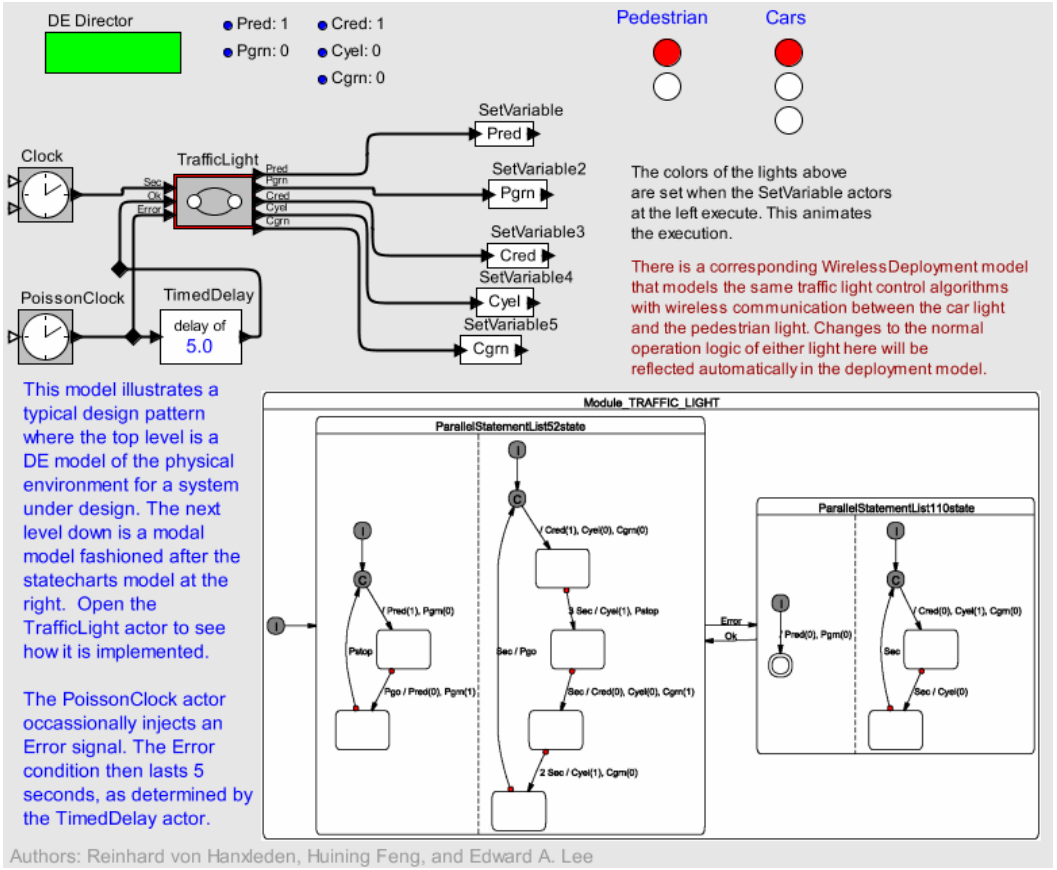
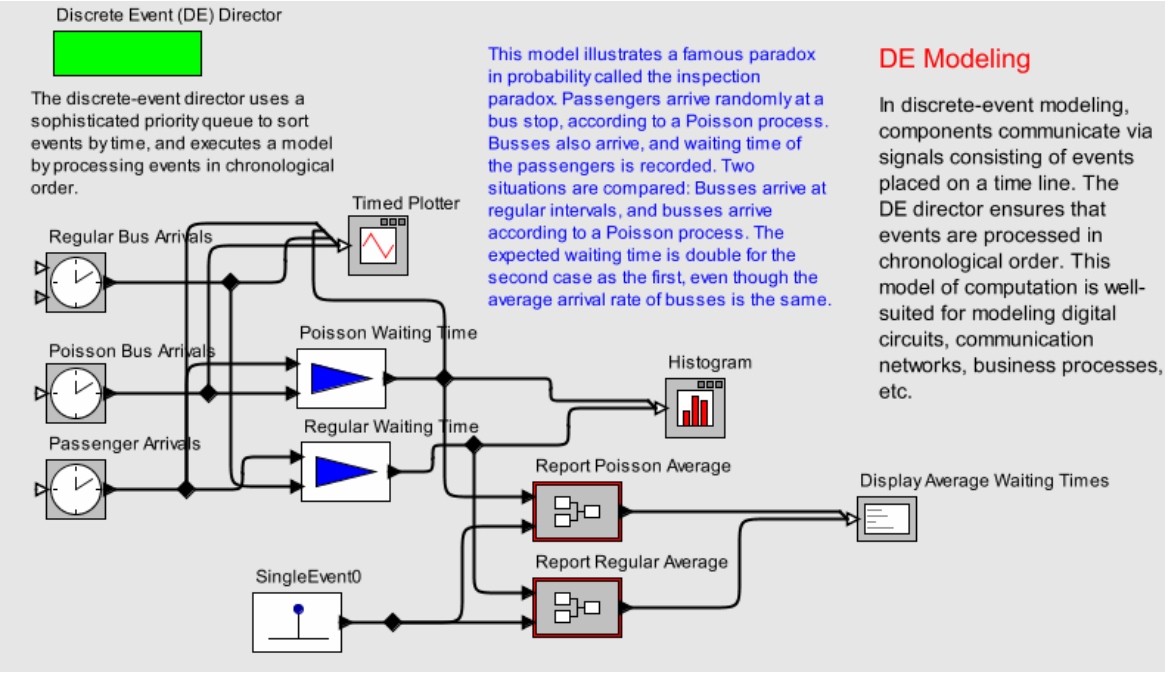
Example Ptolemy Model of Comp.: Synchronous Reactive

- Prinzip:
  - Annahme: unendlich schnelle Maschine
  - Diskrete Ereignisse (DE) werden zyklisch verarbeitet (Ereignisse müssen nicht jede Runde eintreffen)
  - Pro Runde wird genau eine Reaktion berechnet
  - Häufig verwendet in Zusammenhang mit Finite State Machines
- Vorteile:
  - einfache formale Verifikation
- Werkzeuge:
  - Esterel Studio
  - Scade



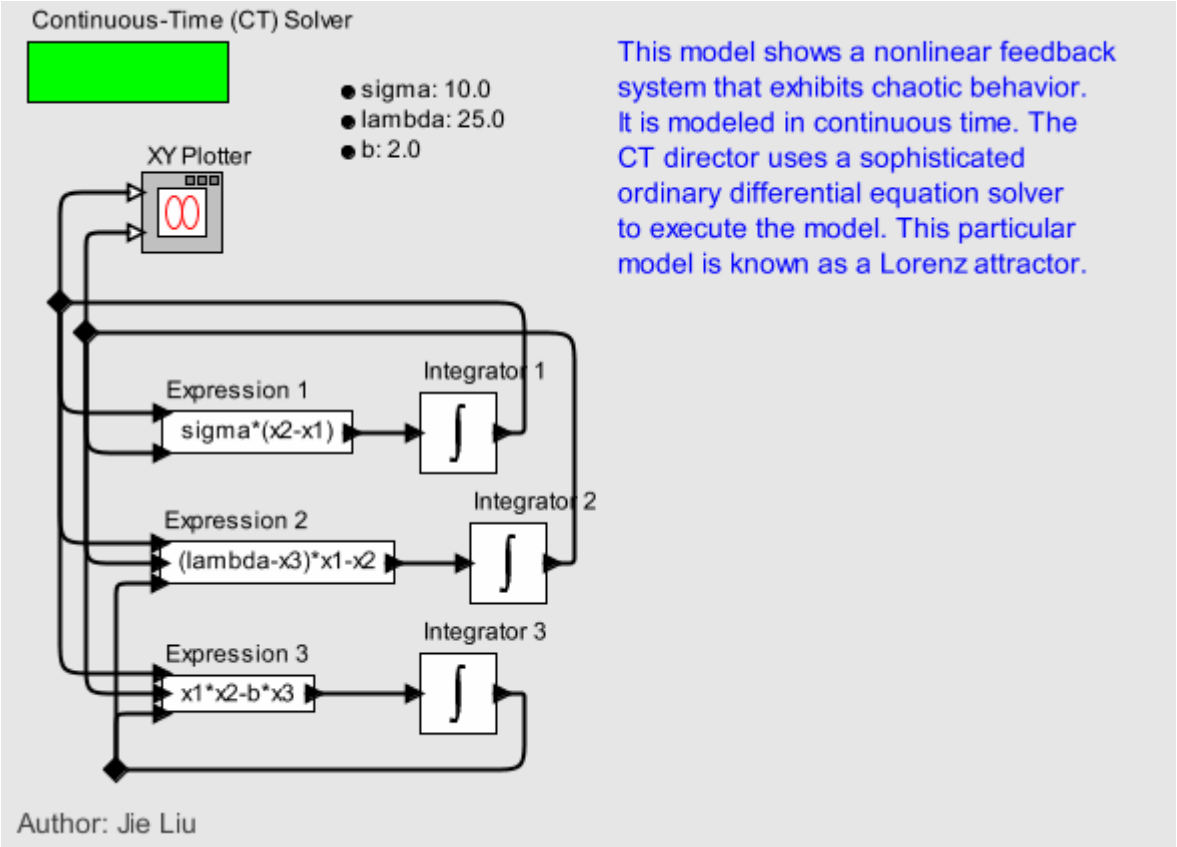
Example Ptolemy Model of Comp.: Discrete Event

- Prinzip:
  - Kommunikation über Ereignisse
  - Jedes Ereignis trägt einen Wert und einen Zeitstempel
- Anwendungsgebiet:
  - Digitale Hardware
  - Telekommunikation
- Werkzeuge:
  - VHDL
  - Verilog
- Varianten:
  - Distributed Discrete Events



Example Ptolemy Model of Comp.: Continuous Time

- Prinzip:
  - Verwendung kontinuierlicher Signale (bestimmt gemäß Differential-gleichungen)
- Anwendungsgebiet:
  - Simulation
- Werkzeuge:
  - Simulink
  - Labview



Weitere Models of Computation

- Component Interaction:
  - Mischung von daten- und anfragegetriebener Ausführung
  - Beispiel: Web Server
- Discrete Time:
  - Erweiterung des synchronen Datenflussmodels um Zeit zwischen Ausführungen zur Unterstützung von Multiratensystemen
- Time-Triggered Execution
  - Die Ausführung wird zeitlich geplant
  - Anwendungsgebiet: kritische Regelungssysteme
  - Werkzeug: Giotto, FTOS
- Process Networks
  - Prozess senden zur Kommunikation Nachrichten über Kanäle
  - Kanäle können Nachrichten speichern: asynchrone Nachrichten
  - Anwendungsgebiet: verteilte Systeme
- Rendezvous
  - synchrone Kommunikation verteilter Prozesse (Prozesse warten am Kommunikationspunkt, bis Sender und Empfänger bereit sind)
  - Beispiele: CSP, CCS, Ada

Häufige Fragen zum Stoff der Vorlesung

- Was sind Aktoren?
  - Unterscheidung zum Begriff Aktoren aus der Mechatronik?
- Wie zuverlässig ist der generierte Code bei modellbasierten Entwicklungswerkzeugen?
- Wie gut ist der generierte Code zu lesen?

- Frage zu dieser Vorlesung:
  - Wann verwendet man synchrone Sprachen, wann synchronen Datenfluss?
    - Unterscheidung zwischen kontrollorientierten (Fragestellung: wie reagiert das System auf Ereignisse) und datenorientierten (Fragestellung: wie werden eingehende Daten verarbeitet) Anwendungen.

Modellierung von Echtzeitsystemen

Reaktive Systeme

Werkzeuge: SCADE, Esterel Studio

Esterel

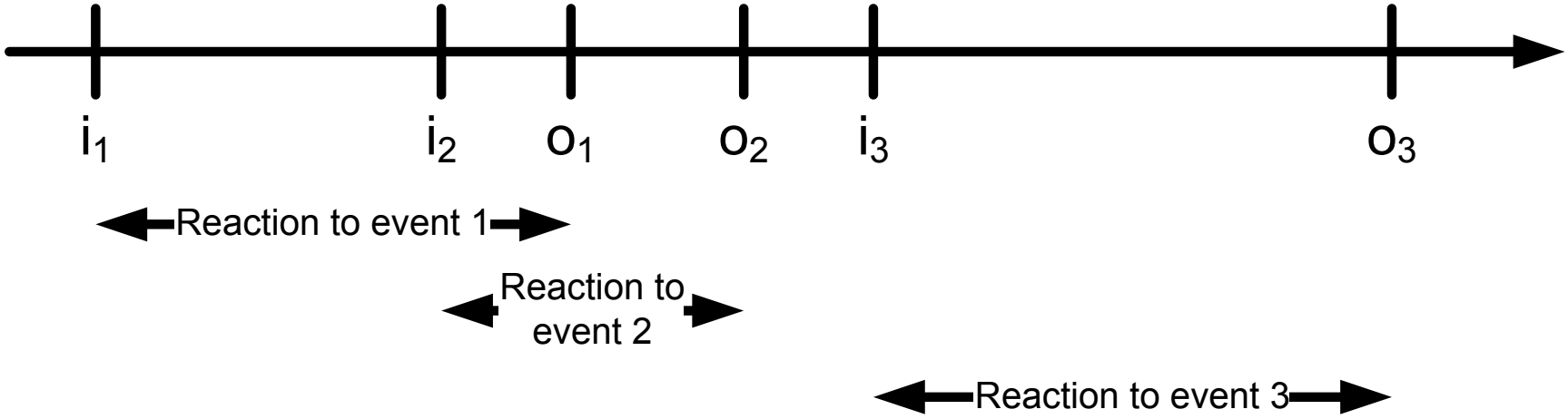
- Esterel ist im klassischen Sinne eher eine Programmiersprache, als eine Modellierungssprache
- Esterel wurde von Jean-Paul Marmorat und Jean-Paul Rigault entwickelt um die Anforderungen von Echtzeitsystemen gezielt zu unterstützen:
  - direkte Möglichkeit zum Umgang mit Zeit
  - Parallelismus direkt in der Programmiersprache
- G. Berry entwickelt die formale Semantik für Esterel
- Es existieren Codegeneratoren zur Generierung von u.a. **sequentiellen** C, C++ Code:
  - Aus Esterel-Programmen mit parallelen Berechnungen wird ein Programm mit einem Berechnungsstrang erzeugt ⇒ deterministische Ausführung
  - Technik basiert aus der Erstellung eines endlichen Automaten.
- In der Übung setzen wir die kommerziellen Werkzeuge Esterel Studio (Synfora, www.synfora.com/) / SCADE (Esterel Technology, www.esterel-technologies.com) zum Erlernen von Esterel / Lustre ein.
- SCADE wurde unter anderem zur Entwicklung des Airbus A380 eingesetzt.
- Ein Esterel-Compiler kann unter http://www-sop.inria.fr/meije/esterel/esterel-eng.html umsonst heruntergeladen werden.

Einführung in Esterel

- Esterel beschreibt reaktive Systemen, das System reagiert auf Eingabeereignisse
- Esterel gehört zu der Familie der synchronen Sprachen, weitere Vertreter: Lustre, Signal, Statecharts
- Synchrone Sprachen zeichnen sich vor allem dadurch aus, dass
  - Interaktionen (Reaktionen) des Systems mit der Umgebung die Basisschritte des Systems darstellen (**reaktives System**).
  - Anstelle von physikalischer Zeit logische Zeit (die Anzahl der Interaktionen) verwendet wird.
  - Interaktionen, oft auch **macro steps** genannt, bestehen aus Einzelschritten (micro steps).

Allgemein: Reaktive Systeme

- Bearbeitung der Ereignisse kann sich überlappen (i input, o output)



Synchrony hypothesis

- Die Synchronitätshypothese (synchrony hypothesis) nimmt an, dass die zugrunde liegende physikalische Maschine des Systems unendlich schnell ist.

  → Die Reaktion des Systems auf ein Eingabeereignis erfolgt augenblicklich. Reaktionsintervalle reduzieren sich zu Reaktionsmomenten (reaction instants).
- **Rechtfertigung**: Diese Annahme ist korrekt, wenn die Wahrscheinlichkeit des Eintreffens eines zweiten Ereignisses, während der initialen Reaktion auf das vorangegangene Ereignis, sehr klein ist.
- Esterel erlaubt das gleichzeitige Auftreten von mehreren Eingabeereignissen. Die Reaktion ist in Esterel dann vollständig, wenn das System auf alle Ereignisse reagiert hat.

Determinismus

- Esterel ist deterministisch: auf eine Sequenz von Ereignissen (auch gleichzeitigen) muss immer dieselbe Sequenz von Ausgabeereignissen folgen.
- Alle Esterel-Anweisungen und -Konstrukte sind garantiert deterministisch. Die Forderung nach Determinismus wird durch den Esterel Compiler überprüft.
- Durch den Determinismus wird die Verifikation von Anwendungen wesentlich vereinfacht, allerdings birgt er auch die Gefahr, dass Ereignisse „vergessen“ werden, falls sie exakt zeitgleich mit höher priorisierten Ereignissen eintreffen.

Esterel an einem Beispiel: Aufzugstür

- Aufgabe: Öffnen und Schließen der Aufzugstür
- Sicherheitsfunktion:
  - Tür darf während der Fahrt nicht geöffnet werden
- 1. Schritt: Definition der Module (parallelen Abläufe)
- 2. Schritt: Definition der Eingangssignale
- 3. Schritt: Definition der Ausgangssignale
- 4. Schritt: Definition der Zustände (inkl. Anfangszustand):
- 5. Schritt: Definition der Zustandsübergänge