Digital Mock-up bei der Entwicklung mechatronischer Produkte

Willy Schweiger, Achim Schön
Friedrich-Alexander-Universität Erlangen-Nürnberg

1. Zusammenfassung

Nach Betrachtung systemtheoretischer Anforderungen, wird der Entwicklungsprozeß eines mechatronischen Produkts in die Konstruktionsmethodik eingebettet. Anschließend erfolgt ein Überblick zur deren numerischen Simulation. Über formale Beschreibungsmethoden der einzelnen Teildisziplinen, leitet sich die Beschreibung des Gesamtsystems ab, welche durch ein Beispiel verdeutlicht wird.

2. Einführung

Digital Mock-up (abgekürzt DMU) ist eine Strategie zur Verbesserung und Verkürzung der zur Entwicklung industrieller Produkte ablaufenden Prozesse. Nach Webster's Int'l Dictionary bedeutet mock-up "a structural model built accurately to scale (in Orginalgröße), chiefly for study, testing or display". Ein DMU ist demnach ein entsprechendes digitales Modell (un mannequin digitale), welches alle Produktfacetten durch entsprechende Datenstrukturen widerspiegelt. DMU fordert, durch massiven Rechnereinsatz ein Produkt zu entwickeln, sowie dessen Funktionalität und sein Verhalten während seiner Betriebsdauer zu optimieren. DMU *muß* vom Moment der Produktidee beginnend eingesetzt werden, soll eine entscheidende Reduktion der Anzahl der Entwicklungszyklen erreicht werden.

Der Einsatz von DMU ist nicht produktspezifisch. Jedoch ist gerade bei der Entwicklung mechatronischer Produkte (abgekürzt MP) ein besonders hohes Verbesserungspotential durch DMU-Einsatz zu erwarten. Nach einer Definition der Johannes-Kepler-Universität, Linz bedeutet Mechantronik "die ganzheitliche Entwicklung von Systemen aus technischen Komponenten ("Mecha"), die mit einer intelligenten Steuerung ("tronik") versehen sind." Die angesprochene Erwartungshaltung resultiert nicht zuletzt aus diesem inhärent interdisziplinären Charakter mechatronischer Produkte

Der folgende Beitrag beschäftigt sich ausschließlich mit dem Gesichtspunkt der Produktfunktionalität. Es wird zunächst eine "mechatronische" Erweiterung der klassischen Konstruktionsmethodik versucht und im weiteren Simulationsmöglichkeiten im Entwicklungsprozeß, insbesondere in dessen frühen Phasen skizziert.

3. Mechatronische Produkte

3.1. Systemtheoretischer Hintergrund

Eine abstrakte Beschreibung für ein technisches System ist durch ein Ein-Ausgabe-System entsprechend Abb. 1 gegeben ([Hub84; Pah97]). Der Eingabevektor E wird dabei durch das System in den Ausgabevektor A mittels einer abstrakten Funktion F transformiert, abgekürzt $A \leftarrow F(E)$.

Diese abstrakte Funktion F muß im Falle eines technischen, und speziell eines mechatronischen Systems mindestens drei Eigenschaften aufweisen:

- F muß ohne Bezugnahme auf den Eingabevektor E erklärbar sein, d.h. F muß *Operatoreneigenschaft* besitzen. Die mit einem Operator verbundene Trennung zwischen

Systemverhalten und Eingang ist für praktische Anwendungen unbedingt notwendig. Nur dann ist es möglich, für die Operatoren Bausteine zu entwickeln, die in sich geschlossen konzipiert werden können, ohne daß auf konkrete Eingangsgrößen Bezug genommen werden muß.

- Zwischen Ein- und Ausgabehistorie muß ein *kausaler Zusammenhang* bestehen. Für eine Eingabehistorie E(-∞,t) soll die Ausgabehistorie vom selben Zeitintervall abhängen, also es soll A(-∞,t) sein (t Zeit). Dies bedeutet, daß der Ausgabevektor nicht von zukünftigen Werten der Eingabe abhängen darf. Auch diese Voraussetzung ist bei technischen Systemen erfüllt.
- Bei einem mechatronischen System enthält F mindestens eine Zustandsvariable, d.h. es existiert eine systeminterne Rückkopplung oder in andern Worten, das System besitzt eine Art Gedächtnis, dessen Umfang von der Ordnung des Systems abhängt (fading memory).

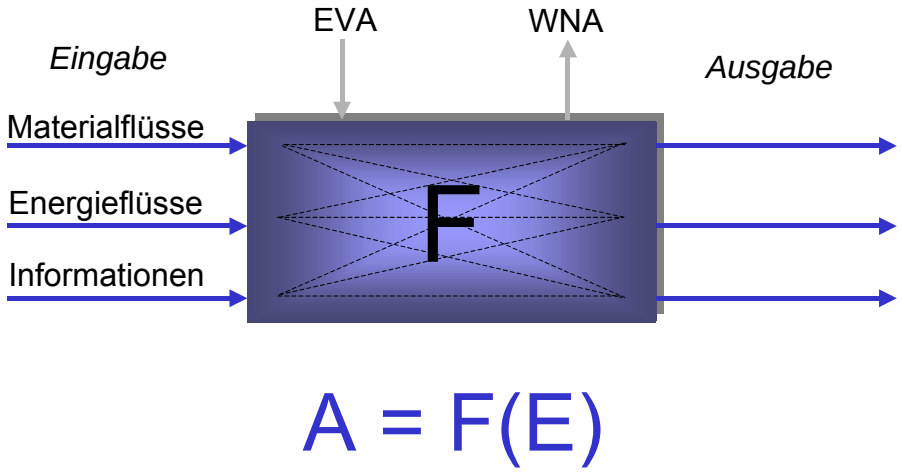
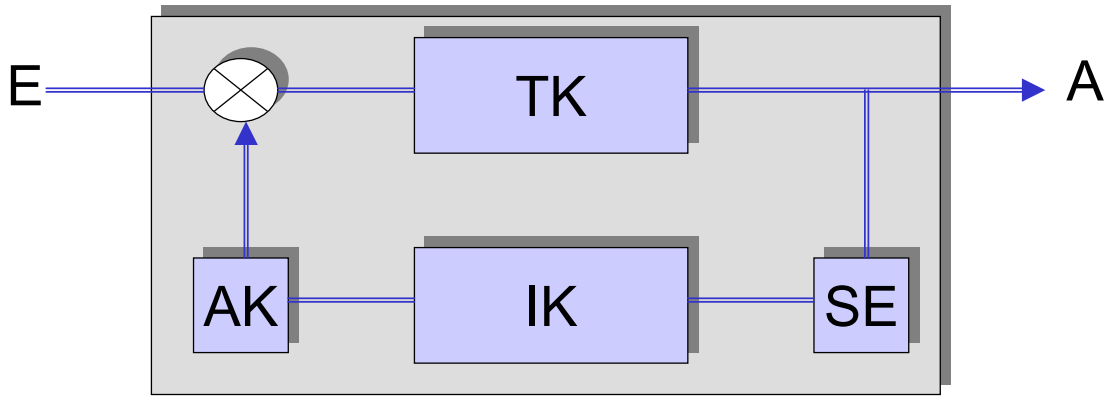


Abb. 1: Abstrakte Beschreibung eines technischen Systems

Mit diesen Forderungen kann als *Referenzarchitektur* eines MPs ein spezielles adaptives Regelsystem entsprechend Abb. 2 verwendet werden. Anstelle des üblichen Reglers ist eine *informationstechnische Komponente* (abgekürzt IK) vorhanden, welche über *Sensoren SE* Informationen erhält und die ihrerseits über *Aktuatoren AK* die von ihr verarbeiteten Informationen zur Steuerung der technischen Komponente einsetzt. Ein sehr ähnlicher Systemaufbau wird auch in [Wal95] beschrieben. Die technische Komponente MS kann ein weitgehend beliebiges physikalisches System sein.



Legende

TK	Technische Komponente
IS	Informationstechnische Komponente
AK	Aktuator(en)
SE	Sensor(en)

Abb. 2: Referenzmodell eines mechatronischen Produkts

3.2. Konstruktionsmethodische Überlegungen

Im Rahmen der klassischen Konstruktionsmethodik ist der Ausgangspunkt in der Konzeptphase die Aufstellung einer geeigneten Funktionsstruktur zur Lösung des gestellten Problems. Die Funktionsstruktur wird soweit in Teilfunktionen untergegliedert, bis Lösungen für die Erfüllung der Teilfunktionen angegeben werden können. Kombiniert man diese Teilfunktionen mit lösungsneutralen physikalischen Effekten, so erhält man die sogenannte Wirkstruktur für die Teilfunktion. Bis zu diesem Zeitpunkt ist der Entwurf funktionsorientiert. Im folgenden, gestaltorientierten Teil des Entwurfs wird durch Ergänzung der Wirkstruktur mit Gestaltungs-merkmalen das Lösungsprinzip für die fragliche Teilfunktion erhalten.

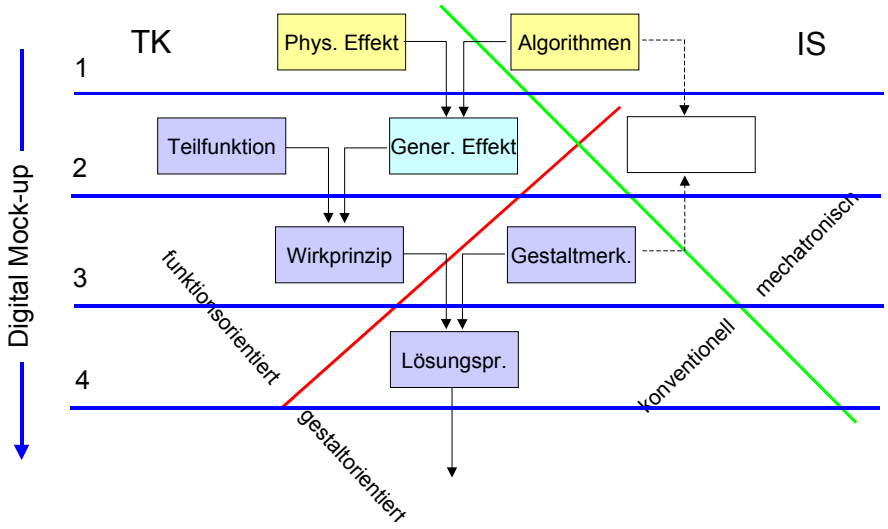


Abb. 3: Konstruktionsmethodische Einbettung

Im Falle mechatronischer Produktentwicklung muß dieser Ansatz dahingehend erweitert werden, daß dem informationstechnologischen Aspekt entsprechend Rechnung getragen wird. Dies kann nur durch eine Hinzunahme von *Algorithmen* als Pendant zu den physikalischen Effekten vollzogen werden. Physikalische Effekte, kombiniert mit Algorithmen

bilden dann *generalisierte Effekte*, deren Einbettung in die allgemeine Konstruktionsmethodik in Abb. 3 dargestellt ist. Diese Einbettung zeigt sehr deutlich, daß Algorithmen der frühen Konzeptphase zuzurechnen sind und damit auch einen wesentlichen Bestandteil der Anforderungsliste für ein MP darstellen. Daraus resultiert aber auch die Notwendigkeit, Simulationen im Sinne eines DMU von Anfang an *nur zusammen* mit den informationstechnologischen Komponenten durchzuführen.

4. Grundlagen zur numerischen Simulation mechatronischer Produkte

Im Sinne von DMU soll hier nur die numerische Simulation fokussiert werden. Gemischte Simulationsmodelle ("Hardware in the Loop") oder die in der Realität erforderlichen Echtzeitsimulationen sollen zunächst unberücksichtigt bleiben, da insbesondere in den frühen Phasen des Entwicklungsprozesses davon ausgegangen werden muß, daß überhaupt noch keine Hardware zur Verfügung steht. Andererseits wurde für DMU die Forderung nach einer integrierten Simulation aufgestellt.

4.1. Ingenieurwissenschaftliche Theorien in der Entwicklung von MP

Es gibt zwei und nur zwei grundlegende physikalische Theorien, welche zur Beschreibung physikalisch-technischer Phänomene und damit bei der Entwicklung von MP von Interesse sind: Die *Rationale Mechanik* (einschl. Thermodynamik) und die *Elektrodynamik*. Beide sind sogenannte Feldtheorien. Durch entsprechende Approximationen werden daraus die Ingenieurtheorien - Technische Mechanik und Elektrotechnik - abgeleitet. Im Falle mechatronischer Entwicklungen treten beide Theorien in der Regel miteinander gekoppelt auf.

Diese Verzahnung der beiden Theorien manifestiert sich einerseits in der wechselseitigen Kopplung mechanischer und elektromagnetischer Einflüsse in den axiomatischen Grundgleichungen (Masse, Impuls, Drall, Energie, magnetische Induktion, Durchflutungsgesetz, Induktionsgesetz, dielektrische Verschiebung) und andererseits im stofflichen Verhalten der involvierten Materialien (die abhängigen Variablen sind Funktionen *aller* unabhängiger Variabler, z.B. ist der Spannungstensor nicht nur eine Funktion der Deformation und der Temperatur, sondern auch abhängig von magnetischer und elektrischer Feldstärkenverteilung, vgl. Piezoeffekt).

4.2. Simulationswerkzeuge

Den verschiedenen Fragestellungen in den frühen Phasen des Entwicklungsprozesses entsprechend, existieren auch unterschiedliche Simulationswerkzeuge für DMU. In den einzelnen Entwicklungsstufen sind *Bewertungen* der Simulationsergebnisse zum Zwecke des weiteren Vorgehens vorzunehmen. Allein aus dieser Tatsache resultiert die Forderung nach unterschiedlich mächtigen Simulationsmethoden. Legt man die Bewertungsstufen entsprechend den Entwicklungsphasen zu Grunde (die Bewertungskriterien selbst sind nicht Gegenstand der vorliegenden Untersuchungen) so lassen sich - ohne Anspruch auf Vollständigkeit - die wichtigsten Methoden wie folgt einordnen (vgl. dazu Abb. 3):

Erste Bewertungsstufe
Hier stehen PE-AL-Tests (Modellierung eines bestimmten physikalischen Effekts PE durch einen Algorithmus AL, z.B. Modellierung von Hystereseeffekten) im Mittelpunkt.

Zweite Bewertungsstufe
In dieser Stufe sind die klassischen Dimensionierungs- und Auslegungsmodelle angesiedelt, aber z.B. auch die Optimierung von Fuzzy-Controllern.

Dritte Bewertungsstufe
Für die Simulation von Wirkprinzipien mit rudimentären Gestaltungsmerkmalen existiert eine Reihe von unterschiedlichen Fragestellungen und damit auch von unterschiedlichen Simulationswerkzeugen.
Für die Simulation physikalischer Phänomene:
- MKS-Systeme (Mehrkörperdynamik mit Primitiva),
- Blockschaltbilder (z.B. [Cel91]) (Zustandsraumdarstellung oder bei linearen Systemen auch Abbildungen durch Funktionaltransformationen),
- Bondgraphen als einheitliche, interdisziplinäre Beschreibungssprache (näheres siehe z.B. [Cel91, Kar90]),

für Frage der Zuverlässigkeit von Systemen
- Fehlerbaumanalysen (beantworten Fragen nach kritischen Pfaden in komplexen Systemen und Wahrscheinlichkeit des ersten Systemausfalls)
- Markov-Graphen für mehrfachen Systemausfall bei reparierbaren Systemen,

die bisher verwendeten Modelle können auch als zeitdiskretisierte Parameter verwendet werden

Vierte Bewertungsstufe
In dieser Stufe ist die Konkretisierung der Gestaltungsmerkmale soweit vorangeschritten, daß nunmehr mit diskreten (in Zeit und Raum) "geometriebeherrschenden" Simulationsmodellen gearbeitet werden kann und sollte (vergl. Sch97). Allen voran, ist dieser Bereich die Domäne der Methode der finiten Elemente mit all ihren Derivaten. Ein Übergang zwischen den parametrisierten MKS-Modellen und den diskreten Modellen gelingt hier z.B. durch X-MKS-Systeme (extended Multibody Systems), welche auf der Component-Mode-Synthesis basieren (Synthetisieurng des Gesamtsystemverhaltens durch Eigenformen der Subsysteme).

5. Formalisierung in den funktionsorientierten Phasen

Bekanntlich sind die frühen Phasen für die "richtige" Entwicklung eines Produktes Alles entscheidend. Die starke Wechselwirkung zwischen gerätetechnischer und informationstheoretischer Entwicklung im Falle mechatronischer Produkte verlangt deswegen eine kompatible Formalisierung beider Aspekte. D.h. erwünscht ist eine formalisierte Simulationstheorie, welche bilateral mit den entsprechenden Algorithmen kooperieren kann. Einen allgemeinen Ansatz hierfür stellen graphentheoretische Mittel dar.

5.1. Bondgraphen als Simulationswerkzeug

Bondgraphen stellen die graphentheoretische Umsetzung einer physikalischen Metatheorie dar, welche auf Analogien zwischen verschiedenen physikalischen Disziplinen basiert [Cel91, Kar90]. Bondgraphen beruhen auf Leistungsbilanzen (entsprechend dem fundamentalen Axiom der Physik, dem Prinzip der virtuellen Leistungen oder dem 1. Hauptsatz) unter Berücksichtigung der anderen Erhaltungssätze (z.B. Impulserhaltung) und der Kompatibilität. Mit den weiteren Bausteinen, wie Ports, Transformatoren und Energiequellen können vermaschte mechanische und elektrische (im wesentlichen parametrisierte) Modellprozesse simuliert werden [Kar90].

5.2. Formalismen der Informatik

Für den Entwurf von Software werden in der Informatik Werkzeuge des *Computer Aided Software Engineering* (CASE) verwendet. Neben Hilfsmittel für Organisation und Kommunikation werden Beschreibungssprachen für Funktionen, Daten und Kontrollstrukturen angeboten.

Es gibt in der Informatik unterschiedlichste Vorgehensweisen zur Systemspezifikation. Nachdem die 80er Jahre von den funktionsorientierten Methoden wie *Structured Analysis / Structured Design* (SA/SD) und *Jackson Structured Development* (JSD) beherrscht wurden, dominieren seit ca. 1990 die objektorientierten Vorgehensweisen. Bekannteste Vertreter sind die *Object Modeling Technique* (OMT) nach [Rum91] *Object Oriented Design* (OOD) nach [Boo91] und die aus beiden resultierende *Unified Modeling Language* (UML). Eine Gemeinsamkeit dieser Methoden ist, daß sie für Steuerungsaufgaben zustandsorientierte Beschreibungen verwenden. Diese spezifizieren die (endliche) Menge aller möglichen logischen Zustände eines Programms und die Ereignisse, welche einen Zustandsübergang bewirken.

Mit einer zustandsorientierten Beschreibung des Systems, wird neben der detaillierten Festlegung des genauen Systemverhaltens auch die Verifikation (formaler Nachweis für das korrekte Verhalten des Programms) und die Validierung (Nachweis, daß das entworfene System den Entwurfszielen der Anforderungen entspricht) möglich.
Mit diesen formalen Mitteln kann man Programme auf:

- **Vollständigkeit**: Es darf nicht vorkommen, daß Ereignisse eintreten, die in diesem Zeitpunkt nicht vorgesehen sind.
- **Verklemmungsfreiheit**: Kein gegenseitiges Warten auf fremde Ergebnisse.
- **Livelockfreiheit**: Keine Aktivitäten ohne Fortschritt.
- **Terminierung**: Vom Startzustand aus muß immer ein Endzustand erreicht werden.
- **Beschränktheit:** keine nicht ausführbaren Nachrichten.

überprüfen.

Die frühsten Beispiele ereignisorientierter Entwurfssprachen stammen aus der Kommunikationstechnologie. Bekannteste Vertreter sind *System Development Language* (SDL), *Communicating Sequential Processes* (CSP) und *Petri-Netze*. Entwicklungen neuerer Zeit werden in weiten Gebieten des Softwareentwurfs eingesetzt, beispielsweise Statecharts nach [Har87], welches teilweise zur Spezifikation sicherheitskritischer Systeme im Automobilbau angewendet wird.

Sämtliche zustandsorientierte Beschreibungssprachen basieren auf dem Konzept der endlichen Automaten. Diese bestehen aus der Menge aller möglichen Zuständen, der Menge alle möglichen Ereignisse und den Transitionen (Übergänge) zwischen den Zuständen.

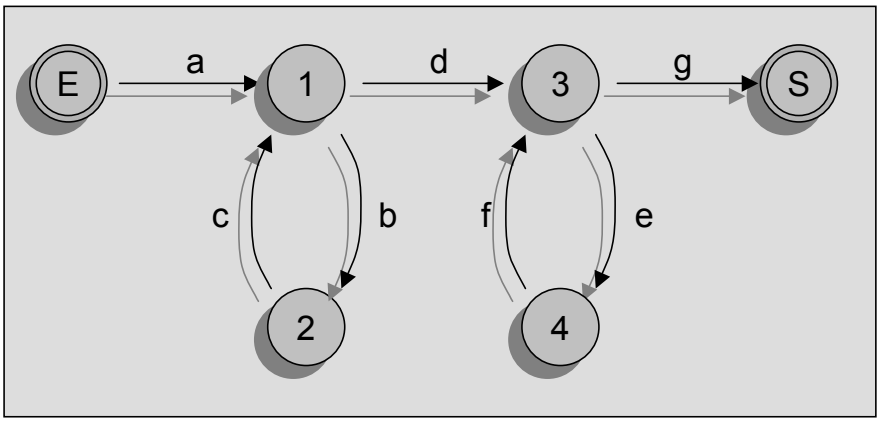
Die formale Definition eines (endlichen) Automaten lautet:

$$A = (Q, \mathrm{E}, *, Q_{-}, Q_{+})$$

Wobei:

Q:		Menge aller möglichen Zustände
E		Menge aller möglichen Ereignisse
$*$	$\subseteq Q \times \Gamma \times Q$	Menge der Transitionen $(q, \Phi \rightarrow q)$
Q_{-}	$\subseteq Q$	Menge der Anfangszustände
$Q+$	$\subseteq Q$	Menge der Endzustände

Faßt man die Ereignisse als Bestandteile eines Eingabealphabets auf, so akzeptiert der Automat eine bestimmte Eingabegrammatik. Eine solche Grammatik wird als *formale*



$Q = \{ E, S, 1, 2, 3, 4 \}$
☜ $= \{ a, b, c, d, e, f \}$
✉ $= \{ (E, a, 1), (1, b, 2), (1, d, 3), (2, c, 1), (3, e, 4), (3, g, S), (4, f, 3) \}$
$Q_- = \{ E \}$
$Q_+ = \{ S \}$

Abb. 4: Beispiel eines endlichen Automaten

Sprache beschrieben. Diese können automatisch ausgewertet und verarbeitet werden. Die Äquivalenzen zwischen Automaten und formalen Sprachen sind in der *Chomsky-Hierarchie* (vergl. [Hop79]) beschrieben. Der in Abb. 4 beschriebene Automat erkennt beispielsweise Ausdrücke der folgender Form:

$$L = \{ a\,(bc)^n\,d\,(ef)^n\,g;\ n \geq 0 \},$$

Als formale Beschreibung komplexer Ausdrücke wird in der Regel die Backus-Naur-Form [Nau60] verwendet.

Formale Beschreibungen von Automaten, bzw. von ihren Sprachen, sind für den Anwender meist unübersichtlich. Deshalb werden graphische Beschreibungsformen gewählt, welche einerseits für den Anwender leicht verständlich und durchschaubar bleiben, andererseits aber umfassend, eindeutig und formal festgelegt sind.

6. Simulation eines mechatronischen Produkts

Im vorhergehenden Kapitel wurden zustandsorientierte Beschreibungen für physikalisch-technische (PT) und für informationstechnologische (IT) Systeme vorgestellt. Während Bondgraphen zeitkontinuierliche Zustandsänderungen besitzen, dominieren bei zustandsorientierten IT-Systemen Ereignisse, welche entweder durch äußere Einflüsse oder durch Ablauf von Wartezeiten initiiert werden. Für eine Simulation des gesamten mechatronischen Produktes ist es notwendig, die beiden Systeme in einem Modell zu koppeln.

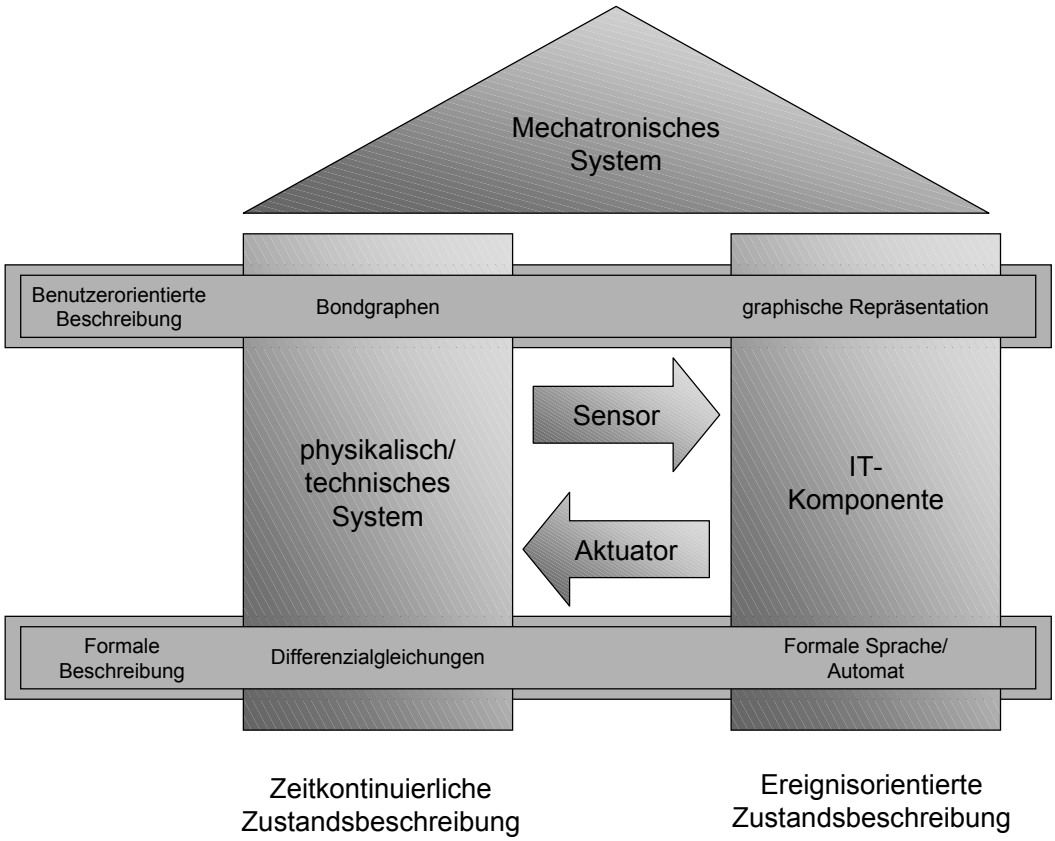


Abb. 5: Mechatronisches System

In Abb. 5 ist die Kopplung des PT-Systems und der IT-Komponente dargestellt. Realisiert werden diese Verbindungen in Form von Sensoren und Aktuatoren. Sensoren nehmen die Informationen des PT-Systems auf und setzen sie in Ereignisse der IT-Komponente um. Aufgrund dieser Ereignisse kann die IT-Komponente in einen nachfolgenden Zustand überwechseln. Nach dem Wechsel wird ein Aktuator die vom IT-System generierten Anweisungen zurück an das PT-System geben.

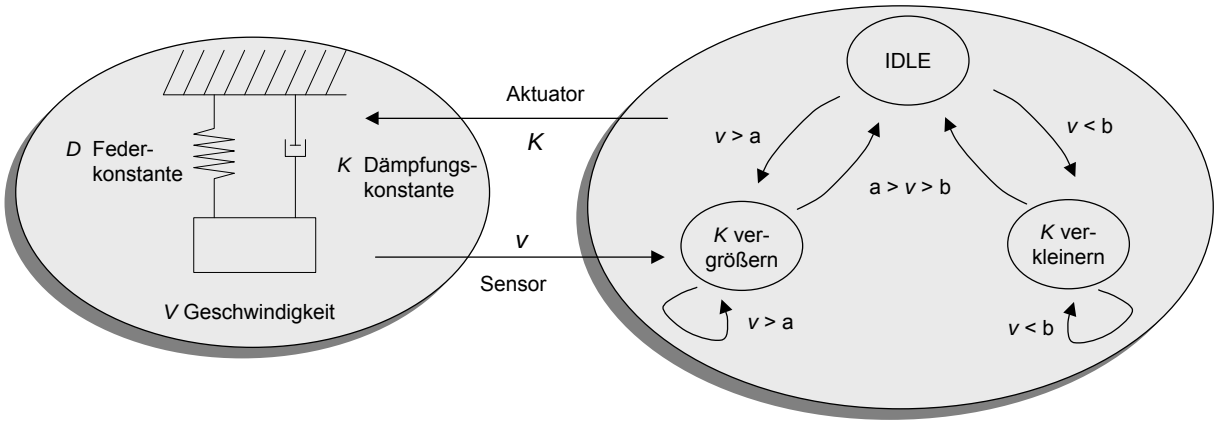
Diese Beschreibungen kann der Benutzer mit komfortablen graphischen Beschreibungsmitteln, beispielsweise mit Hilfe von Bondgraphen und Statecharts, eingegeben. Ausgehend davon generiert ein Interpreter automatisch die Differentialgleichungen des PT-Systems und den Automaten des IT-Systems.

Da der Entwurf der IT-Komponente noch keine detaillierten Informationen über deren Realisierung enthält, können die Übergangszeiten zwischen den einzelnen Zuständen nicht bestimmt werden. Es ist aber möglich Schaltzeiten anzugeben und durch Variation und anschließende Simulation des mechatronischen Systems, Höchstgrenzen für den Zustandsübergang zu bestimmen.

6.1. Beispiel

Zur Verdeutlichung dieses Konzepts wird im folgenden ein kurzes Beispiel das Zusammenwirken zwischen PT- und IT-System skizzieren. Wie in Abb. 6 zu sehen, soll ein Feder-Masse-System durch einen Algorithmus gesteuert werden. Ein Sensor überträgt die gemessene Geschwindigkeit *v* an das IT-System, welche diese als Ereignis interpretiert. Zu Beginn des Kontrollvorgangs befindet sich der Automat im Zustand „IDLE“ und wartet auf Über- bzw. Unterschreitung der Referenzgeschwindigkeiten. Bei einer Überschreitung der

Geschwindigkeit „a“ wechselt der Automat in den Zustand „*K* erhöhen“ und verbleibt dort bis das Ereignis eintritt, daß die gemessene Geschwindigkeit wieder im Intervall [a, b] liegt. Analog verhält es sich bei einer Unterschreitung der Referenzgeschwindigkeit „b“.



$$\ddot{x} + \frac{K}{m}\dot{x} + \frac{D}{m}x + g = 0$$

Q = { IDLE, K vergrößern, K verkleinern }
☜ = { *v* < b, *v* > a, a < *v* < b }
✉ = { (IDLE, *v* > a, *K* vergrößern),
(IDLE, *v* < b, *K* verkleinern),
(*K* vergrößern, a > *v* > b, IDLE),
(*K* vergrößern, *v* > a, *K* vergrößern),
(*K* verkleinern, a > *v* > b, IDLE),
(*K* verkleinern, v > a, K verkleinern) }
Q_- = { IDLE }
Q_+ = { }

Abb. 6: Beispiel eines mechatronischen Produkts

Die Beschreibung des PT-Systems kann, sofern in Form eines Bondgraphen eingegeben, automatisch in eine Bewegungsgleichung umgesetzt werden. Ebenso verhält es sich mit der Beschreibung des Automaten. Sind die Kopplungen über Sensor- und Aktuatorgrößen angegeben, so kann ein gemeinsames Modell der Form:

$$\ddot{x} + \frac{K}{m}\dot{x} + \frac{D}{m}x + g = 0 \qquad \text{wobei} \quad \begin{cases} K \text{ vergrößert wenn } v > a \\ K \text{ verkleinert wenn } v < b \end{cases}$$

generiert und numerisch simuliert werden.

7. Literatur

[Boo91] Booch, G.
Object Oriented Design
Benjamin/Cummings Publishing Company, Inc., 1991.

[Cel91] Cellier F. E.
Continuous System Modeling
Springer, 1991.

[Har87] Harel, D.
Statecharts: A visual Formalism for complex Systems
Science of Computing 8, p. 231-274.

[Hop79] Hopcroft, J.; Ullman, J.
Introduction to Automata Theory, Languages and Computation
Addison-Wesley, 1979.

[Hub84] Hubka V.
Theorie Technischer Systeme
Springer-Verlag, 1984.

[Kar90] Karnopp D. C., Margolis D. L., Rosenberg R. C.
System Dynamics: A unified Approach, 2nd Ed.
John Wiley & Sons, Inc., 1990.

[Nau60] Naur, P.
Revised Report on the Algorithmic Language ALGOL 60
Communications of the ACM, Vol. 3 No.5, pp. 299-314, May 1960.

[Pah97] Pahl G., Beitz W.
Konstruktionslehre, 4. Aufl.
Springer, 1997.

[Rum91] Rumbaugh, J.; et al
Object-Oriented Modeling and Design
Prentice-Hall, 1991.

[Sch97] Schweiger, W., Löffel, C.
Computational Methods in Design - An ordering Scheme
ICED, Tampere 97, Vol. 3, p. 91-96.

[Sol79] Solodownikow W.W., Semjonow W.W., Peschel M., Nedo D.
Berechnung von Regelsystemen auf Digitalrechnern
VEB-Verlag, Berlin, 1979.

[Wal95] Wallaschek J.
Modellierung und Simulation als Beitrag zur Verkürzung der Entwicklungszeiten mechatronischer Produkte
VDI-Berichte Nr. 1215, S. 35-50, 1995.

Autoren:

Prof. Dr.-Ing. Willy Schweiger
Lehrstuhl für Konstruktionstechnik
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Pestalozziring 20
D-91058 Erlangen
Tel.: ++49 9131 / 6199-11
Fax.: ++49 9131 / 6199-30
e-mail: schweiger@mfk.uni-erlangen.de
URL: http://www.mfk.uni-erlangen.de/personen/schweiger.html

Dipl.-Inf. Achim Schön
Lehrstuhl für Konstruktionstechnik
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Martensstraße 9
91058 Erlangen
Tel. ++ 49 9131 / 85-7984
Fax.: ++ 49 9131 / 85-7988
e-mail: schoen@mfk.uni-erlangen.de
URL: http://www.mfk.uni-erlangen.de/personen/schoen.html