Ein generischer Simulator für Systeme mit Strukturdynamik

P. Schwarz[1], C. Clauß[1], C. Nytsch-Geusen[2], A. Nordwig[2], A. Schneider[1], P. Schneider[1], M. Vetter[3]

[1]Fraunhofer-Institut für Integrierte Schaltungen (IIS), EAS Dresden
[2]Fraunhofer-Institut für Rechnerarchitektur und Softwaretechnik (FIRST) Berlin
[3]Fraunhofer-Institut für Solare Energiesysteme (ISE) Freiburg

Kurzfassung

Es wird ein Simulator MOSILAB vorgestellt, der für kontinuierlich-diskrete Systeme geeignet ist. Als Eingabesprache wird Modelica verwendet. Der Simulator unterscheidet sich in wesentlichen Punkten von kommerziell verfügbaren Simulatoren: er soll Systeme mit Strukturdynamik behandeln (d. h. Modelle ändern sich während der Simulation in ihrer mathematischen Struktur) und er ist als ein offenes, modulares System konzipiert, das leicht um neue Algorithmen erweitert werden kann. Außerdem ist es möglich, daraus Spezialsimulatoren für dedizierte Anwendungen zuzuschneiden („generischer Simulator“).

1 Einleitung

Für die Simulation dynamischer Systeme gibt es eine Vielzahl von Simulatoren, Matlab/Simulink und Dymola seien als Beispiele aufgeführt. Darüber hinaus gibt es für spezifische physikalische Domänen weitere, ebenfalls sehr verbreitete Simulatoren (SPICE, Saber und ModelSim in der Elektronik, Adams in der Mehrkörpermechanik, ...). Trotzdem stellt man Defizite fest: für zahlreiche spezifische Anwendungen sind die kommerziellen Simulatoren für kleinere Firmen zu teuer (gerade weil ihr umfangreiches Leistungsvermögen bei weitem nicht ausgenutzt wird), Erweiterungen oder Verbesserungen der Simulationsalgorithmen sind für den Anwender unmöglich und die Kopplung mit anderen Simulatoren wird kaum unterstützt.

Ein wichtiger Punkt ist auch, dass die eingesetzten Modelle während der Simulation nicht oder nur eingeschränkt geändert werden können. Parameteränderungen sind meist möglich, die Zahl der (Differential-)Gleichungen muss dagegen i.d.R. konstant bleiben. Bei der Analyse von komplexen Systemen ist es aber sinnvoll, während der Simulation in bestimmten Zeitintervallen Modelle zustandsabhängig in ihrer Struktur grundlegend zu ändern, u. U. auch zeitweise eine Kopplung mit anderen Simulatoren vorzusehen. Im Abschnitt 2 wird das an einem Beispiel erläutert. Im Unterschied zur üblichen „Multi-Level-Simulation" wird also die Struktur und Genauigkeit der Modelle nicht einmalig vor der Simulation festgelegt, sondern während der Simulation geändert. Aus Gründen des Rechenaufwandes soll jeweils „so genau wie nötig", aber nicht „so genau wie möglich" gerechnet werden. Natürlich ist dieses wünschenswerte Vorgehen mit zahlreichen Problemen verbunden (Eignung der Modellbeschreibungssprache, Umschalten der Simulationsalgorithmen, Sichern konsistenter Übergabewerte zwischen den Intervallen mit unterschiedlichen Modellen, ...).

In einem Verbund von sechs Fraunhofer-Instituten wird ein derartiger Simulator MOSILAB (**Mod**eling and **Si**mulation **Lab**oratory) entwickelt [1]. Der Projektname GENSIM soll auf einen „generischen Simulator" hinweisen, aus dessen Bestand an Modulen Spezialsimulatoren für dedizierte Anwendungen „generiert“ werden können.

2 Anwendungen

Bereits während der Simulatorentwicklung wird das Konzept in drei Anwendungsbereichen erprobt [1]:

- Brennstoffzellensysteme für die stationäre Energieversorgung (z.B. von Gebäuden und Siedlungen),
- Hygrothermische Vorgänge in Gebäuden,
- Spangebende Werkstoffbearbeitung (z. B. durch Bohren und Fräsen).

Durch diese weit gespannten Anwendungen kann gezeigt werden, dass es sich um ein offenes Simulationssystem handelt, in das auch spezialisierte Algorithmen (z. B. das Plugflow-Verfahren aus der Gebäudesimulation [2]) eingebunden werden können. Zusätzlich ist die Kopplung mit anderen Simulatoren wie Matlab/Simulink und FEMLAB – als Beispiel für einen Solver für partielle Differentialgleichungen – vorgesehen [3], [4].

Am Beispiel eines Systems der Gebäudeautomatisierung [5] soll die Behandlung der Modell-Strukturdynamik erläutert werden. Dabei geht es nicht um eine Darstellung der algorithmischen Probleme oder um Details der Modellierung, sondern um die prinzipielle Vorgehensweise aus Anwendersicht. Im Bild 1 ist angedeutet, welche Modellierungsebenen (System, Subsysteme, Komponenten) und welche physikalisch-chemischen Effekte zu betrachten sind. Im Mittelpunkt der Simulationsaufgabe soll die Analyse eines Brennstoffzellensystems stehen, das in Verbindung mit den energetischen Vorgängen im Gebäude und der Steuerungstechnik betrachtet werden muss. Für die Brennstoffzelle (BZ) liegen unterschiedlich genaue Modelle vor. Das Gesamtmodell enthält für alle Teilsysteme (also auch für die BZ) relativ grobe Modelle, mit denen das Systemverhalten schnell und meistens ausreichend genau ermittelt werden kann.

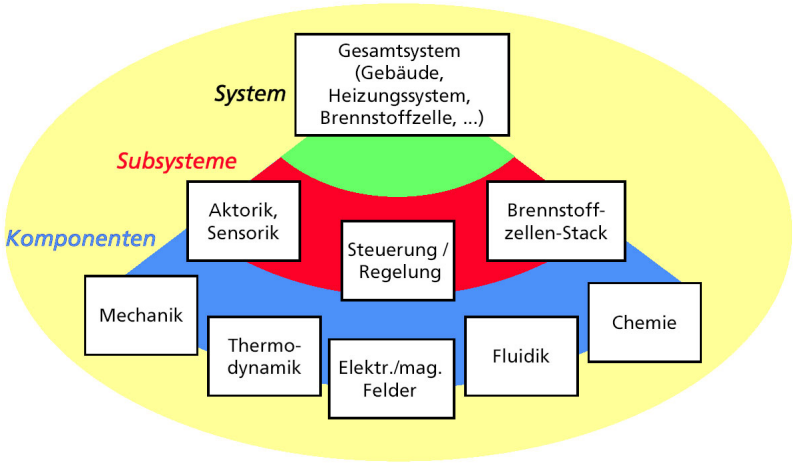

Bild 1: Modellierungsebenen eines komplexen Systems der Gebäudeautomatisierung

Im Bild 2 ist ein Simulationsszenario für das genannte System dargestellt. Es können 6 Zeitintervalle unterschieden werden:

1. Mit dem (groben) Gesamtsystemmodell (a) wird die Simulation begonnen.
2. Es tritt eine Situation ein, in der im Gesamtsystemmodell ein wesentlich präziseres Modell (b) des BZ-Stacks eingesetzt wird. Auch dieses Modell kann mit einem MOSILAB-Algorithmus simuliert werden.
3. Es wird zu der Situation 1. zurückgekehrt: es reicht eine „grobe“ Gesamtsystem-simulation mit dem Modell (a).
4. Hier wird wieder die Genauigkeit des präzisen MOSILAB-Modells (b) der BZ benötigt. (Situation wie im Zeitintervall 2).
5. Es ist immer noch eine präzise Simulation der BZ erforderlich. Allerdings sei jetzt sogar eine noch größere Genauigkeit nötig: die Strömungsverhältnisse in einem Kanal der BZ müssen simuliert werden. Da in MOSILAB kein Algorithmus für die Strömungssimulation implementiert ist, muss ein Modell (c) für einen speziellen Strömungssimulator (CFD – Computational Fluid Dynamics) eingesetzt und dieser mit den anderen Lösungsalgorithmen gekoppelt werden.
6. Es herrschen wieder einfache Verhältnisse: Rückkehr zum einfachen Gesamtsys-temmodell (a) wie in den Zeitintervallen 1. und 3., auch auf den sehr rechenzeit-aufwändigen CFD-Simulator kann verzichtet werden.

Außerdem ist im Bild 2 rechts dargestellt, wie im MOSILAB-Programmsystem die einzelnen Modelle (einschließlich der Simulatorkopplung zu einem CFD-Simulator) eingebettet sind und über Schnittstellen mit dem Simulationskern kommunizieren. Dabei ist es möglich, dass die Modelle die Lösungsalgorithmen aus dem Simulatorkern benutzen oder über eigene Solver verfügen oder ein fremder Simulator aufgerufen wird (Simulatorkopplung).

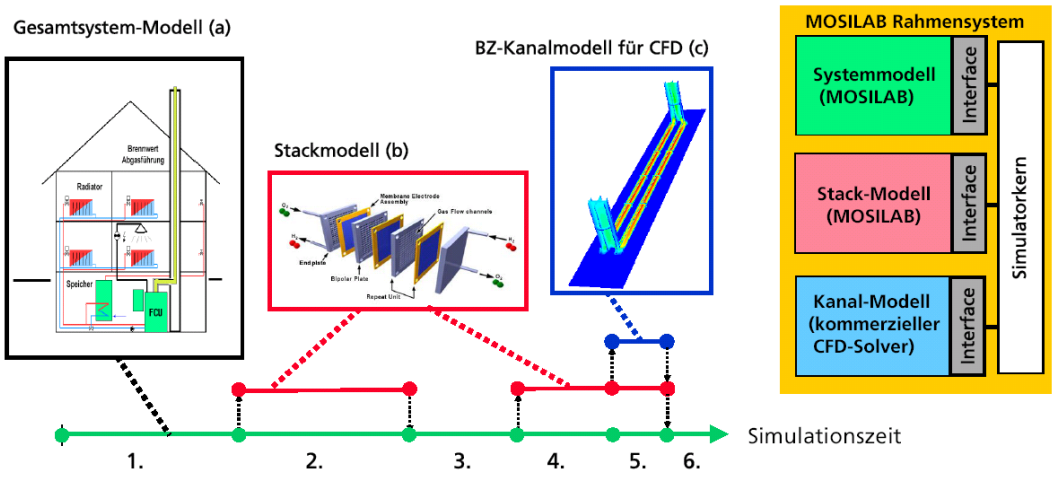

Bild 2: Simulationsszenario mit Strukturdynamik

3 Aufbau des Simulators

Die Struktur des Simulationssystems ist im Bild 3 dargestellt. Es wurde ein kompilierendes Prinzip gewählt, d. h. die Modellbeschreibung wird in ein C++ Programm transformiert, das nach Übersetzung und Linken mit einer Laufzeitumgebung und den Anweisungen zur Simulationssteuerung als ausführbares Programm zur Verfügung steht. Der Anwender kommuniziert mit dem Simulator über ein graphisches Interface (IDE, Interactive Development Environment).

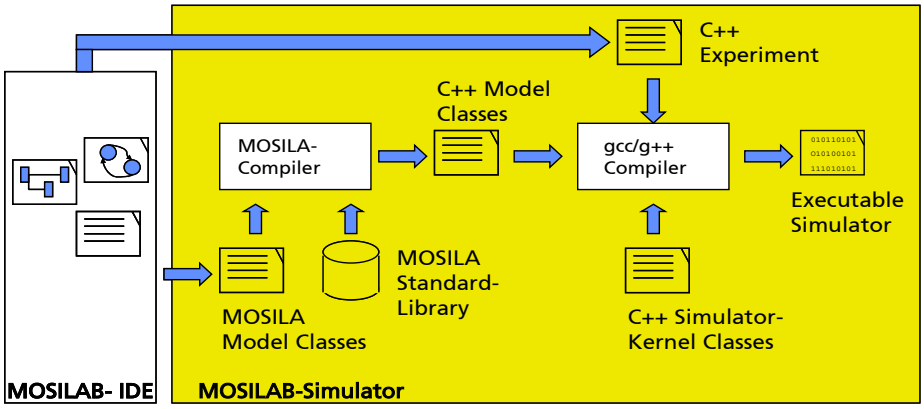

Bild 3: Struktur des Simulationssystems

4 Numerische Lösungsalgorithmen

Die Modellbeschreibung in textueller oder graphischer Form führt bei kontinuierlichen Teilsystemen auf eine interne mathematische Beschreibung durch nichtlineare, meist implizite Differentialgleichungen (DAE, differential-algebraic equations), die durch implizite Integrationsverfahren gelöst werden. Zur Zeit stehen vier Lösungsalgorithmen für Differentialgleichungen bereit: Trapezverfahren, implizites und explizites Euler-Verfahren sowie IDA [6] als leistungsfähiger universeller DAE-Solver (eine Weiterentwicklung von DASSL [7]). Während IDA grundsätzlich für die Lösung nichtlinearer Differentialgleichungen eingesetzt wird, können mit den anderen drei (wesentlich einfacheren) Verfahren im Bedarfsfall spezielle Teilsysteme, z. B. mit wesentlich langsameren dynamischen Verhalten, simuliert werden. Das Integrationsverfahren in IDA ist ein BDF-Verfahren (Backward Differentiation Formula) variabler Ordnung, die dabei zu lösenden nichtlinearen Gleichungssysteme werden mit einem Newton-Verfahren bearbeitet. Die diskreten Teilsysteme werden mit Discrete-Event-Algorithmen behandelt. Die Strukturdynamik der Modelle wird vom Anwender durch Statecharts beschrieben (s. Abschnitt 5); die Berechnung von Statechart-Modellen erfolgt mittels einer speziellen Statechart Simulation Engine, also einem dedizierten Lösungsalgorithmus.

Der grundsätzliche Ansatz der gemeinsamen diskret-kontinuierlichen Simulation (Mixed-Signal-Simulation) ist im Bild 4 dargestellt. Dabei wurde vor allem der Aspekt der Strukturdynamik betont: mit den Informationen aus den zu einem Simulationszeit-

punkt aktiven Statecharts wird das DAE-System des kontinuierlichen Modells aufgebaut und durch Anwendung eines Diskretisierungsverfahrens in ein algebraisches Gleichungssystem (AE) überführt, das numerisch gelöst wird.

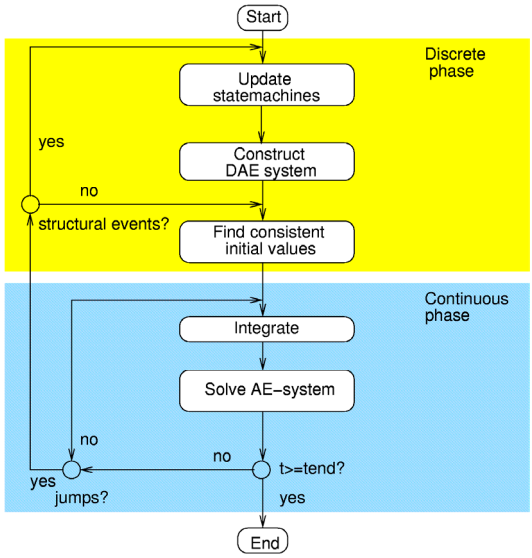

Bild 4: Mixed-Signal-Simulationszyklus

5 Modellbeschreibung und Nutzerinterface (IDE)

Für die Systembeschreibung sollte eine weit verbreitete, möglichst standardisierte Modellierungssprache benutzt werden. Von den in Frage kommenden Sprachen (wie VHDL-AMS, Verilog-AMS und Modelica) wurde **Modelica** (www.modelica.org) gewählt. Diese objektorientierte Sprache passt am besten zum gewählten Modellierungs- und Simulationskonzept, ist stark interdisziplinär ausgerichtet und wird durch Simulatoren wie Dymola unterstützt. Es gibt umfangreiche Modellbibliotheken (z. T. frei verfügbar) und eine internationale Modelica-Arbeitsgruppe sorgt für die Weiterentwicklung der Sprache. Auf Einzelheiten einer Modelica-Beschreibung muss hier nicht näher eingegangen werden, da es dafür genügend Material gibt [10], [11]. Aufbauend auf Modelica wird im Projekt GENSIM geprüft, ob und welche Spracherweiterungen sinnvoll sind, die die Besonderheiten der Modellstrukturdynamik berücksichtigt und objektorientierte Statecharts [8] enthält.

Statecharts sind ein bewährtes Hilfsmittel zur Beschreibung komplexer Systeme [12], [13] und haben zusätzlich durch die Einbettung in UML (Unified Modeling Language) eine weite Verbreitung gefunden. UML wird als Standard unter der Federführung der Object Management Group OMG (http://www.omg.org/gettingstarted/index.htm) entwickelt. Auch im Modelica-Umfeld werden Statecharts verwendet [9], [14], [15]. Im folgenden wird ein Vorschlag für die textuelle Beschreibung von Statecharts in MOSILAB vorgestellt.

Im Bild 5 ist ein Ausschnitt aus einer Statechart-Beschreibung zur Steuerung eines Luftfahrzeuges (Senkrechtstarter) gezeigt. Man erkennt die hierarchische Schachtelung und die Aktionen, die bei den Übergängen zwischen den Zuständen ausgeführt werden.

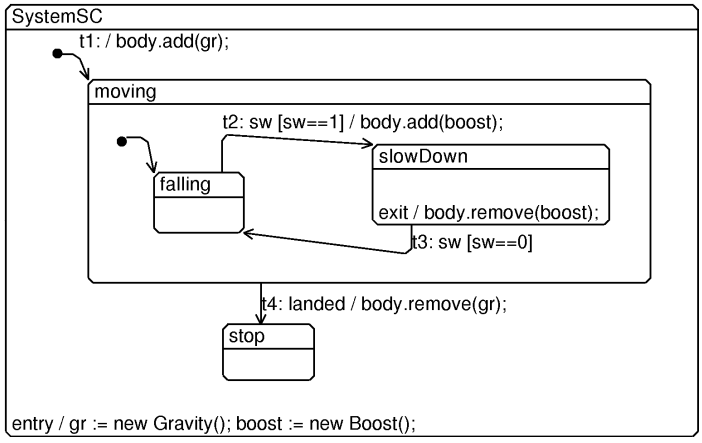

Bild 5: Beispiel einer Modellbeschreibung mit hierarchischen Statecharts

Die textuelle Beschreibung (angepasst an die Modelica-Syntax) lautet:

```
model System
...
statechart
  state SystemSC extends State;              // State ist die Basisklasse

    state Moving extends State;      // Definition des Zustandes "Moving"
      state SlowDown extends State;
        ...
      end SlowDown;
      State falling, State start(isInitial=true);      // Einfuehrung der
      SlowDown slowDown;                               // Zustaende
      ...
      transition t2 : falling -> slowDown         // Zustandsuebergang t2
        event sw guard sw==1 action body.add(boost);
      end transition;
      transition t3 : slowDown -> falling         // Zustandsuebergang t3
        event sw guard sw==0
      end transition;
    end Moving;

    State stop, start(isInitial=true);
    Moving moving;

    entry action    // wird ausgefuehrt, wenn Zustand SystemSC aktiv wird
      gr := new Gravity(); boost := new Boost(empty=false);
    end entry;
    ...
  end SystemSC;
end System;
```

Statecharts können nicht nur zur Beschreibung von Discrete-Event-Systemen allgemein benutzt werden, sondern speziell auch zu Steuerung des Umschaltens zwischen verschiedenen Modellklassen (Modell-Strukturdynamik). Dadurch werden das Wissen und die Zielstellungen des Entwerfers in die Modellbeschreibung eingebracht, weitere Steuerungsmöglichkeiten ergeben sich mit den Kommandos zur Simulatorsteuerung.

Einen Eindruck von der graphischen Oberfläche eines Teils der MOSILAB- Modellierungsumgebung vermittelt Bild 6. Sichtbar sind ein Klassenbrowser, ein Graphikeditor für Klassendiagramme, Statecharts und Kollaborations-Diagramme sowie ein Textfenster zur Eingabe von Modelltexten.

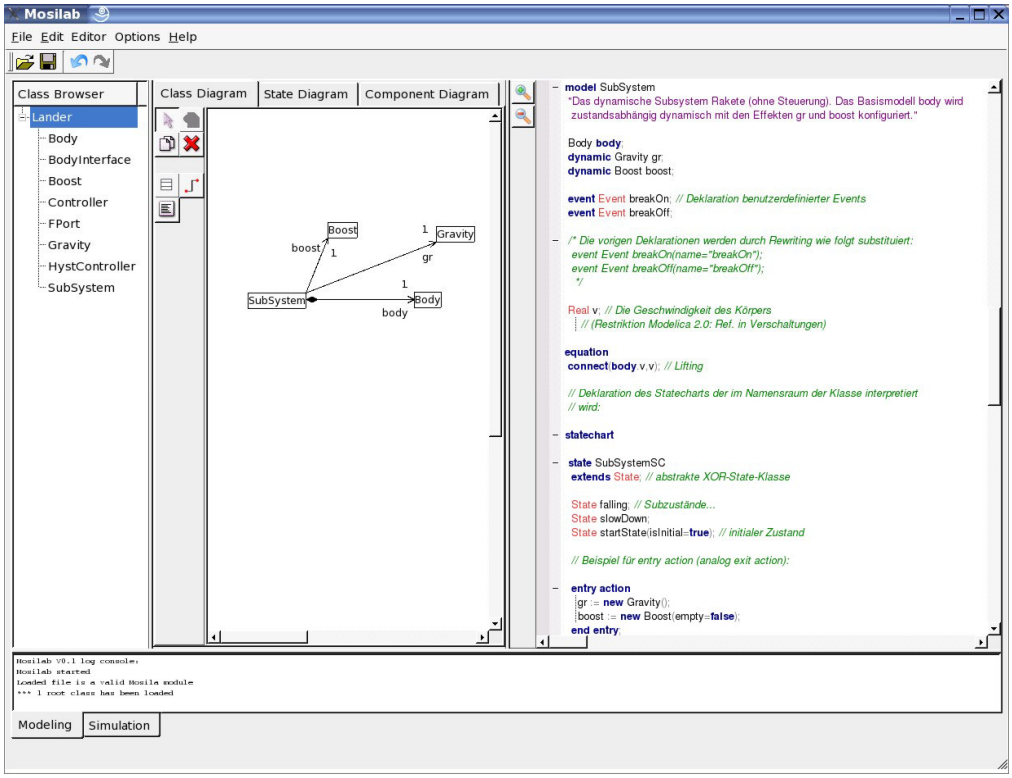

Bild 6: IDE des Simulationssystems im Modellierungs-Modus

Zusammenfassung: Es wurde ein Simulationssystems MOSILAB vorgestellt, dass es durch seine modulare Struktur gestattet, generisch Spezialsimulatoren zu entwickeln und mit anderen Simulatoren gekoppelt zu werden. Die Modellierungssprache baut auf Modelica auf und verwendet hierarchische Statecharts. Aus algorithmischer Sicht, aber auch aus der Sicht komplexer Anwendungen, ist die Unterstützung der Modell-Strukturdynamik hervorzuheben.

6 Literatur

[1] *Nytsch-Geusen, C.et al.*: MOSILAB: Development of a Modelica based generic simulation tool supporting model structural dynamics. 4. Intern. Modelica Conf., Hamburg 2005.

[2] *Wittwer, C.*: ColSim – Simulation von Regelungssystemen in aktiven solarthermischen Anlagen. Dissertation Universität Karlsruhe (TH), 1999.

[3] *Martin, R. u.a.*: Simulatorkopplung mit FEMLAB. FEMLAB Konferenz, Frankfurt, November 2005.

[4] *Clauß, C.; Reitz, S.; Schwarz, P.*: Simulation mechanisch-elektrischer Wechselwirkungen am Beispiel eines sensorischen Mikrosystems. SIM2000 – Simulation im Maschinenbau, Dresden 2000, S. 183-196.

[5] *Muche, L.; Schneider, P..; Vetter, M.; Wittwer, C.*: Modellierung und Simulation der Energieversorgung von Gebäuden mittels Brennstoffzellensystem. 5. GMM/ITG/GI-Workshop "Multi-Nature Systems", Dresden 2005, S. 25-32

[6] *Hindmarsh, A. C. et al.*: SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. ACM Transactions on Mathematical Software, 2005.

[7] *Petzold, L.R.*: A description of DASSL: A differential/algebraic system solver. IMACS Trans. Scientific Computing Vol. 1 (1993), pp. 65-68.

[8] *Nordwig, A.:* Integration von Sichten für die objektorientierte Modellierung hybrider Systeme. Dissertation, Verlag dissertation.de, Berlin 2003.

[9] *Ferreira, J.A.; Estima de Oliveira, J.P.*: Modelling hybrid systems using Statecharts and Modelica. 7th IEEE Intern. Conf. on Emerging Technologies and Factory Automation, Barcelona 1999.

[10] *Tiller, M.*: Introduction to Physical Modeling with Modelica. Kluwer, Dordrecht 2001.

[11] *Fritzson, P.*: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley, Chichester 2004.

[12] *Harel, D.*: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming 8 (1987), 231-274.

[13] *Harel, D., Politi, M.*: Modeling Reactive Systems with Statecharts: The STATEMATE Approach. McGraw-Hill, New York 1998.

[14] *Elmqvist H., Mattsson S.E., Otter M.*: Object-oriented and hybrid modeling in Modelica. Journal Europeen des systemes automatises (JESA) 35(1), I–X (2001).

[15] http://www.dynasim.se/ModelicaStandardLibrary/help/Modelica_StateGraph.html , enthält eine vollständige Bibliothek