Embedded Systems / Eingebettete Systeme

Studiengang Informatik
Campus Minden

Matthias König

FH Bielefeld
University of
Applied Sciences

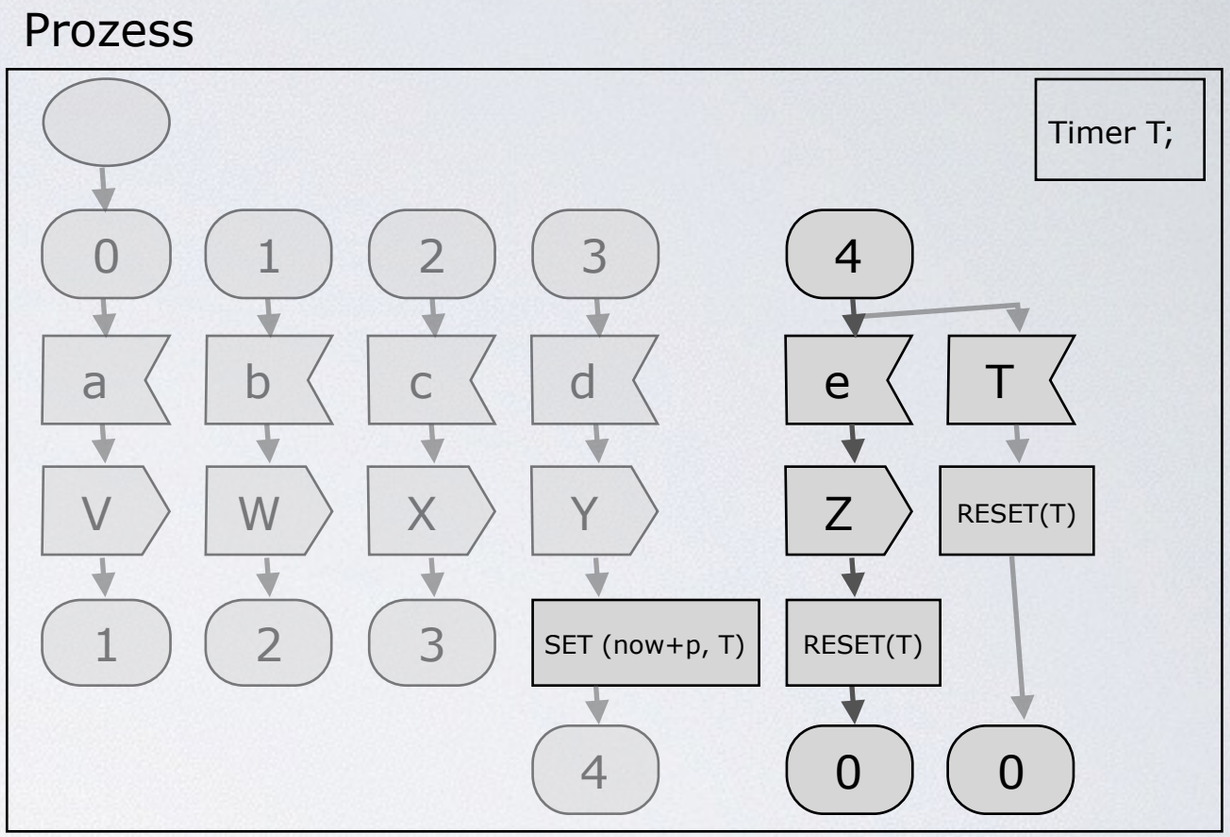
Beispiel einer Anwendung: Toaster

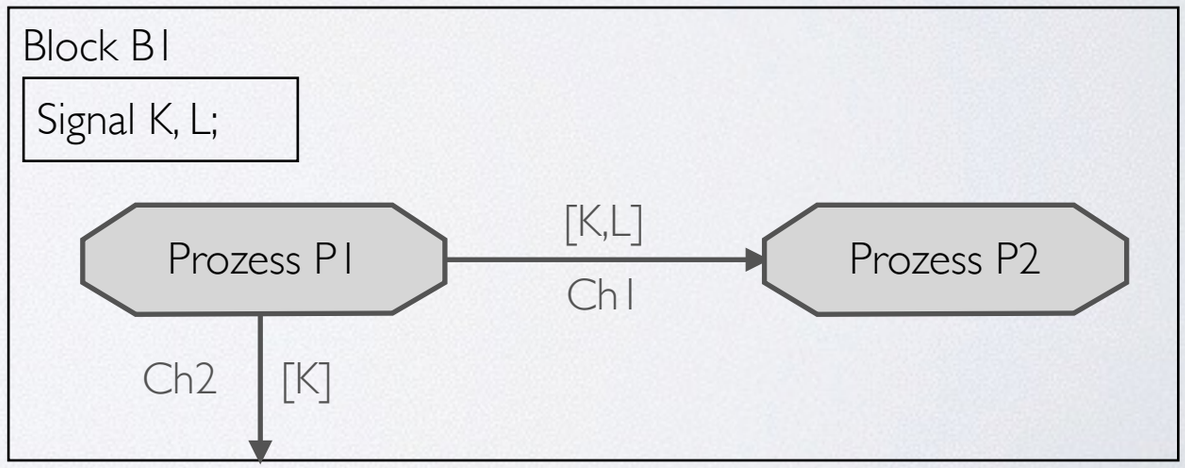
- Sensor für Einstellungen und Start/Stop
- Steuerung des Röstens mit Timer
- Üblich ist Temperatursensor und An-/Aus-Steuerung der Heizvorrichtung (Oszillieren)

Beispiel eines Toasters mit Digitalanzeige

Wiederholung: SDL

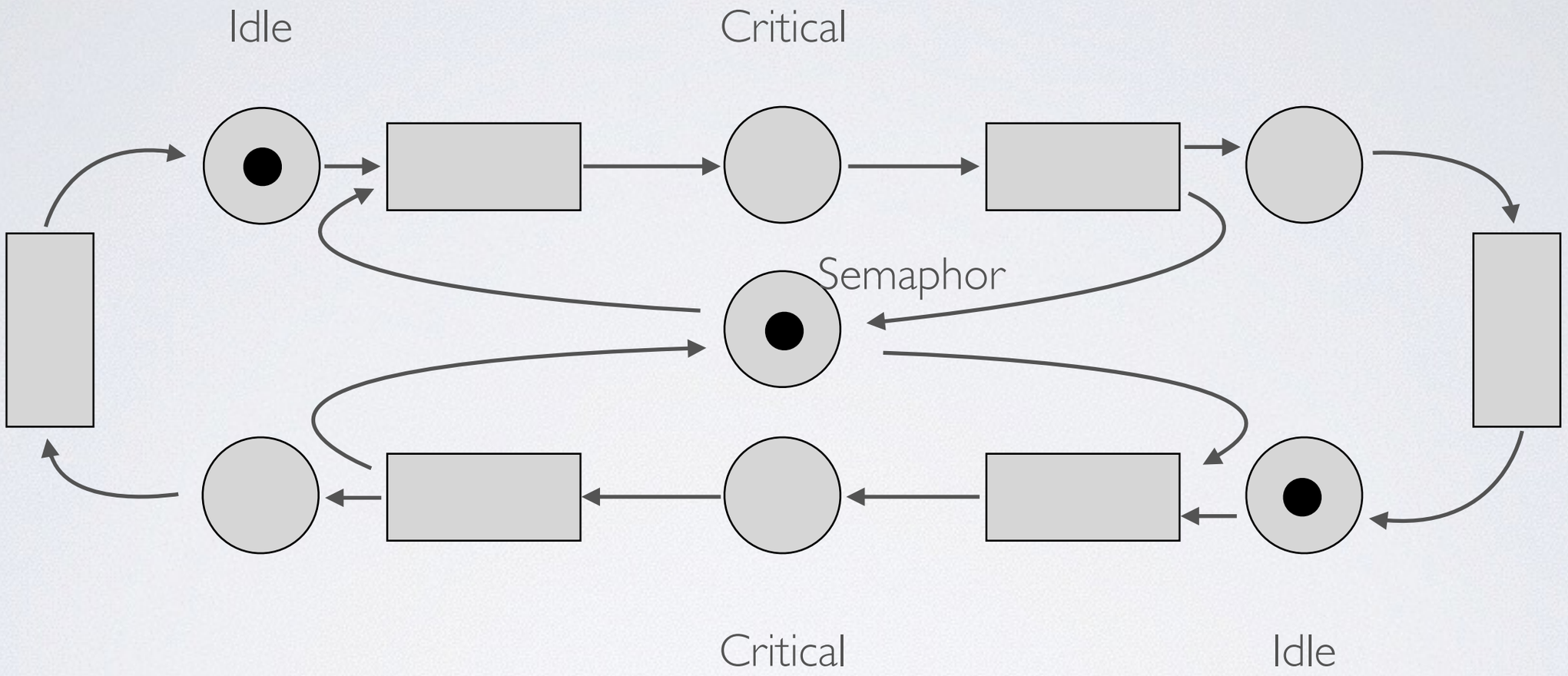
- Basiert auf
  - Zustandsautomaten
  - Asynchronen Nachrichtenaustausch
- Für verteilte Systeme ausgelegt
- Unterstützt graphische und textuelle Repräsentation





[Quelle: Marwedel, Eingebettete Systeme]

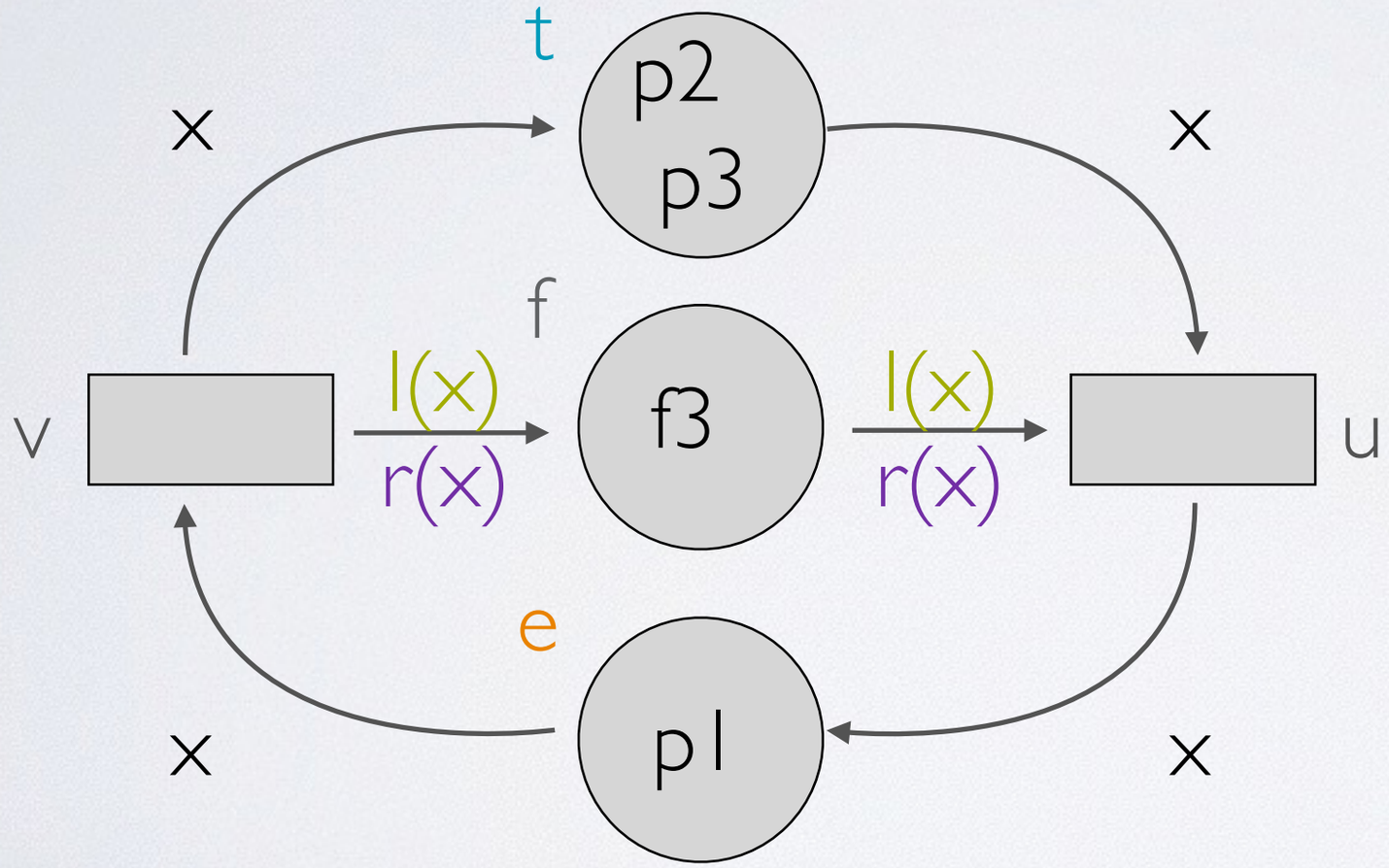
Petrinetz-Beispiel: Semaphor



[Quelle: Marwedel, Eingebettete Systeme]

Philosophenproblem: Prädikat-/Ereignisnetz

- Variablen für Philosophen (p1, p2, p3) und Gabeln (f1, f2, f3)



x: Philosoph
l: linke Gabel
r: rechte Gabel
t: denkt
e: isst
f: Gabel ist frei

[Quelle: Marwedel, Eingebettete Systeme]

UML/SysML

Unified Modeling Language UML

- Graphische Modellierungssprache
- Ausgelegt für Spezifikation von Software
- Viele UML-Diagramme an anderen graphischen Modellierungssprachen angelehnt

UML: Diagrammarten

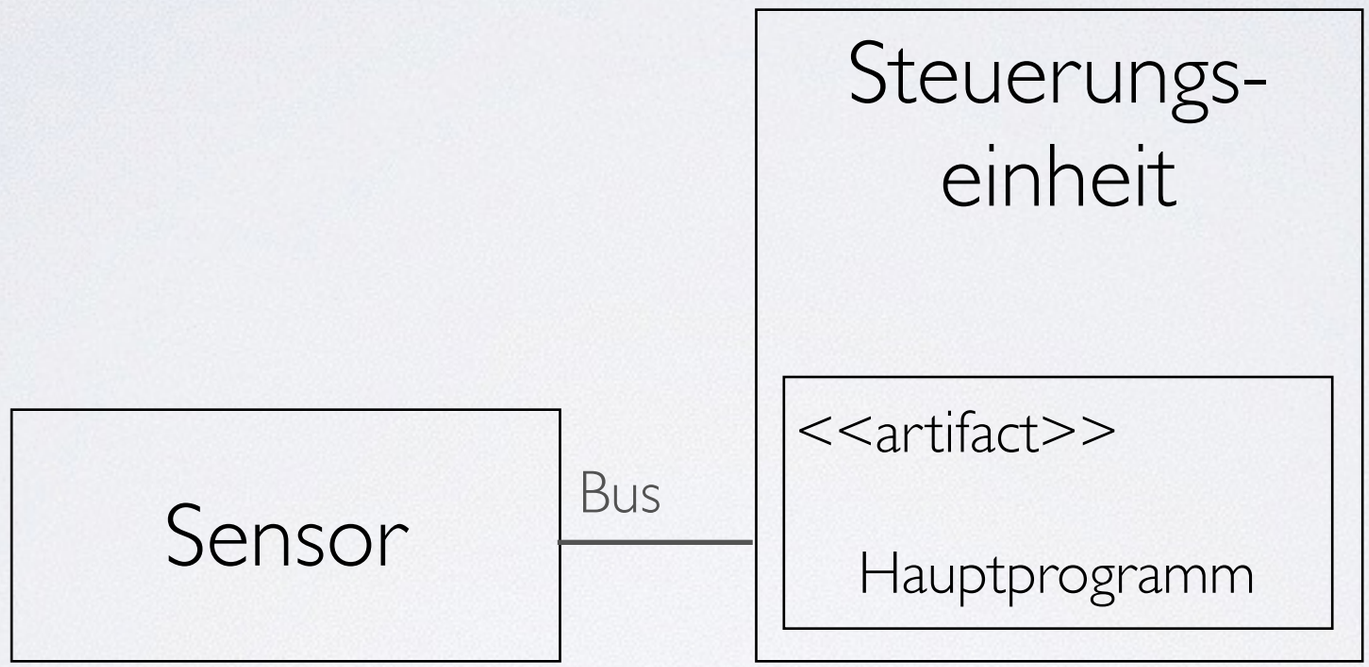
- Use-Case-Diagramm
- Komponentendiagramm
- Paketdiagramm
- Klassendiagramm und Kommunikationsdiagramm

Anwendung, Aufteilung, Klassenbeziehungen

- Automatendiagramm (ähnlich zu StateChart)
- Aktivitätsdiagramm (ähnlich zu Petrinetz)
- Verteilungsdiagramm
- Zeitverlaufsdiagramm und Sequenzdiagramm...

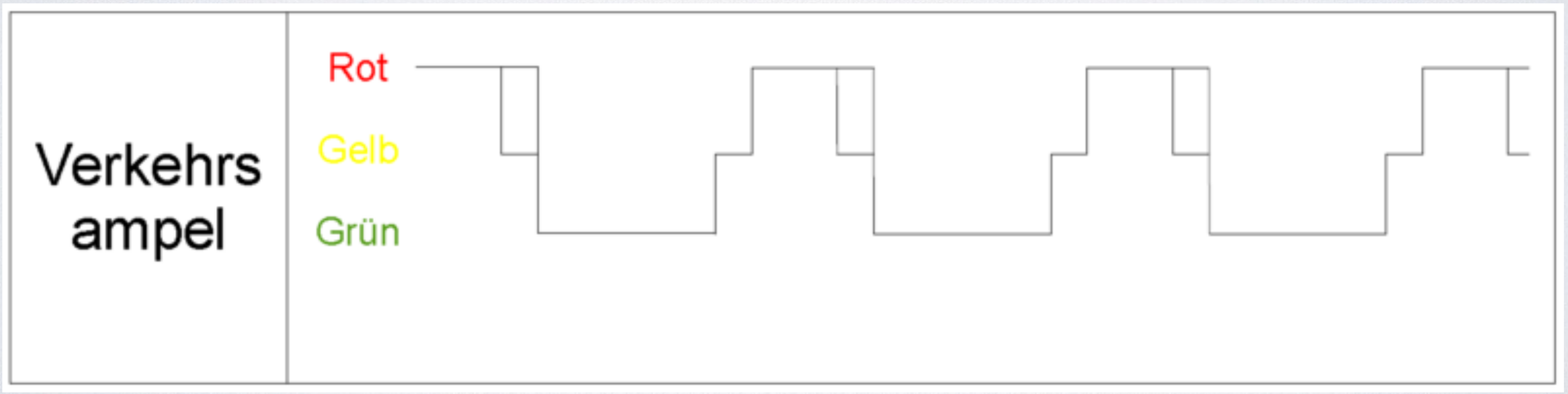
UML: Verteilungsdiagramm

- Aufteilung von Software (Artefakten) auf Hardware (Knoten)



UML: Zeitverlaufsdiagramm

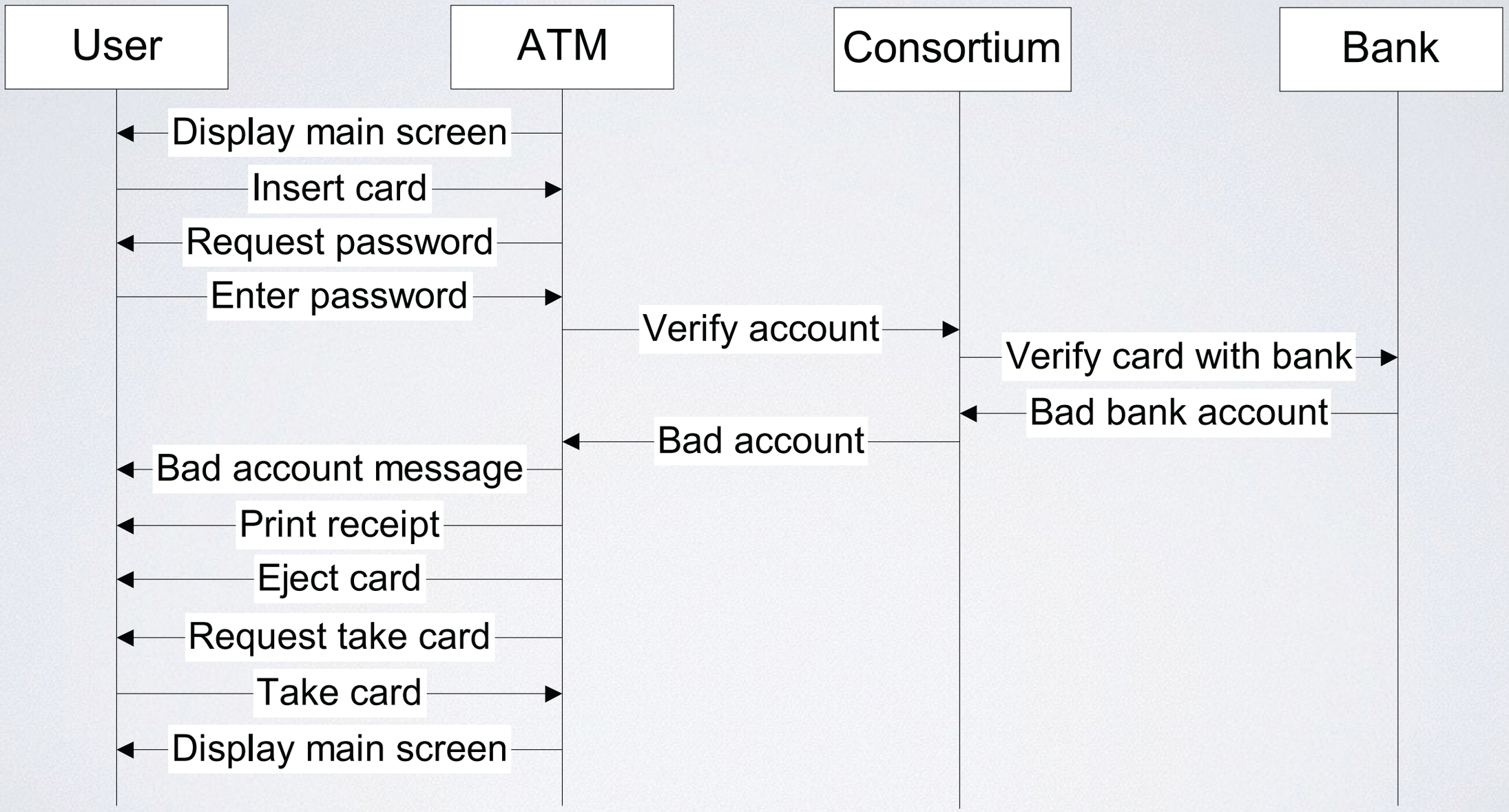
- Zweidimensionale Darstellung von Objektzuständen
- Zustand gegenüber Zeit
- Einsatzbar für Echtzeitsysteme



[Quelle: http://upload.wikimedia.org/wikipedia/de/5/55/Zeitverlaufsdiagramm.png

UML: Sequenzdiagramm

- Darstellung von Ablaufplänen

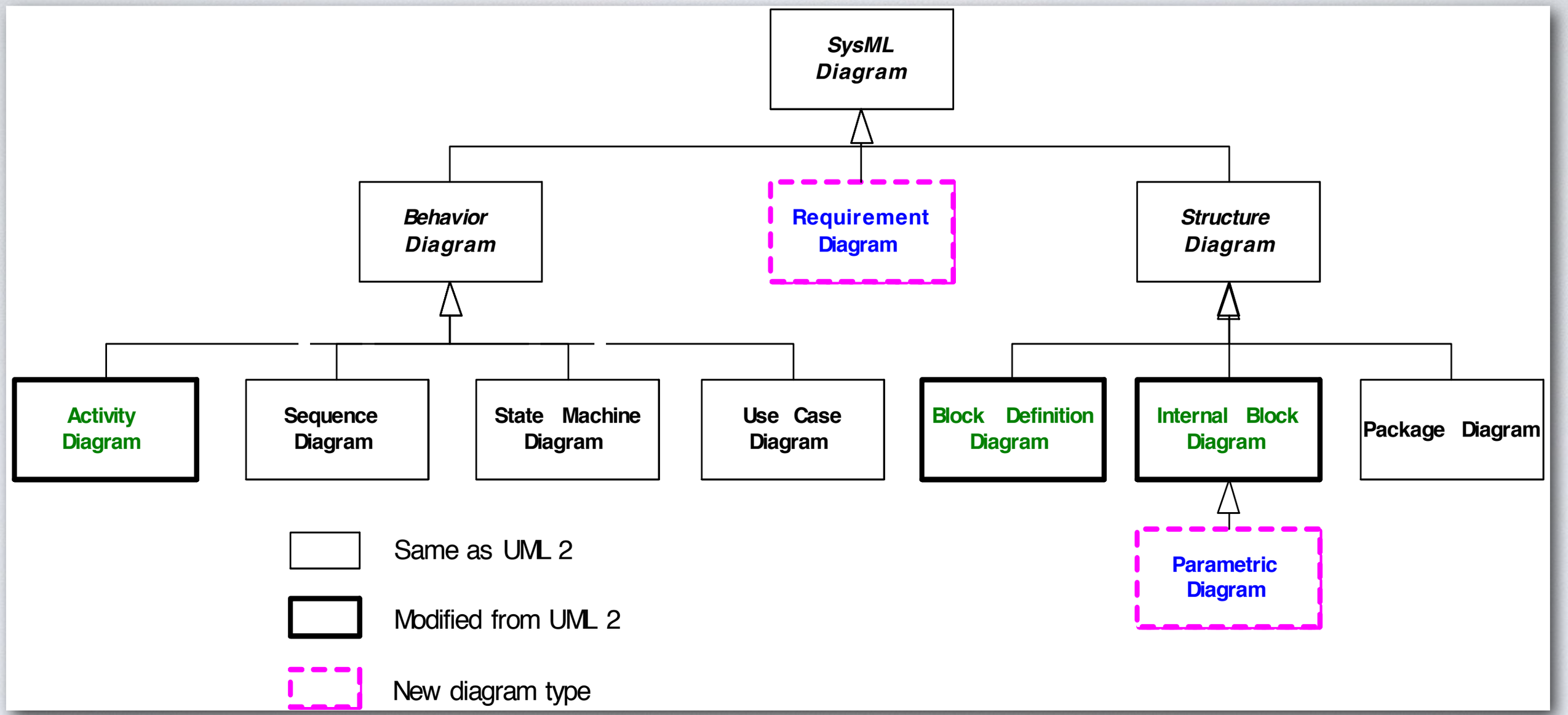


[Quelle: Latronico, Koopman, 2001]

System Modeling Language SysML

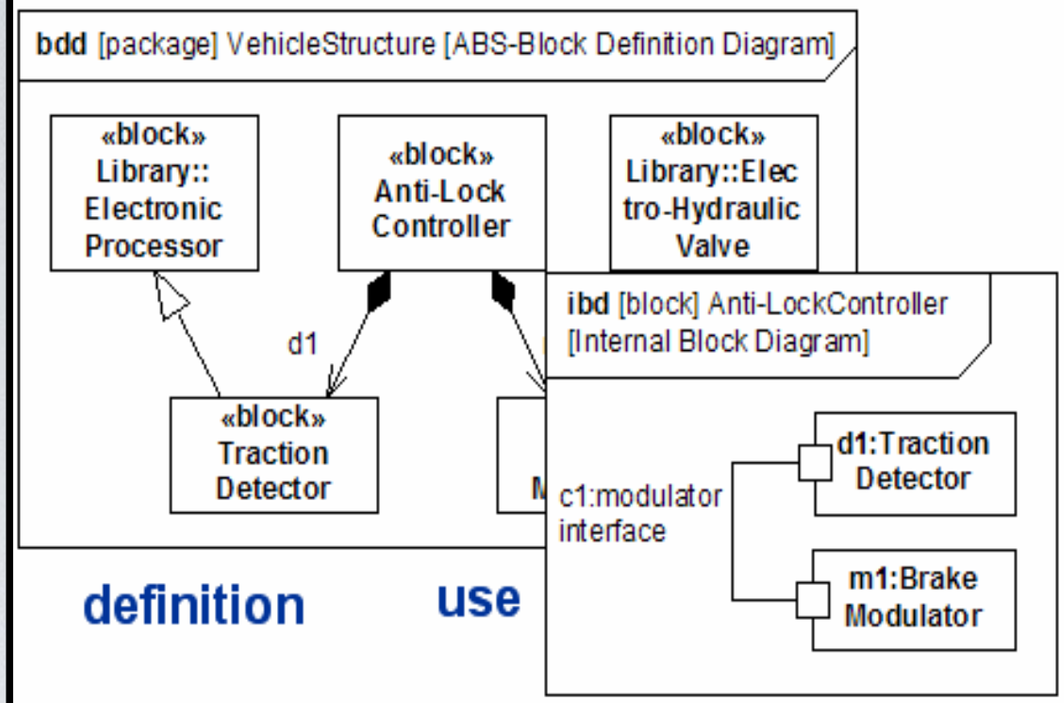
- Basiert auf UML
- *von Software zum System*
- Modellierung von komplexen Systemen
- Unterstützt Design, Analyse und Test
- Umfasst insbesondere für Embedded Systems
  - Erweitertes Aktivitätsdiagramm
  - Erweitertes Automatendiagramm
  - Parameterdiagramm und Anforderungsdiagramm

SysML-Taxonomie

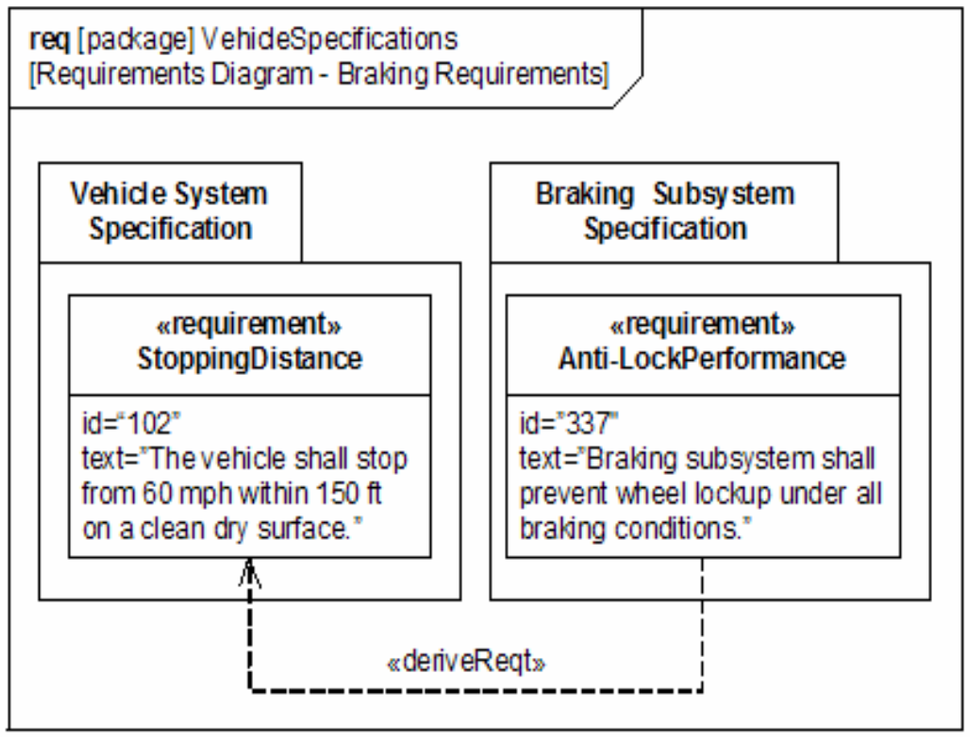


Quelle: OMG Systems Modeling Language (OMG SysML), S.156 , http://www.omg.org/spec/SysML/1.2/

1. Structure

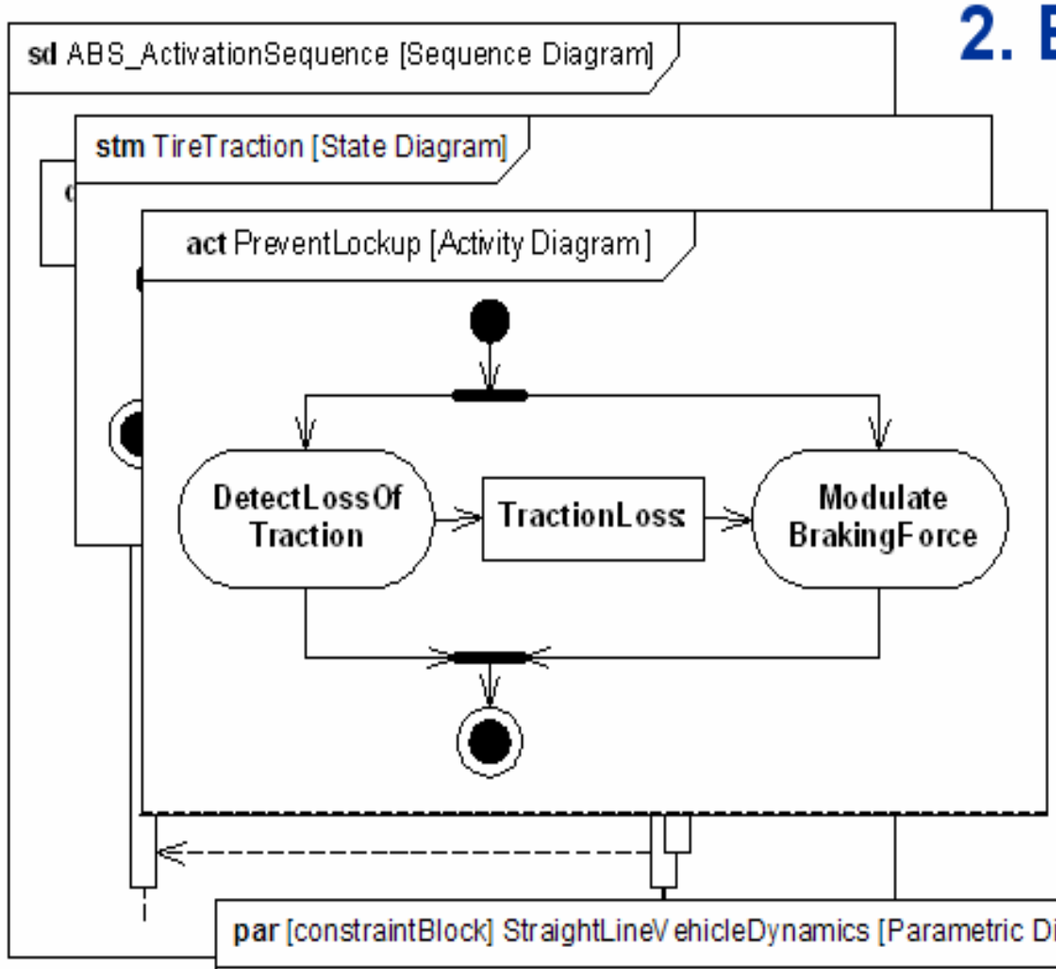


definition **use**



3. Requirements

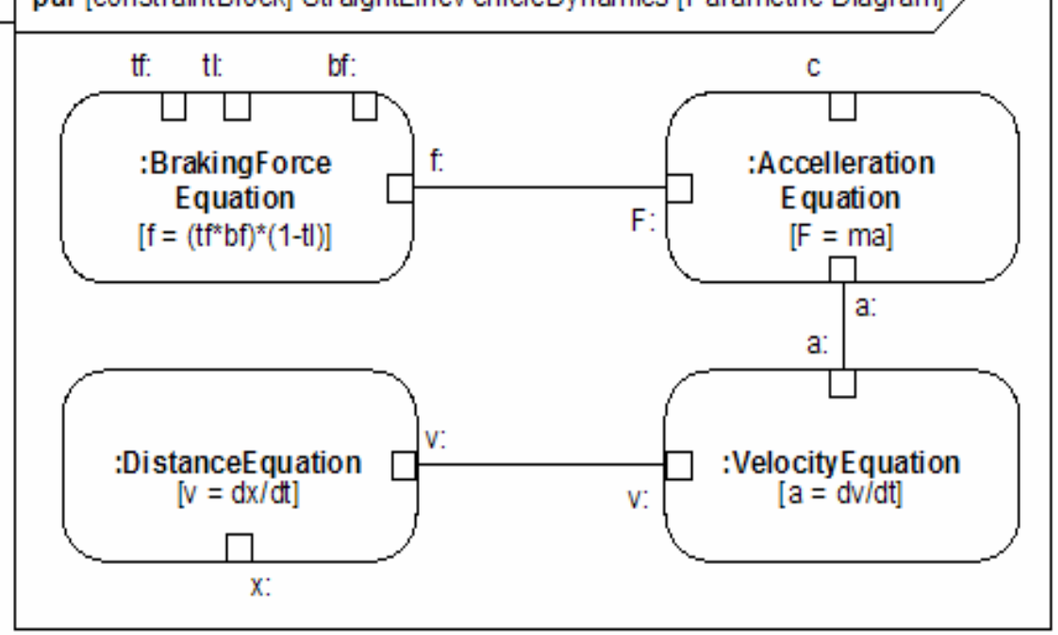
2. Behavior



interaction

state machine

activity/ function

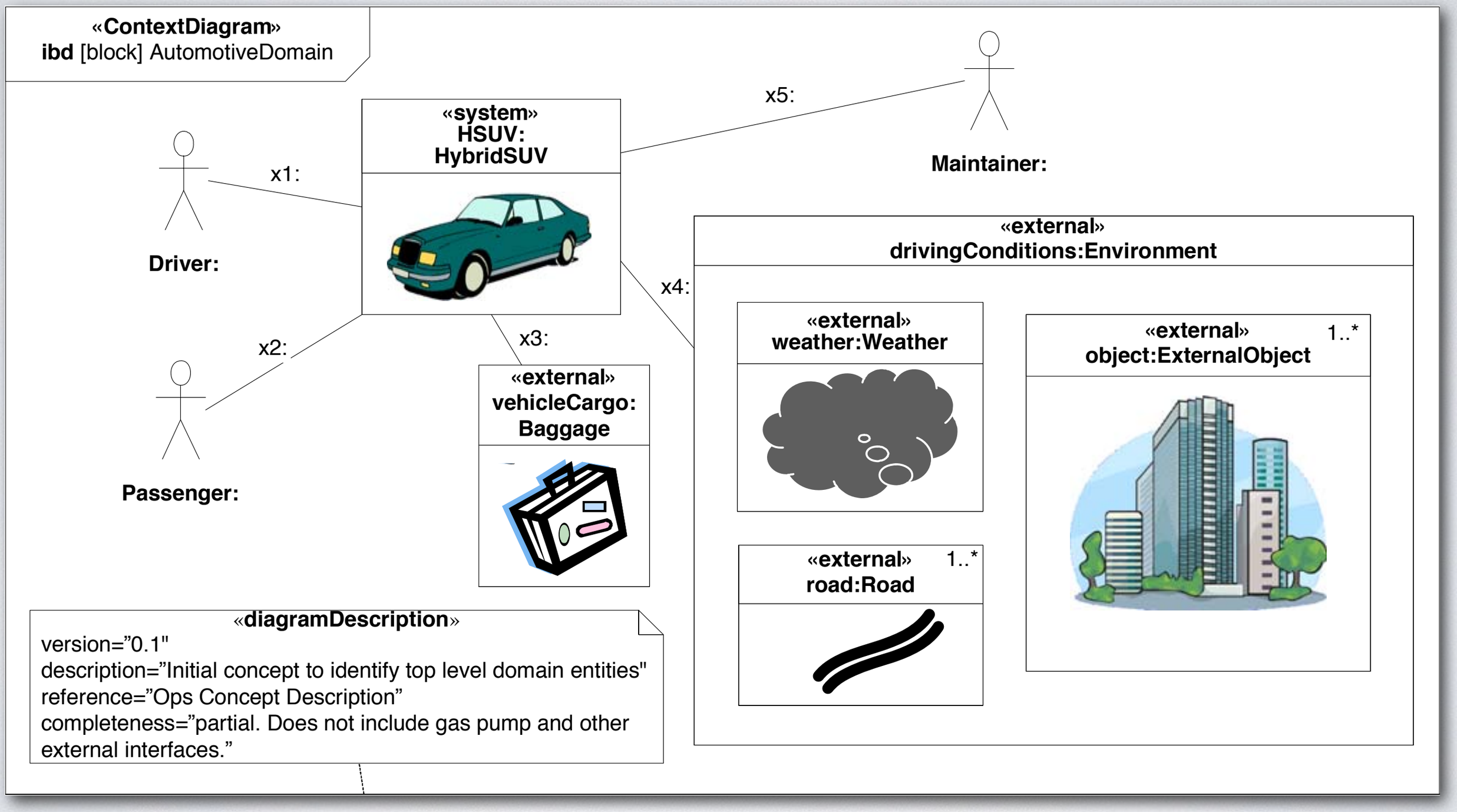


4. Parametrics

Basisbestandteile von SysML: Beispiel Auschnitt ABS

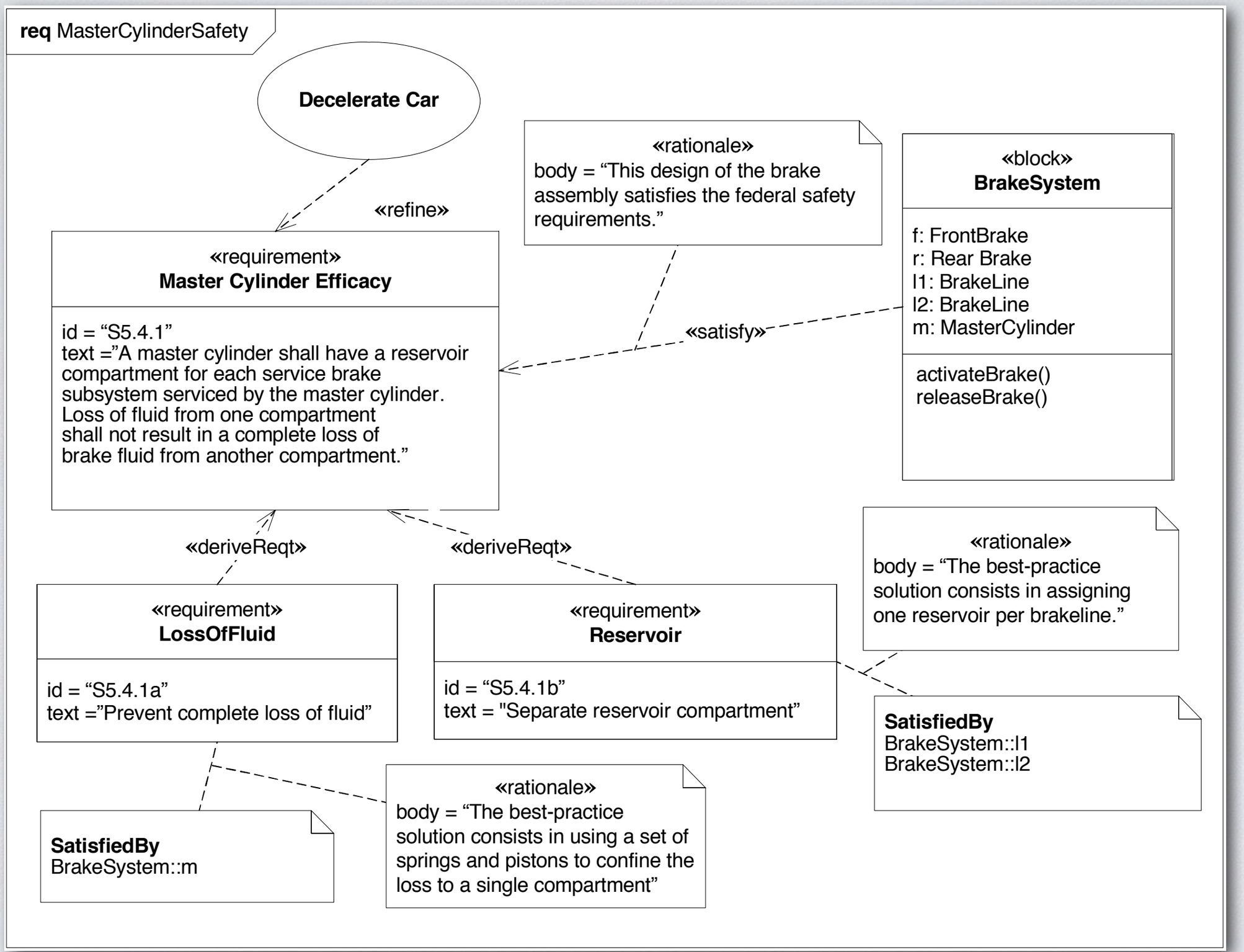
[Quelle: Hause, 2006]

Nutzerdefiniertes Kontextdiagramm (SysML)



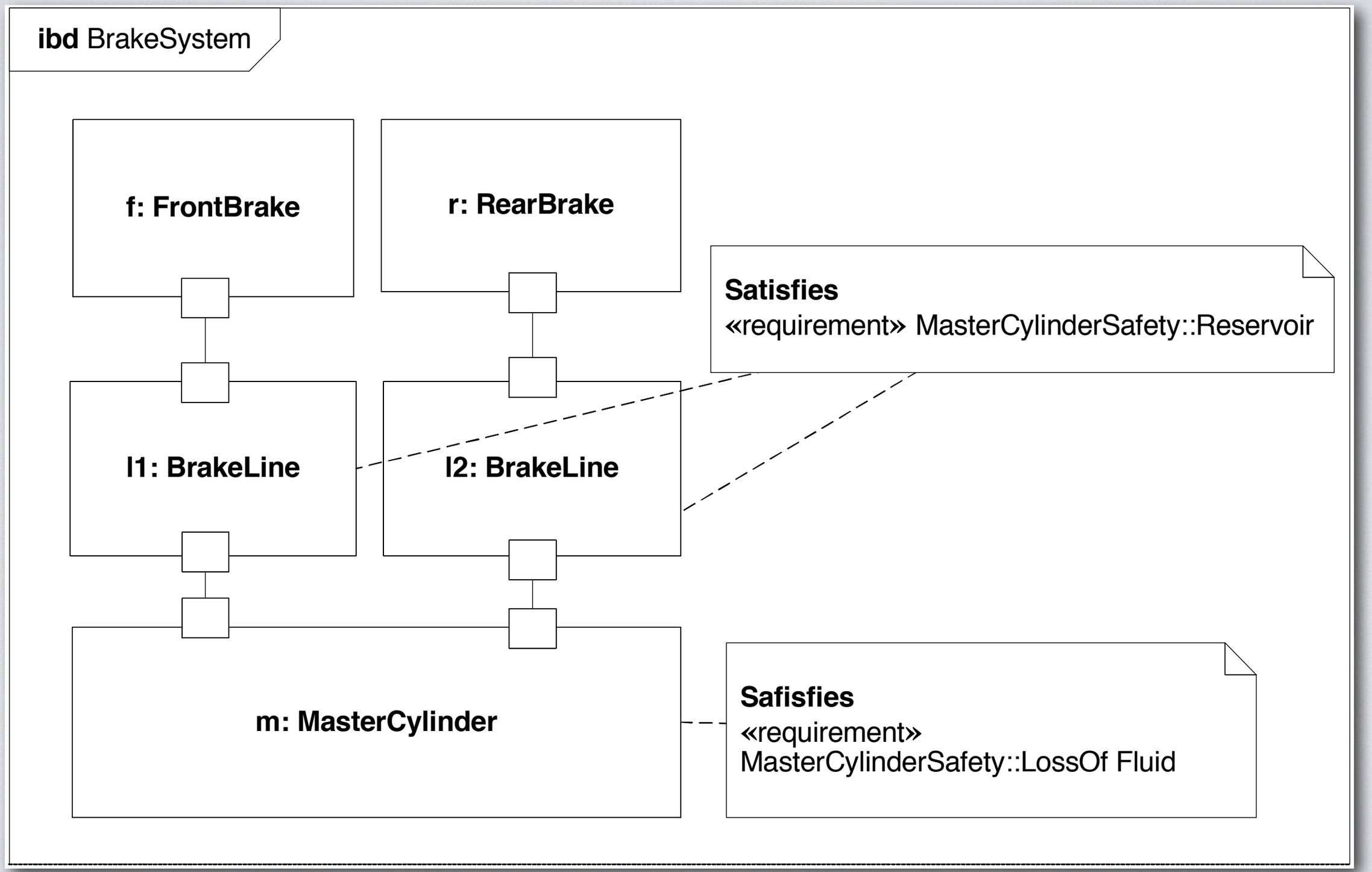
Quelle: OMG Systems Modeling Language (OMG SysML), S.164 , http://www.omg.org/spec/SysML/1.2/

Requirements-Diagramm



Quelle: OMG Systems Modeling Language (OMG SysML), S.151 , http://www.omg.org/spec/SysML/1.3/

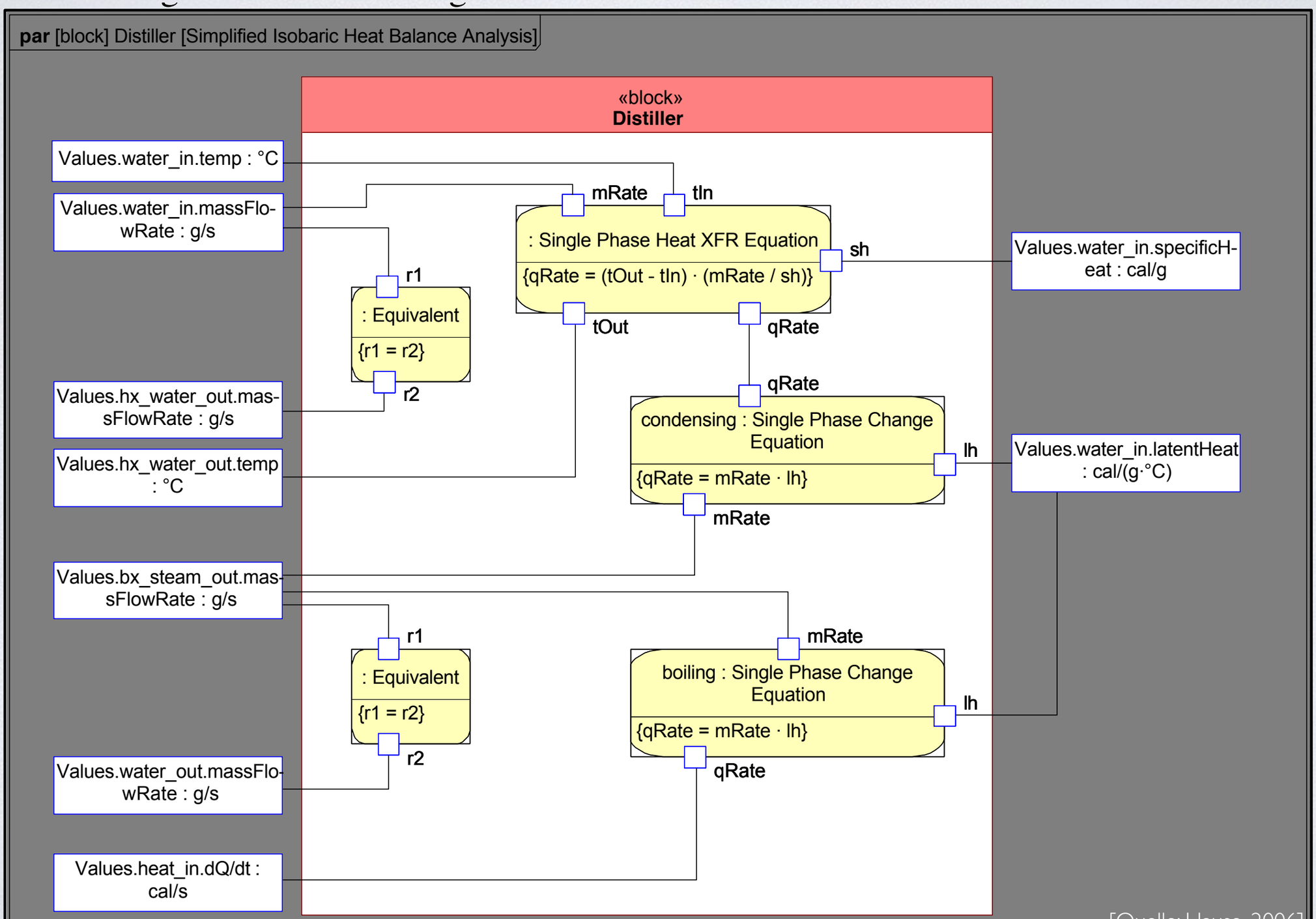
Internes Blockdiagramm (SysML)



Quelle: OMG Systems Modeling Language (OMG SysML), S.152 , http://www.omg.org/spec/SysML/1.3/

SysML: Parameterdiagramm

- definiert quantitative Bedingungen / mathematische Formeln.



[Quelle: Hause, 2006]

Modellierung und Nachrichtenaustausch

- Bisher:
  - Broadcasting (Nachricht an alle),
    z.B. Statecharts
  - Asynchroner Nachrichtenaustausch, (ohne Warten auf Empfang), z.B. SDL
- Noch nicht:
  - Synchroner Nachrichtenaustausch (Sender und Empfänger werden blockiert/synchronisiert),
    z.B. Programmiersprache ADA

APA

Programmiersprache ADA

- entwickelt im Auftrag des US Verteidigungsministerium mit Ziel der Verbesserung der Softwareentwicklung,
- ISO/ANSI standardisiert,
- für Echtzeit und eingebettete Systeme entworfen,
- unterstützt u.a.
  - Nebenläufigkeit,
  - Ausnahmebehandlung,
  - Laufzeittest.
- Cross-Compiling entsprechend z.B. Toolchains für C

ADA: Hello World

```
with Ada.Text_IO; use Ada.Text_IO;
  procedure Hello is
begin
   Put_Line("Hello, world!");
end Hello;
```

[Quelle: Wikipedia: Ada_(programming_language)

ADA & Arduino

```
with AVR.MCU;
with AVR.Wait;
use AVR;

 procedure blink is

     procedure Delay_MS(MS : Natural) is
     begin
         for X in 1..MS loop
             AVR.Wait.Wait_4_Cycles(8000);
         end loop;
     end;

     LED : Boolean renames MCU.PortB_Bits(5);

begin
     MCU.DDRB_Bits := (others => DD_Output);

     loop
         LED := True;
         Delay_MS(1000);
         LED := False;
         Delay_MS(1000);
     end loop;

end blink;
```

[Quelle: Sourceforge, AVR-Ada]

ADA: Prozesse (Tasks)

```
procedure example is
    -- Zwei Prozesse
    task a;
    task b;

    task body a is
    -- lokale Deklarationen für a...
    begin
        -- Anweisungen für a...
    end a;

    task body b is
    -- lokale Deklarationen für b...
    begin
        -- Anweisungen für b...
    end b;

begin
    -- tasks a, b starten vor Anweisungen von example
end example;
```

[Quelle: Marwedel, Eingebettete Systeme]

ADA: Nachrichtenaustausch / Rendez-Vous

```
with Ada.Text_IO; use Ada.Text_IO;
procedure example is
    task a is
      entry call(x:integer); -- Erlaubt Aufruf von call(x) von anderen Tasks.
    end a;
    task b;

    task body a is
    begin
      Put_Line("task a: waiting");
      delay 3.0;
      Put_Line("task a: accepting");
      accept call(x:integer);  -- Erst hier Annahme von call(x) erlaubt.
    end a;

    task body b is
    begin
      Put_Line("task b: call a");
      a.call(1); -- Aufruf call von Task a, warten bis Aufruf akzeptiert wird.
      Put_Line("task b: ready");
    end b;

begin
    Put_Line("running");
end example;
```

ADA: Nachrichtenaustausch (timed entry calls)

```
with Ada.Text_IO; use Ada.Text_IO;
procedure sel_example is
    task a is
      entry call(x:integer); -- Erlaubt Aufruf von call(x) von anderen Tasks.
    end a;
    task b;

    task body a is
    begin
      Put_Line("task a: not accepting");
      delay 3.0;
      Put_Line("task a: accepting");
      accept call(x:integer);  -- Erst hier Annahme von call(x) erlaubt.
    end a;

    task body b is
    begin
      Put_Line("task b: call a");
      for i in 1..3 loop select -- 3 Durchläufe a 1s da Task a 3s wartet.
         a.call(1); -- Aufruf call von Task a, warten bis Aufruf akzeptiert wird.
         Put_Line("task b: ready");
      or
         delay 1.0; -- Wenn call von a nicht innerhalb 1s möglich, nicht warten...
         Put_Line("task b: wait for a");
      end select; end loop;
    end b;
...
```

ADA: Beispiel aus Wikipedia

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Traffic is

   type Airplane_ID is range 1..10;          -- 10 airplanes (= tasks)

   task type Airplane(ID: Airplane_ID);      -- task type representing airplanes
   type Airplane_Access is access Airplane; -- access type (reference) to Airplane

   protected type Runway is                  -- a protected object - the shared runway
      entry Assign_Aircraft(ID: Airplane_ID);
      entry Cleared_Runway (ID : Airplane_ID);
      entry Wait_For_Clear;
   private
      Clear: Boolean := True; -- protected private data - generally more than just a
flag...
   end Runway;
   type Runway_Access is access all Runway;

   -- the air traffic controller takes requests for takeoff and landing
   task type Controller(My_Runway: Runway_Access) is
      entry Request_Takeoff (ID: in Airplane_ID; Takeoff: out Runway_Access);
      entry Request_Approach(ID: in Airplane_ID; Approach: out Runway_Access);
   end Controller;

   Runway1    : aliased Runway;              -- instantiate a runway
   Controller1: Controller(Runway1'Access); -- and a controller to manage it
```

[Quelle: Wikipedia: Ada_(programming_language)]

```
------ the implementations of the above types ------
 protected body Runway is
    entry Assign_Aircraft (ID : Airplane_ID)
      when Clear is   -- the entry guard - tasks are blocked until this is true
    begin
      Clear := False;     Put_Line (Airplane_ID'Image (ID) & " on runway ");
    end;

    entry Cleared_Runway (ID : Airplane_ID) when not Clear is
    begin
       Clear := True;     Put_Line (Airplane_ID'Image (ID) & " cleared runway ");
    end;

    entry Wait_For_Clear when Clear is begin
       null;
    end;
 end Runway;

 task body Controller is
 begin
    loop
       My_Runway.Wait_For_Clear;    -- wait until runway is available
       select                       -- wait for two types of requests
          when Request_Approach'count = 0 =>  -- landings have priority
           accept Request_Takeoff (ID : in Airplane_ID; Takeoff : out Runway_Access) do
             My_Runway.Assign_Aircraft (ID);  -- reserve runway
             Takeoff := My_Runway;            -- tell airplane which runway
          end Request_Takeoff;                -- end of the synchronised part
       or
          accept Request_Approach (ID : in Airplane_ID; Approach : out Runway_Access) do
             My_Runway.Assign_Aircraft (ID);
             Approach := My_Runway;
          end Request_Approach;
       or                           -- terminate if nobody left who could call
          terminate;
       end select;
    end loop;
 end;
```

[Quelle: Wikipedia: Ada_(programming_language)]

```
task body Airplane is
      Rwy : Runway_Access;
   begin
      Controller1.Request_Takeoff (ID, Rwy); -- wait to be cleared for takeoff
      Put_Line (Airplane_ID'Image (ID) & "  taking off...");  delay 2.0;
      Rwy.Cleared_Runway (ID);
      delay 5.0; -- fly around a bit...
      loop
         select   -- try to request a runway
            Controller1.Request_Approach (ID, Rwy); -- this is a blocking call
            exit; -- if call returned we're clear for landing - proceed...

         or delay 3.0;  -- timeout - if no answer in 3 seconds, do something else
            Put_Line (Airplane_ID'Image (ID) & "   in holding pattern");
         end select;
      end loop;
      delay 4.0;  -- do landing approach...
      Put_Line (Airplane_ID'Image (ID) & "              touched down!");
      Rwy.Cleared_Runway(ID);  -- notify runway that we're done here.
   end;

   New_Airplane: Airplane_Access;
begin
   for I in Airplane_ID'Range loop  -- create a few airplane tasks
      New_Airplane := new Airplane(I); delay 3.0;
   end loop;
end Traffic;
```

[Quelle: Wikipedia: Ada_(programming_language)

Hardware-Beschreibungssprachen

Hardware-Beschreibungssprachen

- Für Design und Test integrierter Schaltungen
- Zeitliches Verhalten und Nebenläufigkeit berücksichtigt
- Bekannte Sprachen:
  - Verilog
  - Very High Speed Integrated Circuit Hardware Description Language VHDL
  - SystemC
- Syntheseefähiger Code kann in Schaltung überführt werden

VHDL - Beispiel


```
entity full_adder is
port(a, b, carry_in: in Bit; -- input ports
     sum,carry_out: out Bit); -- output ports
end full_adder;

architecture behavior of full_adder is
begin
    sum <= (a xor b) xor carry_in after 10 ns;
    carry_out <= (a and b) or (a and carry_in) or
                 (b and carry_in) after 10 ns;
end behavior;
```

[Quelle: Marwedel, Eingebettete Systeme]

SystemC - Beispiel

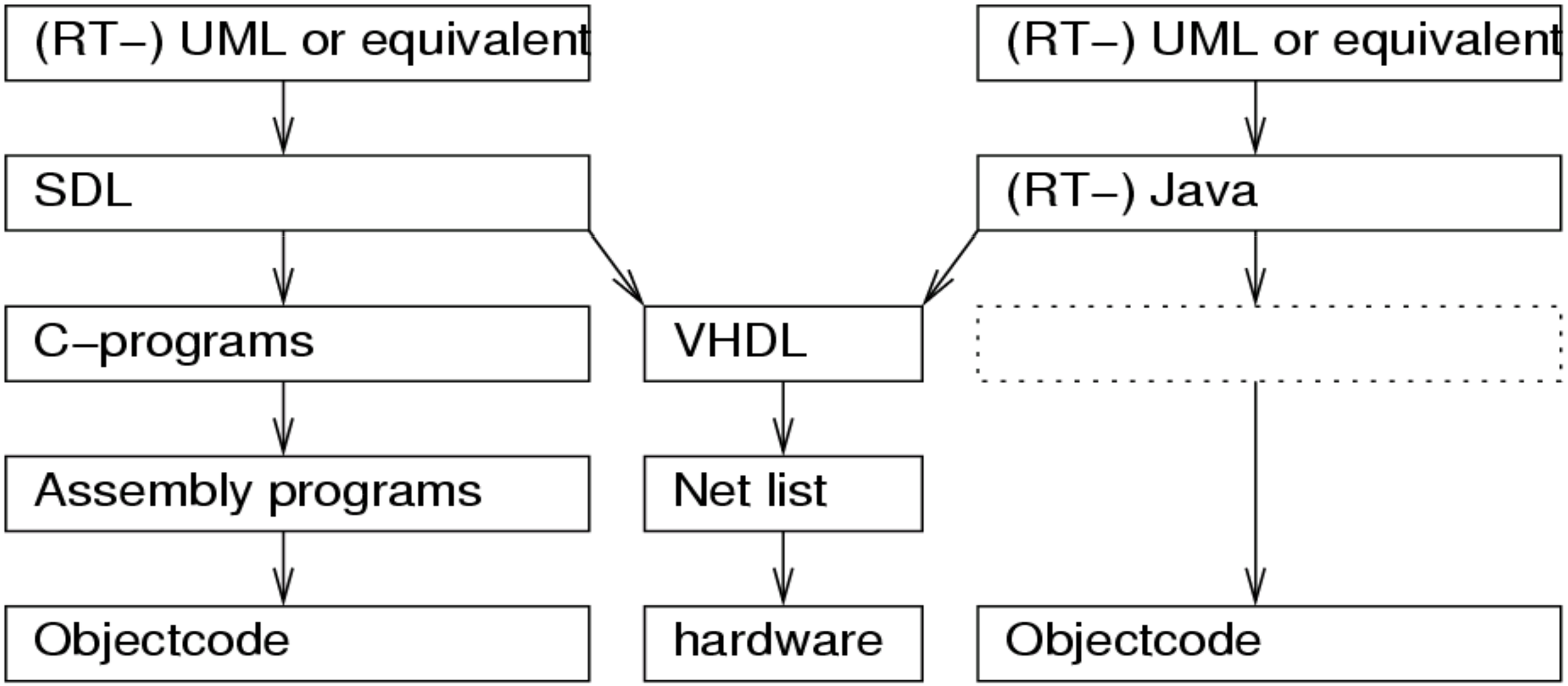
```
#include "systemc.h"

SC_MODULE(adder) // Moduldeklaration (eine Art Klasse)
{
  sc_in<int> a, b;          // Zwei Eingangs-Ports (a und b)
  sc_out<int> sum;          // Ein Ausgangs-Port

  SC_CTOR(adder)
  {
    SC_THREAD(doit);
    sensitive <<a <<b;
  }
    void doit()
    {
      while(true)
      {
        sum.write(a.read() + b.read());
        wait();
      }
    }
};
```

[Quelle: Wikipedia, SystemC]

Realisierungsszenario mit mehreren Sprachen



[Quelle: Marwedel, Eingebettete Systeme]

Modellbasierte Softwareentwicklung eingebetteter Systeme

Einführung für Vortrag nächste Woche

Modellbasierte Softwareentwicklung

- Modelle als
  - höheren Abstraktionsgrad
  - plattformunabhängige Beschreibung
  - Ausgangsbasis zur Generierung von Quellcode
  - zur Simulation, Verifikation und Validierung

versprechen (≠ immer erreichen) mehr Produktivität

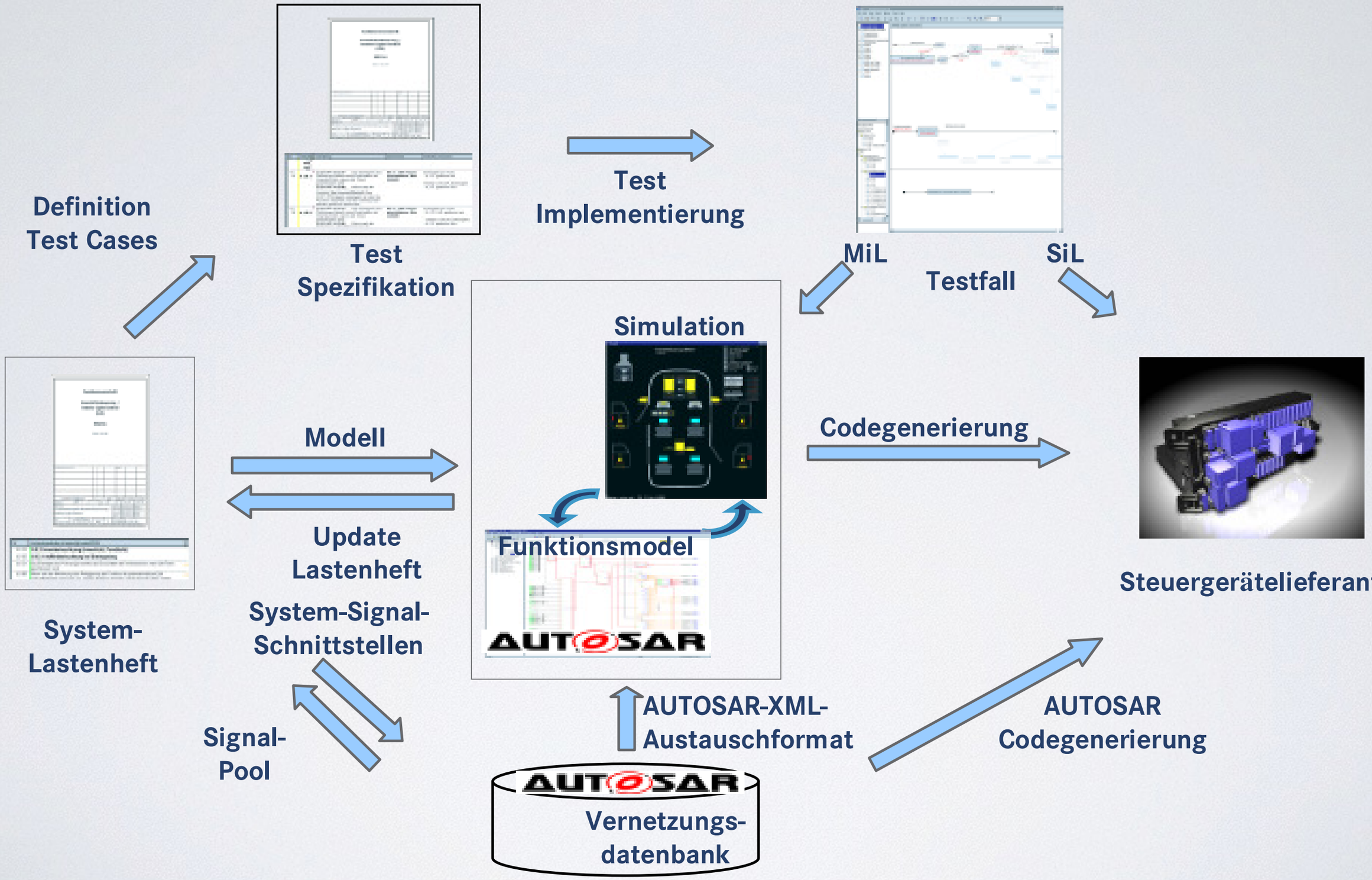
Eingebettete Systeme und Modelle

- Eingebettete Systeme: **Interaktion** mit der physikalischen Umwelt mittels Sensoren und Aktoren

- Modelle zur Beschreibungen dieser Interaktion
  - Zustandsmaschinen
  - Mathematische Modelle (z.B. Differentialgleichungen)

- (vgl. auch frühere Vorlesung)

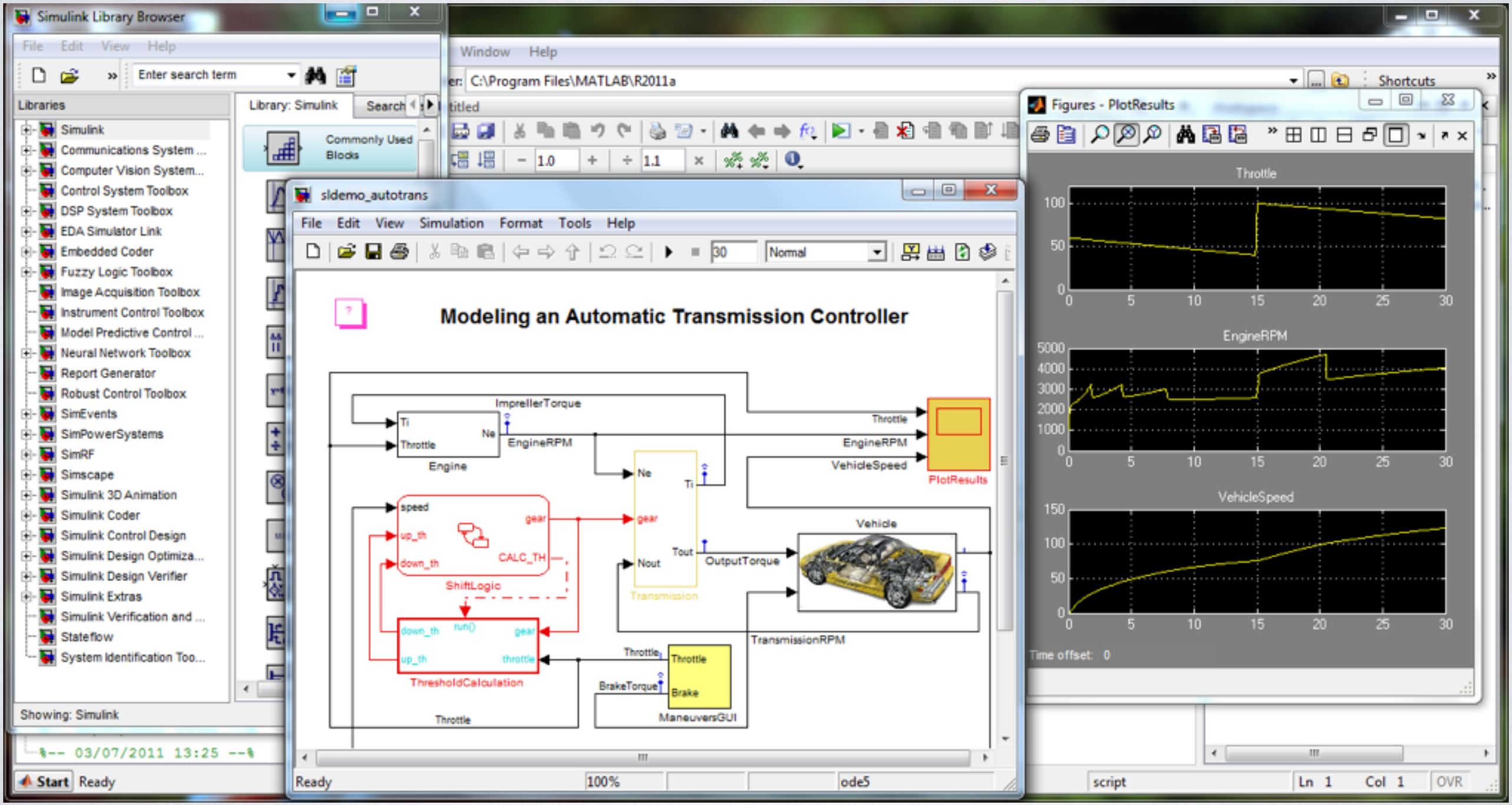
Zusammenhänge bei der modellbasierten Entwicklung am Beispiel unter AUTOSAR



[Quelle: Dziobek et al, 2012, Herausforderungen bei der modellbasierten Entwicklung…]

Vielzahl weiterer Spezifikationssprachen

- Beispiel: Matlab/Simulink für modellbasierte Entwicklung

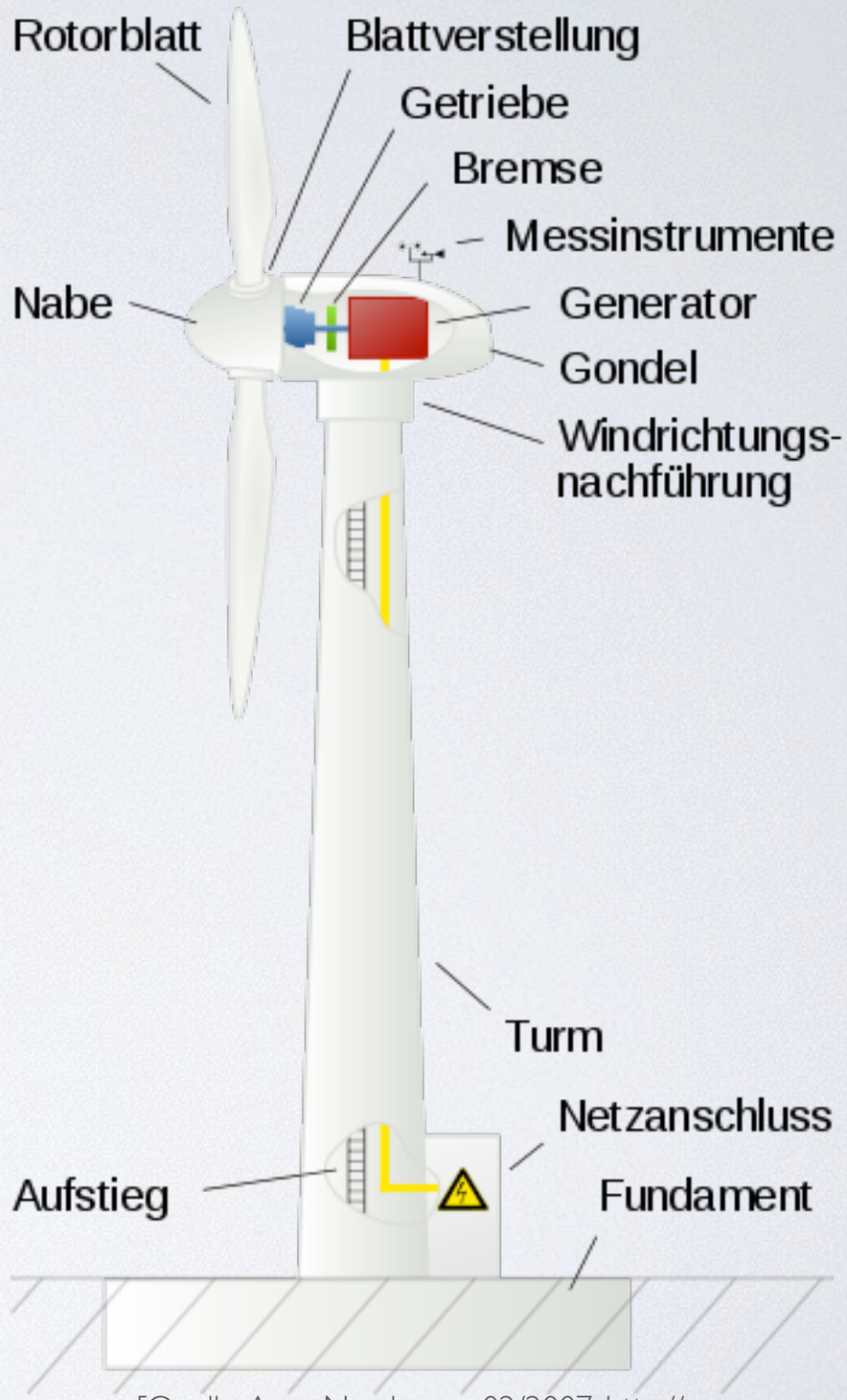


[Quelle: http://upload.wikimedia.org/wikipedia/en/e/ed/MATLAB_Simulink_screenshot_showing_thermostat_demo_simulation.png]

Beispiel einer Anwendung: Windrad

- Eingebettetes System übernimmt
  - Betriebsführung, Überwachung
  - Azimut- und Blattwinkelregelung
- Steuerungskomponenten in
  - Turmfuß
  - Gondel
  - Narbe
  - verbunden über Bus

[Quelle: Koll, Die Steuerung macht's, 2010]



[Quelle: Arne Nordmann, 02/2007, http://commons.wikimedia.org/wiki/File:Windkraftanlage.svg]

Literatur / Quellen

- Andreas **Koll**, Windkraft - Die Steuerungs macht's, Energie&Technik, 02.06.2010,URL:http://www.energie-und-technik.de/smart-grid-smart-metering/technik-know-how/it-infrastruktur/article/27523/1/Die_Steuerung_machts/, abgerufen am 06.05.2012
- Matthew **Hause**, The SysML Modelling Language, Fifth European Systems Engineering Conference, 2006
- Elizabeth **Latronico**, Philip **Koopman**, Representing Embedded System Sequence Diagrams As A Formal Language, UML 2001, Toronto, 2001
- Peter **Marwedel**, Eingebettete Systeme, Springer-Verlag, 2008
- OMG, OMG Systems Modeling Language (OMG SysML), Version 1.2, 2010
- OMG, OMG Systems Modeling Language (OMG SysML), Version 1.3, 2012
- Sourceforge, AVR Ada, URL: http://sourceforge.net/apps/mediawiki/avr-ada/index.php?title=USB_Boarduino, abgerufen am 06.05.2012
- **Wikipedia.org**, Ada (programming language), URL: http://en.wikipedia.org/wiki/Ada_(programming_language)), abgerufen am 26.05.15
- **Wikipedia.org**, SystemC, URL: http://de.wikipedia.org/wiki/SystemC, abgerufen am 26.05.15