# Modellbasierter Entwurf von Steuerungen und Regelungen

Urban Brunner, Hochschule Karlsruhe

**Keywords**
Model-based design, rapid prototyping, incremental design/improvement, hierarchical object-based modelling, supervisory control, hybrid simulation, automatic code generation, model reduction, design of reduced-order controllers.

## 1. Einleitung

Heute wird der sichere, wirtschaftliche und umweltschonende Betrieb großer Anlagen gefordert und dies zu Recht. Ein solcher Betrieb ist dank den verbesserten Produktionsverfahren einerseits und den enormen Fortschritten der Mikroelektronik andererseits möglich geworden. In der Zukunft wird die Konkurrenzfähigkeit von Prozessanlagen in noch vermehrtem Maße durch deren Kosten für Engineering und Unterhalt bestimmt. Der Einsatz mächtiger Werkzeuge und die Wiederverwendung von Steuerungssoftware sollen Entwicklungszeiten („Rapid Prototyping") verkürzen und so Kosten senken.

Als Antwort auf den zunehmenden Druck zur Wiederverwendung von Steuerungssoftware und die Forderung nach (verifizierbar) korrektem Steuerungscode wurden in den letzten zehn Jahren verschiedene Methoden zur Strukturierung des Entwurfs von Prozesssteuerungen vorgeschlagen. Konzeptionell orientierte man sich dabei im Wesentlichen am (erfolgreichen) modellbasierten Reglerentwurf, obwohl in der diskreten Steuerungswelt kein Maß für Nähe und somit auch keine „Robustheit" existieren. Unter einem modellbasierten Steuerungsentwurf wird meist die automatische Erzeugung des Steuerungscodes mittels eines Werkzeugs (Compiler) anhand einer formalen Beschreibung des Steuerungsziels und eines Modells des zu steuernden Prozesses verstanden. Die eigentliche Steuerungsaufgabe „reduziert" sich dann auf eine formale Beschreibung des Steuerungsziels und des zu steuernden Prozesses. Die zur Lösung dieser Aufgabe verwendete Methodik muss für deren Akzeptanz in der Praxis plausibel sein, sodass der Spezifikationsnachweis auch für Nicht-Experten verständlich und überprüfbar ist.

Der gesteuerte Prozess ist ein gemischt kontinuierlich-diskretes System und dessen Verhalten kann bekanntlich unerwartete Phänomene aufweisen Daher genügt in vielen Anwendungsfällen eine auf der Endlichkeit des Zustandsraums beruhende Erreichbarkeitsanalyse mit automatischer Codegenerierung nicht; d.h. der Steuerungscode muss auch validiert werden.

Unter diesem Gesichtspunkt wird im ersten Teil dieses Beitrags eine auf der Steuerungssynthese von Wonham & Ramadge [1] beruhende Methodik vorgeschlagen und mit andern Vorgehensweisen verglichen. Die Modellbildung erfolgt hierarchisch objekt-basiert. Das für die automatische Codegenerierung benötigte Modell des gesteuerten Prozesses wird schrittweise durch sukzessives Verhindern unerwünschter Zustandsübergänge des zu steuernden Prozesses erhalten, so dass das Steuerungsziel inkrementell und zugleich durch (hybride) Simulation abgesichert erreicht wird.

Im zweiten Teil des Beitrags soll der modellbasierte (Steuerungs-)Entwurf aus einer etwas grundsätzlicheren Sicht betrachtet werden. Die Schritte „Modellbildung" und „Steuerungssynthese" sind eigentlich nicht unabhängig und es stellt sich die Frage, wie genau/detailliert muss das Modell sein bzw. wie findet man ein geeignetes Modell. Hier wird das geeignete Modell iterativ gefunden. Eine vergleichbare Problematik beinhaltet der Entwurf eines reduzierten Reglers für ein System hoher Ordnung; denn auch hier sind Modell-(Reduktion) und Reglersynthese gekoppelt. Reduzierte Regler können in der Regel nicht direkt bestimmt werden und das Problem wird als ein Optimierungsproblem formuliert. Dabei ist eine geeignete Parametrisierung besonders hilfreich. Die Vorstellung einer entsprechenden Methode zum Entwurf reduzierter Regler wird den Beitrag abschließen.

## 2. Systematischer Steuerungsentwurf mit automatischer Codeerzeugung

In der Vergangenheit wurde die Steuerung einer prozesstechnischen Anlage vergleichsweise wenig strukturiert entwickelt. Als Strukturierungsmittel benützte man die in EN 61131-3 genormten graphischen SPS-Programmiersprachen „Funktionsbausteinsprache“ und „Ablaufsprache“. Die Wiederverwendung von mächtigeren Softwarekomponenten kann nur durch weitere Abstraktion, d.h. Modellierung, erreicht werden. Wegen des Wunsches aus der Praxis nach Verwendung bereits eingeführter Werkzeuge sowie wegen ihrer Anschaulichkeit und formalen Analysierbarkeit einerseits und der engen strukturellen Beziehung zur Ablaufsprache anderseits basieren viele der neueren Steuerungsentwurfsmethoden auf Petri-Netzen (vgl. z.B. [2]).

Grundlage der Verifikation einer Steuerung mittels Erreichbarkeitsanalyse sowie Ausgangspunkt der automatischen Codegenerierung ist eine formale Beschreibung des gesteuerten Prozesses, beispielsweise in Form eines Petri-Netzes [2]. Anhand dieses Petri-Netzes können durch Konstruktion des Erreichbarkeitsgraphen der entsprechende „Endliche Automat" erhalten –und anschließend analysiert– bzw. durch Abwicklung des Petri-Netzes (Markenspiel) die gesuchte Steuerungssequenz und somit die Steuerung gefunden werden. Die vorgeschlagenen Methoden unterscheiden sich im Wesentlichen in der Konstruktion des Modells des gesteuerten Prozesses. So wird in [2] ein Petri-Netz Modell des gesteuerten Prozesses aufgrund physikalischer und ingenieurmäßiger Überlegungen direkt aufgestellt, während in [1] das Modell des gesteuerten Prozesses anhand eines Modells des ungesteuerten Prozesses und einer formalen Beschreibung des Steuerungsziels implizit bestimmt und daraus die Steuerung (sog. „Supervisor") berechnet werden.

## 3. Eine Entwurfsmethodik für Prozesssteuerungen

Die in diesem Kapitel vorgestellte Methodik beruht auf der Steuerungssynthese von Ramadge & Wonham; jedoch werden zur Prozessbeschreibung anstelle „Regulärer Sprachen" sog. „Statecharts“ [3] benutzt. Im Gegensatz zu den (klassischen) Zustandsautomaten können Satatecharts hierarchische Strukturen aufweisen, was bei der Modellbildung sowohl ein sukzessives Verfeinern von Teilmodellen (Top-down) als auch ein Aggregieren/Zusammenfassen zu Teilmodellen (Bottom-up) erlaubt. Das für die automatische Codegenerierung gesuchte Modell des gesteuerten Prozesses und somit die gesuchte Steuerung werden durch folgende Schritte entwickelt.

### 1. Schritt: Hybride Modellbildung des ungesteuerten Prozesses

Der ungesteuerte Prozess wird durch ein hybrides (z.B.Simulink/Stateflow) Modell beschrieben. (Aus Platzgründen wird hier auf das Beispiel in [4] verwiesen.) Das hybride Modell besteht aus dem zeit-kontinuierlichen Teil, der den zu steuernden Prozess beschreibt, und einer Statechart, die diesen kontinuierlichen Prozess durch ein diskretes Ereignis-getriebenes System approximativ beschreibt und den „Handbetrieb“ der Anlage ermöglicht. Regelungstechnisch kann diese Statechart als „zweckmäßiger" diskreter (Ereignis-getriebener) Beobachter für den kontinuierlichen Prozess interpretiert werden. Selbstverständlich kann dieser Beobachter später auch für Prozessvisualisierungsaufgaben herangezogen werden.

### 2. Schritt: Hybride Modellbildung des gesteuerten Prozesses

Im zweiten Schritt wird der gesteuerte Prozess durch ein hybrides Modell beschrieben. Dazu müssen lediglich der Handbetrieb durch eine Steuerung, bzw. die bestehende Statechart durch eine entsprechende ersetzt werden. Bekanntlich können im gesteuerten Prozess nur solche Ereignissequenzen ablaufen, die auch im ungesteuerten möglich sind; d.h. mit $M_u$ als Bezeichnung der Menge der Ereignissequenzen des ungesteuerten Prozesses und mit $M_g$ für die Menge der Ereignissequenzen des gesteuerten Prozesses gilt:

$$M_g \subseteq M_u$$

Die neue Statechart erhält man daher aus der bestehenden Statechart, indem man deren Transitionen an entsprechende Bedingungen knüpft und ggf. Aktionen einführt, sodass unerwünschte Zustandsübergänge nicht mehr ablaufen können. Dabei sollen diese zusätzlichen Bedingungen das ursprüngliche Verhalten möglichst wenig einschränken. Durch dieses sukzessive Blockieren von Zustandsübergängen wird

die gesuchte Steuerung schrittweise konstruiert, wobei der jeweils erreichte Stand der Spezifikationserfüllung mittels hybrider Simulation überprüft und abgesichert wird. Es kann aber auch der Fall eintreten, dass das diskrete Teilmodell aus Schritt 1 zu grob ist und das spezifizierte Ziel nicht erreicht werden kann. In diesem Fall muss das Teilmodell gezielt verfeinert werden, was durch ein iteratives Ausführen der Schritte 1 und 2 erreicht werden kann (vgl. auch Kapitel 4). Aus konzeptioneller Sicht kann dieser zweite Schritt als das Festlegen der Rückführungen beim Reglerentwurf aufgefasst werden.

3. Schritt: Validierung der Steuerung

Das Verhalten der gesteuerten Anlage wird nun mittels Simulation analysiert. Sollte es sich dabei zeigen, dass beispielsweise die Spezifikation unvollständig gewesen ist, muss die Spezifikation entsprechend geändert werden. Selbstverständlich muss nur Schritt 2 erneut durchgeführt werden, während das Modell des ungesteuerten Prozesses aus Schritt 1 in der Regel unverändert wieder verwendet werden kann. Es könnte aber auch vorkommen, dass das Erfüllen einer stringenteren Spezifikation eine feinere Diskretisierung erfordert und das diskrete Teilmodell in Schritt 1 entsprechend modifiziert werden muss.

4. Schritt: Automatische Code-Generierung

Mit den MATLAB-Tools *Real-Time Workshop*® und *StateflowCoder*® kann aus der in Schritt 2 entwickelten Statechart C-Code erzeugt werden. Da die Simulation von Schritt 3 und der generierte Code auf demselben Modell beruhen, sind Simulation und Code konsistent; d.h. der Steuerungscode ist korrekt (Verifikation). Schließlich wird aus dem C-Code der Code für die Zielhardware (SPS, Mikrokontroller oder PC) generiert. Bei diesem Vorgang können auch benötigte Treiber für spezielle Peripherie eingebunden werden.

5. Schritt: Hardware-in-the-loop Simulation

Schließlich können durch eine Hardware-in-the-loop Simulation die Steuerung und insbesondere die Rechenzeitverhältnisse überprüft werden. Dazu wird auf dem PC der kontinuierliche Prozess anhand des kontinuierlichen Teilmodells aus Schritt 1 simuliert und über I/O-Karten mit der Zielhardware verbunden, wo der generierte Code abläuft.

## 4. Die Modellbildung als Schlüssel des Entwurfs

Die hier beschriebene Methodik entspricht dem in der Praxis als Rapid Prototyping bezeichneten Vorgehen, wobei die Modellbildung die aufwendigste –weil nicht gänzlich automatisierbare– Teilaufgabe darstellt. Die in Schritt 1 gesuchte Diskretisierung (engl. partitioning) des kontinuierlichen Zustandsraums ist eine Approximation, d.h. eine Abstraktion des kontinuierlichen Prozesses. Dabei stellt sich zwangsläufig die Frage, wie genau muss diese Approximation sein, um das Steuerungsziel erfüllen zu können. Die Modellbildung und der formale Steuerungsentwurf sind nicht unabhängige Teilprobleme eines Entwurfs. Ein solch gekoppeltes Problem löst man meist iterativ. In [5] wird iterativ eine immer feinere Diskretisierung erreicht, bis schließlich das Steuerungsziel garantiert werden kann. In einem gewissen Sinne durchaus vergleichbar mit dem Vorgehen in [5] werden auch bei der hier vorgeschlagenen Methodik die „zweckmäßige" Diskretisierung (d.h. das geeignete Modell) und die gesuchte Steuerung iterativ gefunden. Voraussetzung dazu ist ein Simulationstool, das eine inkrementelle Modellbildung unterstützt.

Die grundsätzliche Problematik des modellbasierten Entwurfs zeigt sich besonders anschaulich beim Entwurf von „reduzierten" Reglern für Prozesse hoher Ordnung. Moderne Regelungsmethoden benötigen bekanntlich ein internes Modell („Beobachter") der zu regelnden Strecke. Bei verschiedenen Anwendungen (z.B. bei Flugzeugen als Back-up-Regler) ist man aber an einfach zu realisierenden Reglern interessiert. Für den Entwurf eines vereinfachten Reglers sind prinzipiell die beiden folgenden Vorgehensweisen denkbar; nämlich die „Prozessreduktion" (zuerst Reduktion des Prozessmodells und dann Reglersynthese anhand des reduzierten Modells) und die „Reglerreduktion" (zuerst Reglersynthese und dann Reduktion des Reglers hoher Ordnung). In [6] wird anhand eines optimalen Reglers mit vorgegebener Ordnung gezeigt, dass die Riccati-Gleichung für eine LQ-optimale Regelung und die Lyapunov-Gleichungen für die Ordnungsreduktion mittels balancierter Realisierung gekoppelt sind und simultan gelöst werden müssen. In

Kapitel 5 soll kurz ein Verfahren vorgestellt werden, bei dem der vereinfachte Regler mittels Iteration bestimmt wird.

## 5. Methode zum Entwurf vereinfachter Regler für Systeme hoher Ordnung

Die im Vortrag vorgestellte Methode kann hier wegen der Platzbeschränkung nur grob skizziert werden. Das Entwurfsmethode beruht auf der in [7] eingeführten „closed-loop Reduktion“ und deren Eigenschaft, dass mit wachsendem Approximationsparameter k neben den Polen die Nullstellen Systems *G(s)* und somit dessen closed-loop Verhalten besser approximiert werden (vgl. Kasten 1).

Ausgehend von der Tatsache, dass der resultierende Regelkreis bestehend aus dem vereinfachten Regler und dem Prozess hoher Ordnung eine gewisse Robustheit aufweisen muss, wird iterativ ein Regelziel gesucht, sodass bei der Reduktion die Robustheitseigenschaft (hier eine Phasenreserve von > 60°) erhalten bleibt. Schritt 1 und Schritt 2 sind in Kasten 2 zusammengefasst. Im 3. Schritt wird geprüft, ob der approximierte loop-gain $G_{ap}(s)$ die Robustheitseigenschaft erfüllt oder nicht. Falls ja ist das Regelziel gefunden, andernfalls wird das Regelziel (implizit) angepasst, indem die lineare Zustandrückführung $\underline{f}'$ durch $k\underline{f}'$ mit $k>1$ (z.B. $k = 1{,}3$) ersetzt wird:

$$\underline{f}_{j+1} = k * \underline{f}_j \quad , \text{wobei } k > 1$$

Dann wird der 2. Schritt (die Approximation/ Reduktion) erneut ausgeführt. Man beachte, dass dabei der loop-gain $G_{i+1}(s) = kG_i(s)$ ein „LQ-optimaler“ loop-gain bleibt und die (Phasen-)Approximation mit jedem Iterationsschritt besser wird, sodass nach wenigen Iterationsschritten das geeignete Regelziel gefunden wird. Der vereinfachte Regler wird dann durch eine einfache Rechenoperation erhalten [7]. Eine zu lange Iteration ist ein Hinweis, dass die Ordnung des approximierten loop-gain $G_{ap}(s)$ bzw. der approximierten Strecke $P_{ap}(s)$ zu niedrig gewählt wurde.

**Literatur**:


[1] Ramadge, P.J. und Wonham, W.M.: The Control of Discrete Event Systems. Proc. of the IEEE, Vol. 77(1), 1989, S. 81-98.

[2] Litz, L. und Frey, G.: Methoden und Werkzeuge zum industriellen Steuerungsentwurf - Historie, Stand, Ausblick. atp 4/99, S. 145-156.

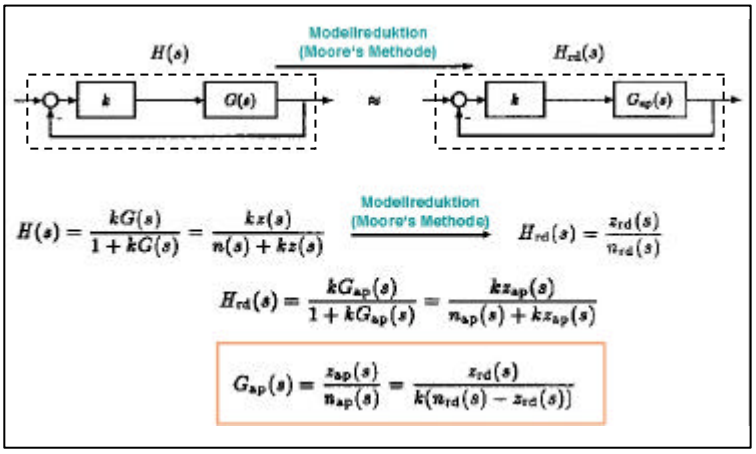
[3] Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming, Vol. 8, 1987, S 231.274.

[4] Hoffmann, J. und Brunner, U.: Matlab & Tools für die Simulation dynamischer Systeme. Addison-Wesley, München, 2002.

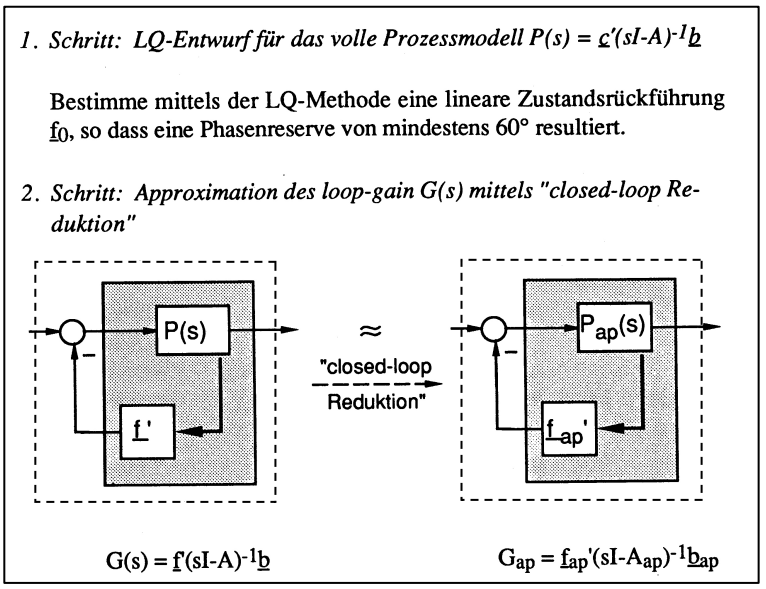
[5] Moor, T., Raisch, J., O'Young, S.: Supervisory control of hybrid systems via *l*-complete Approximations. Proc. of the fourth Workshop on Discrete Event Systems, 1989.

[6] Hyland D.C., and Bernstein D.S.: The optimal Projection Equations for Model Reduction and the Relationships among the Methods of Wilson, Skelton, and Moore. IEEE Trans. on Autom. Control, Vol.30, No.12, pp.1201-1211.

[7] Brunner, U.: New method for the design of a reduced-order controller. INT. J. CONTROL, Vol. 52(5), 1990, S. 1065-1082.


**Kasten 1:** *„closed-loop Reduktion”*




**Kasten 2:** *Schritte 1 und 2 der Entwurfsmethode*