erlaubt nebenläufige (unabhängige oder abhängige) System-Simulationen als Komponenten (sog. Simulationskontexte),
  - jeder Simulationskontext verwaltet ein eigenständiges Ensemble pseudo-paralleler Prozesse mit individueller Stack-Verwaltung, die in Abhängigkeit vom jeweiligen Modellzeitverbrauch alternierend (als Koroutinen) ausgeführt werden
  - jeder Simulationskontext verfügt dafür über einen individuellen Prozess-Ereigniskalender
- **Standardfall**:
  es gibt genau einen (Default-)Simulationskontext, der (nur) ein Ensemble pseudo-paralleler Prozesse verwaltet, und nur einen einzigen Prozess-Ereigniskalender besitzt
- neue **Version** der ODEMx-Bibliothek
  - erlaubt u.a. die laufzeiteffizientere Modellierung einzelner zeitdiskreter Ereignisse neben Prozessen (Mix)

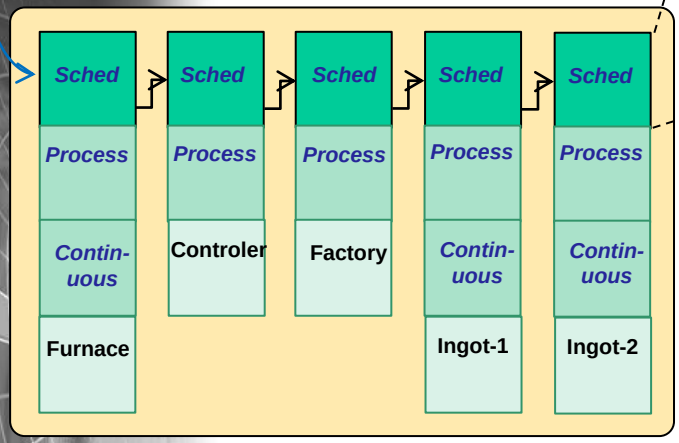
Klassenvererbungshierarchie (Ausschnitt)

Ausschnitt aus der ODEMx-Online-Doko



Timer als Ereignis
(nicht als Prozess !)





Sched

#execute()=0
+getExecutionTime() const =0
+getPriority() const =0
+getSchedType() const
+getTime() const
+isScheduled() const
+Sched (Simulation &sim, …)
+setExecutionTime(SimTime time)=0
+setPriority(Priority newPriority)=0
+~Sched ()

abstrakte passive Klasse

Abteilung der **Attribute**
leer: nicht genannt/vorhanden

Abteilung der **Operationen**

Sichtbarkeit: +,-,#,~

Konstruktor sorgt für Zuordnung zu einem Simulationskontext

Process

- ProcessState processState_;
- Priority priority_;
- SimTime executionTime_;

#virtual int main() = 0;
+setExecutionTime(SimTime time);
#void execute();
+void holdFor (SimTime *t*)

abstrakte aktive Klasse

Grundzustände:
CREATED, CURRENT,
RUNNABLE, IDLE, TERMINATED

Priorität:
Gleichzeitigkeitskonfliktbehebung bei der
Sequentialisierung von Ereignissen

Ereigniszeit:
Basis der Sortierung des Terminkalenders (ExecutionList)
SimTime-Typ kann eingestellt werden

Abstrakter Lebenslauf: main
redefinition von execution
(Fortsetzung im Lebenslauf)

Zeitverbrauch (Terminkalender)

Konkrete aktive Klasse

Factory

-SimTime dt
-Ingot* ing

#int main() {
holdFor (dt);
ing= new Ingot (…);
}

Konkreter Lebenslauf

```
#ifdef ODEMX_USE_CONTINUOUS typedef double SimTime #else typedef Poco::UInt64 SimTime; #endif
```

Prozessverwaltung

- realisiert über Simulationskontext- Funktionalität
  - Verwaltung von zustandsabhängigen Prozesslisten und des Ereigniskalenders
  - Erfassung sämtlicher Objekte (Zufallszahlengeneratoren, Synchronisationsobjekte, …) passiver Klassen

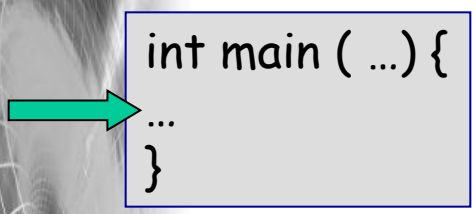
  ➔ stellt sämtliche **Kontextinformation** für die Simulation eines Systems/Teilsystems bereit

einfache Prozess- und Ereignisverwaltung
durch Nutzung eines
DefaultSimulation-Objektes

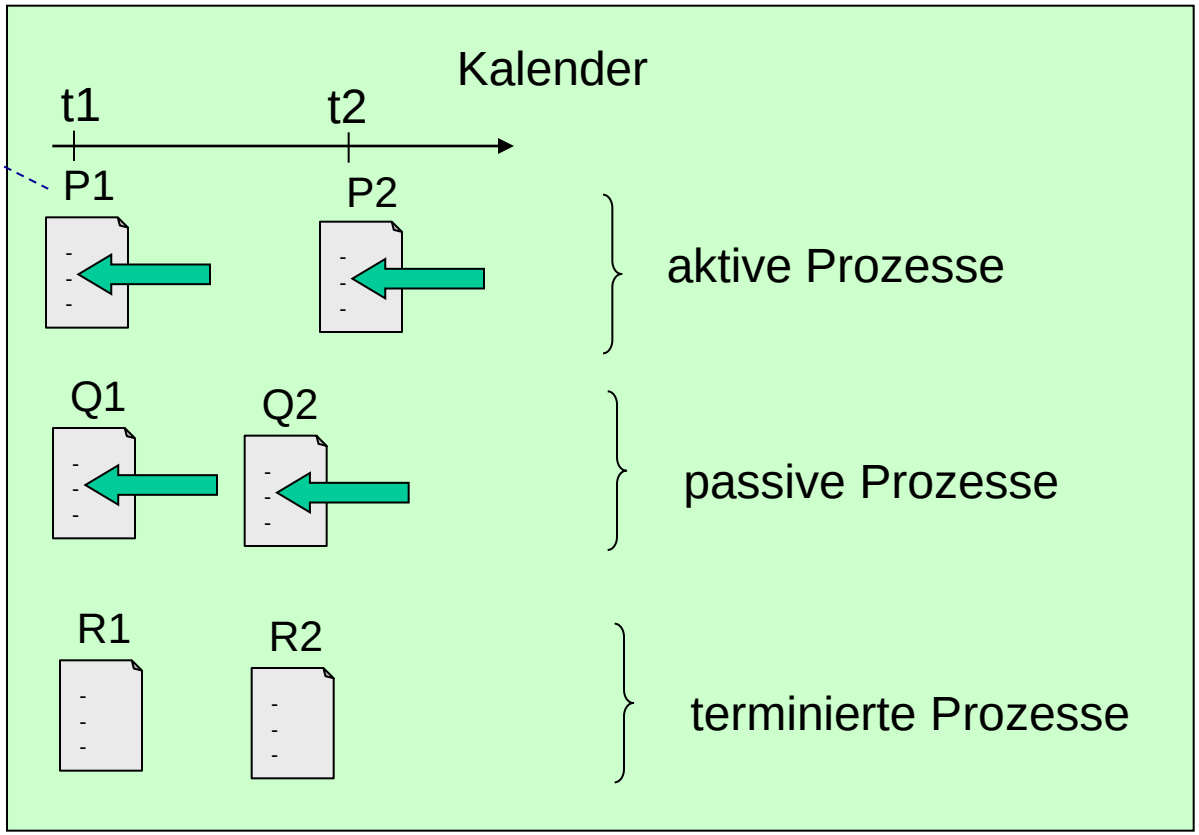
hierarchische Prozess- und Ereignisverwaltung
parallele Prozessverwaltung von Teilsystemen
durch Nutzung von Objekten von
nutzereigenen Simulation-Ableitungen
(ein Objekt pro Ableitung)

Prozessverwaltung in ODEMx

Klasse Process
verfügt
über eine virtuelle
Memberfunktion main
(Lebenslauf des Prozesses)



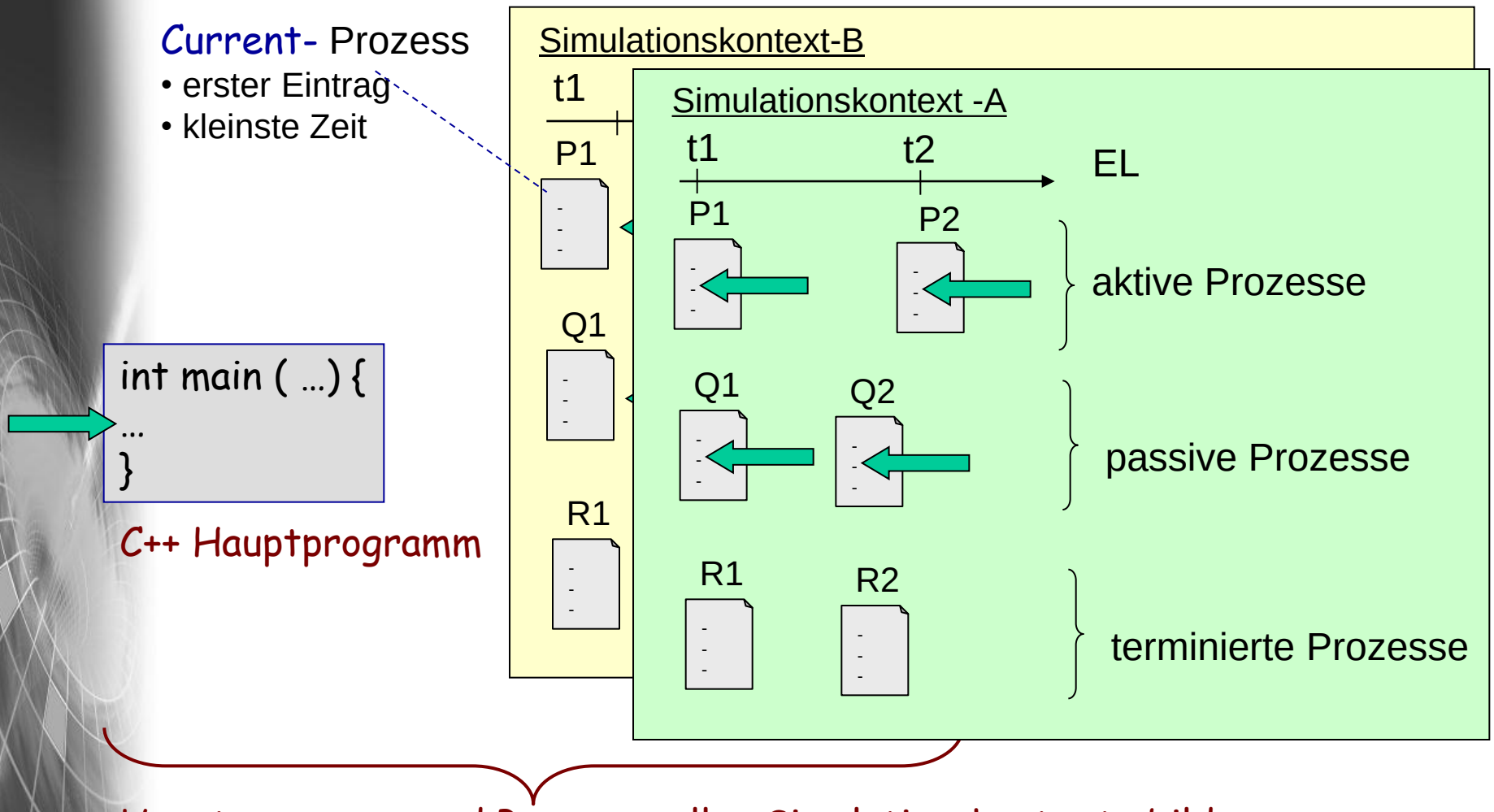
C++ main program



Ensemble von main()-Funktionen aller existenten Prozess-Objekte wird zusammen mit eigentlichem C++ Hauptprogramm als Ensemble von Coroutinen quasiparallel ausgeführt

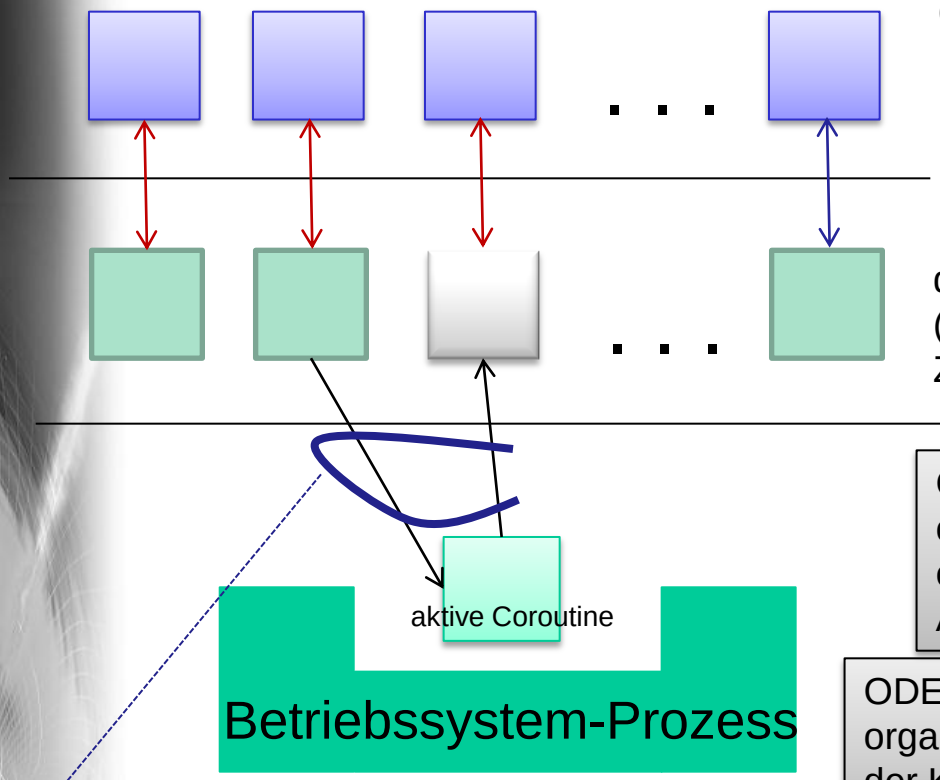
Grundidee einer hierarchischen Prozessverwaltung

Current- Prozess
- erster Eintrag
- kleinste Zeit



Hauptprogramm und Prozesse aller Simulationskontexte bilden
ein hierarchisches Coroutinensystem (auf einer Ein-Prozessor-Maschine)

Universelle ODEMx- Urvariante



dynamisches Ensemble von realen/gedachten Prozessen (zeitskret/zeitkontinuierlich) (Original)

Struktur-Äquivalenz (1:1)

dynamisches Ensemble von ODEMx-Prozessen (Simulationsmodell)
Zustand jedes Prozesses auf der Halde

ODEMx-Laufzeitsystem
organisiert Scheduling der ODEMx-Prozesse entsprechend ihrem Modellzeitverbrauch für Änderung ihrer Zustandsgrößen

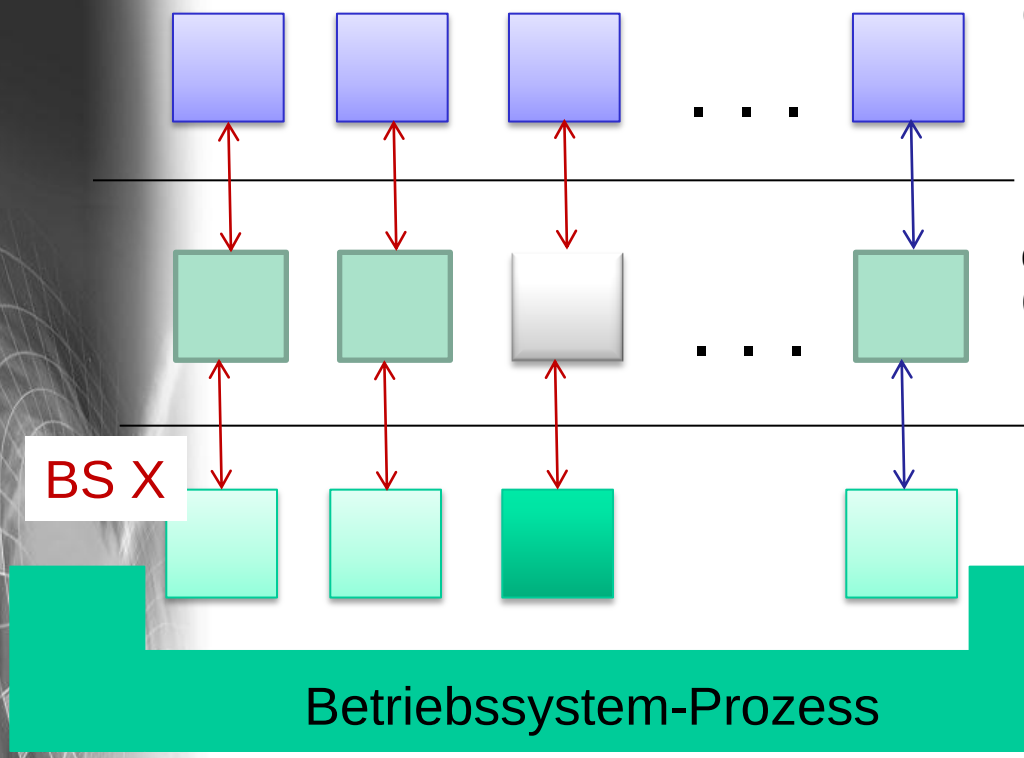
ODEMx-Laufzeitsystem
organisiert Speicherung und Restaurierung der Koroutinen-Zustände (Laufzeitkeller, Register)

rechenzeitaufwändiger

… im Vergleich zu prozeduralen Next-Event-Verfahren (Scheduler Anfang der Vorlesung)

aber hoher Grad an einer (auf natürliche Weise erreichbaren) Strukturäquivalenz von Modell und Original

Laufzeitverbesserte Varianten



dynamisches Ensemble von realen/gedachten Prozessen (zeitdiskret/zeitkontinuierlich) (Original)

Struktur-Äquivalenz (1:1)

dynamisches Ensemble von ODEMx-Prozessen (Simulationsmodell)

Struktur-Äquivalenz (1:1)

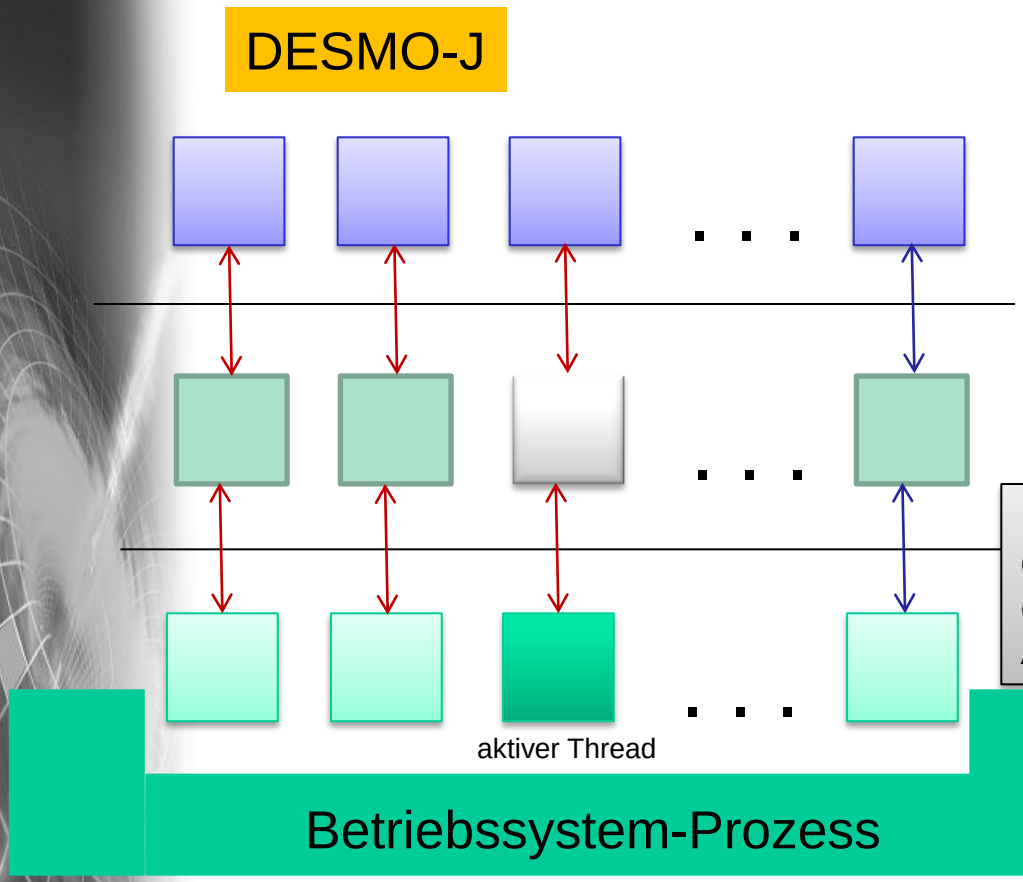
ODEMx-Laufzeitsystem
organisiert Scheduling der ODEMx-Prozess- entsprechend ihrem Modellzeitverbrauch für Änderung ihrer Zustandsgrößen

X= Fibres, MS Windows

X= uContext, Linux

ODEMx-Laufzeitsystem
organisiert Umsetzung von Zeigern zu Koroutinen-Zustände (Laufzeitkeller) und speichert und restauriert lediglich Register

Java-basierte Simulationsbibliothek (DESMO-J)



dynamisches Ensemble von realen/gedachten Prozessen (zeitdiskret/zeitkontinuierlich) (Original)

Struktur-Äquivalenz (1:1)

dynamisches Ensemble von DESMO-J-Prozessen (Simulationsmodell)

DESMO-J-Laufzeitsystem
organisiert Scheduling der DESMO-J-Prozesse entsprechend ihrem Modellzeitverbrauch für Änderung ihrer Zustandsgrößen

aber laufzeitschlechter als ODEMx-Urvariante (schwergewichtigere Lösung)

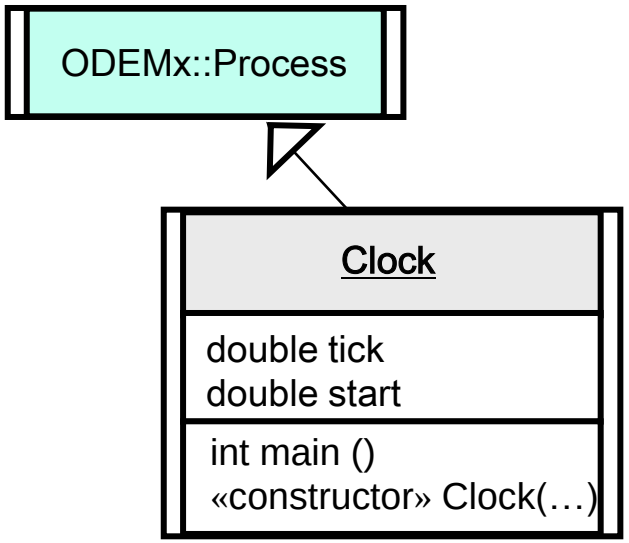
2. Prinzip der Next-Event-Simulation

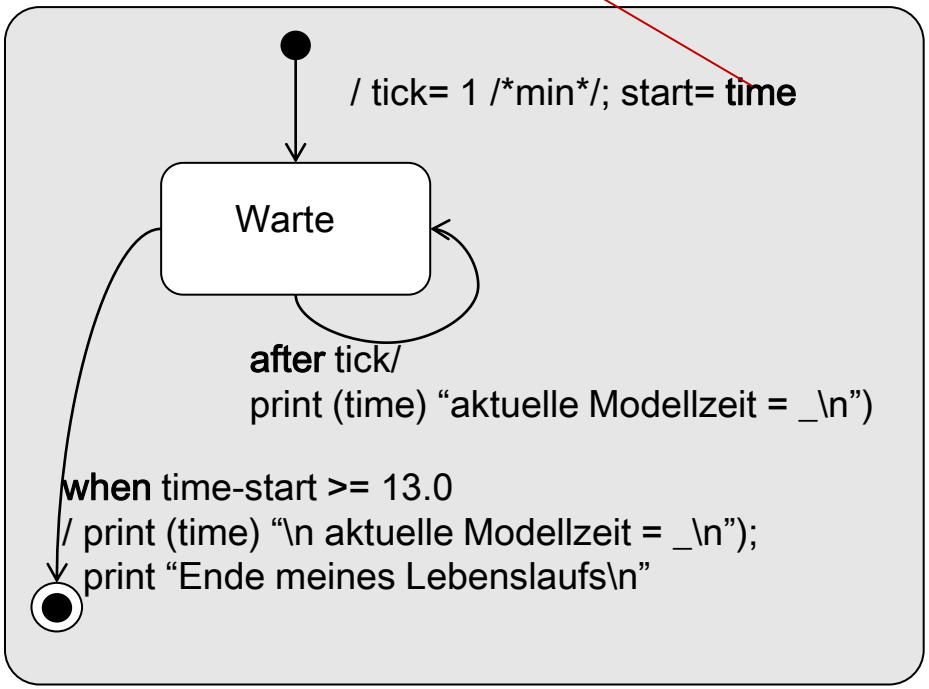
1. Charakterisierung der Next-Event-Simulation
   - Ereignisse, Next-Event-Scheduler
   - Barren-Beispiel
   - Zusammenhang von ereignisbasierter und prozessbasierter Modellbeschreibung

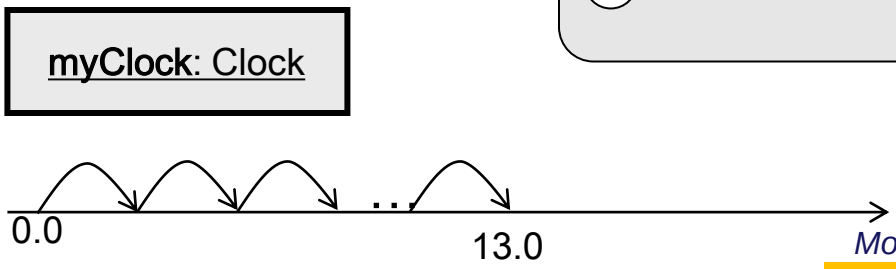
2. Umsetzung des Prinzips in ODEMx
   - Aufbau von ODEMx (erster Blick)
   - Triviales Clock-Beispiel

Nachbildung einer Uhr

aktuelle Modellzeit
zum Generierungs-/Startzeitpunkt
des jeweiligen Clock-Objektes







Ausgabe eines Punkt-Zeichens (als Tick)
zu ganzzahligen Modellzeitpunkten
als Prozess (der Ereignisfolge erzeugt)

System-Erweiterung:
zwei Uhren werden nicht synchron gestartet: Ausgabe einer überlagerten farblich markierten Punktfolge