Lehrstuhl für
Montagesystemtechnik und Betriebswissenschaften
der Technischen Universität München

Verteilte, kooperative Steuerung maschinennaher Abläufe

Hans Meier

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. K. Bender

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. G. Reinhart
2. Univ.-Prof. Dr.-Ing. G. Färber

Die Dissertation wurde am 19.12.2000 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 10.5.2001 angenommen.

Forschungsberichte iwb

Band 155

Hans Meier

Verteilte kooperative Steuerung maschinennaher Abläufe

herausgegeben von
Prof. Dr.-Ing. G. Reinhart

Herbert Utz Verlag UTZ

Forschungsberichte iwb

Berichte aus dem Institut für Werkzeugmaschinen
und Betriebswissenschaften
der Technischen Universität München

herausgegeben von

Univ.-Prof. Dr.-Ing. Gunther Reinhart
Technische Universität München
Institut für Werkzeugmaschinen und Betriebswissenschaften (iwb)

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist
bei Der Deutschen Bibliothek erhältlich

Zugleich: Dissertation, München, Techn. Univ., 2001

ISBN 3-8316-0044-9

Printed in Germany

Herbert Utz Verlag GmbH, München
Tel.: 089/277791-00 · Fax: 089/277791-01

Geleitwort des Herausgebers

Die Produktionstechnik ist für die Weiterentwicklung unserer Industriegesellschaft von zentraler Bedeutung. Denn die Leistungsfähigkeit eines Industriebetriebes hängt entscheidend von den eingesetzten Produktionsmitteln, den angewandten Produktionsverfahren und der eingeführten Produktionsorganisation ab. Erst das optimale Zusammenspiel von Mensch, Organisation und Technik erlaubt es, alle Potentiale für den Unternehmenserfolg auszuschöpfen.

Um in dem Spannungsfeld Komplexität, Kosten, Zeit und Qualität bestehen zu können, müssen Produktionsstrukturen ständig neu überdacht und weiterentwickelt werden. Dabei ist es notwendig, die Komplexität von Produkten, Produktionsabläufen und -systemen einerseits zu verringern und andererseits besser zu beherrschen.

Ziel der Forschungsarbeiten des *iwb* ist die ständige Verbesserung von Produktentwicklungs- und Planungssystemen, von Herstellverfahren und Produktionsanlagen. Betriebsorganisation, Produktions- und Arbeitsstrukturen sowie Systeme zur Auftragsabwicklung werden unter besonderer Berücksichtigung mitarbeiterorientierter Anforderungen entwickelt. Die dabei notwendige Steigerung des Automatisierungsgrades darf jedoch nicht zu einer Verfestigung arbeitsteiliger Strukturen führen. Fragen der optimalen Einbindung des Menschen in den Produktentstehungsprozeß spielen deshalb eine sehr wichtige Rolle.

Die im Rahmen dieser Buchreihe erscheinenden Bände stammen thematisch aus den Forschungsbereichen des *iwb*. Diese reichen von der Produktentwicklung über die Planung von Produktionssystemen hin zu den Bereichen Fertigung und Montage. Steuerung und Betrieb von Produktionssystemen, Qualitätssicherung, Verfügbarkeit und Autonomie sind Querschnittsthemen hierfür. In den *iwb*-Forschungsberichten werden neue Ergebnisse und Erkenntnisse aus der praxisnahen Forschung des *iwb* veröffentlicht. Diese Buchreihe soll dazu beitragen, den Wissenstransfer zwischen dem Hochschulbereich und dem Anwender in der Praxis zu verbessern.

Gunther Reinhart

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) der Technischen Universität München.

Herrn Prof. Dr.-Ing. Gunther Reinhart, dem Leiter dieses Instituts, gilt mein besonderer Dank für die wohlwollende Förderung und großzügige Unterstützung meiner Arbeit.

Bei Herrn Prof. Dr.-Ing. Georg Färber, dem Leiter des Lehrstuhls für Realzeit-Computersysteme der Technischen Universität München, möchte ich mich für die Übernahme des Koreferates und die aufmerksame Durchsicht der Arbeit sehr herzlich bedanken. Ebenso danke ich Herrn Prof. Dr.-Ing. Klaus Bender, dem Leiter des Lehrstuhls für Informationstechnik im Maschinenwesen der Technischen Universität München, für die Übernahme des Vorsitzes.

Darüber hinaus bedanke ich mich bei allen Mitarbeiterinnen und Mitarbeitern des Instituts sowie allen Studenten, die mich bei der Erstellung meiner Arbeit unterstützt haben, recht herzlich.

München, im Mai 2001 *Hans Meier*

1 Einleitung

1.1 Ausgangssituation

Die Automatisierung von Produktionseinrichtungen hat für produzierende Unternehmen einen hohen Stellenwert erreicht. Dies liegt einerseits in wirtschaftlichen und technologischen Notwendigkeiten und andererseits in Aspekten der Arbeitserleichterung und der Risikoverminderung für den Menschen begründet (*Anker & Wirth 1994*). *Milberg (1997)* formuliert hierzu: „Komplexität und Individualität einerseits und die Forderungen nach Produktivität und Qualität andererseits machen innovative und flexible Automatisierung und Technisierung notwendig." Dies hat sich auch auf die Anbieter von Produktionsanlagen ausgewirkt, indem die Steuerungstechnik als Differenzierungsmerkmal gegenüber den Wettbewerbern heute einen entscheidenden Faktor darstellt. Dabei kommen einerseits der realisierten Funktionalität, der Leistungsfähigkeit und der Steuerungsstruktur besondere Bedeutung zu. Andererseits ist auf Restriktionen wie den Kostenrahmen oder spezifische Anwendervorschriften zu achten und gleichzeitig mit der rasanten Weiterentwicklung der Steuerungshardware Schritt zu halten (siehe Bild 1-1).

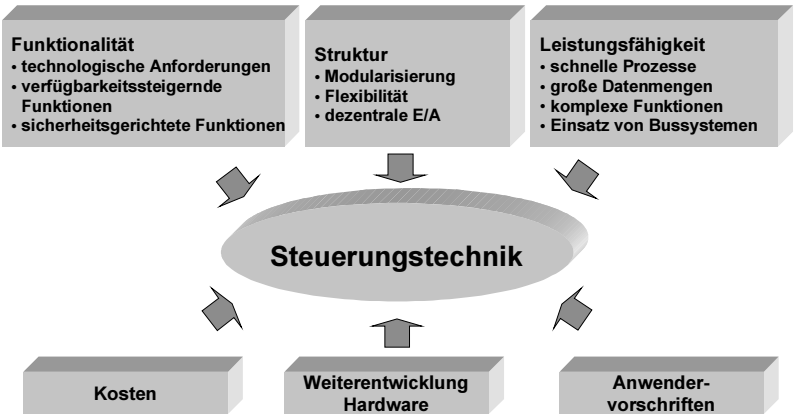

Bild 1-1: Einflüsse auf die Entwicklung von Steuerungen

Zu den vielfältigen, heute üblichen, technologisch bedingten Steuerungsfunktionen kommen inzwischen verfügbarkeitssteigernde Funktionen (Diagnose- und Fehlertoleranzfunktionen, *Reinhart u. a. 1999a*) und in jüngster Zeit auch ver-

mehrt sicherheitsgerichtete Funktionen (*Zeller 1999*) hinzu. Der Umfang der Steuerungsfunktionalität wie auch die Zahl der zu verarbeitenden Prozesssignale steigen folglich weiterhin rapide an. Dies wird insbesondere durch die Einführung diagnosefähiger Sensorik und Aktorik (*Schnell 1997*) noch verstärkt.

In Bezug auf die Leistungsfähigkeit einer Steuerung stellt die Zeit, mit der die Steuerung auf eine Änderung eines Prozesssignals zu reagieren vermag, die entscheidende Größe dar. Im allgemeinen ist für die Realisierung einer Vielzahl von Funktionen eine bestimmte Reaktionszeit sicher zu unterschreiten. In vielen Fällen übt die erreichbare Reaktionszeit auch einen Einfluss auf die Leistungsfähigkeit der gesteuerten Maschine aus, wenn diese Reaktionszeiten wie beispielsweise beim Werkzeugwechsel in einem Fräsbearbeitungszentrum sich additiv auf die Span-zu-Span-Zeit auswirken.

Die rasante Entwicklung in der Mikroelektronik unterstützt zwar einerseits die Bestrebungen, niedrigere Kosten, eine höhere Leistungsfähigkeit und eine verbesserte Modularität zu ermöglichen. Andererseits wird dadurch aber auch eine ständige Veränderung von Modulen und Strukturen der steuerungstechnischen Hardware ausgelöst, mit der in der Entwicklung der Steuerungstechnik Schritt gehalten werden muss. Zusätzlich ist es durchaus auch üblich, dass durch den Maschinenanwender konkrete Steuerungshardwaremodule oder –strukturen vorgeschrieben werden. Damit kommt der Wiederverwendbarkeit der Steuerungssoftware auch bei unterschiedlichen Hardwaremodulen und –strukturen eine große Bedeutung zu.

Ebenfalls durch die Entwicklung in der Mikroelektronik, aber auch durch das Modularitätsbestreben getrieben findet im Werkzeugmaschinenbau dezentralisierte Installationstechnik zunehmende Verbreitung (*Wagner & Bürgel 1997)*. Motivation hierfür sind Kostenvorteile, die in erster Linie durch geringere Installationskosten bei der Verkabelung und durch die standardisierungsbedingte Reduzierung von Montagefehlern entstehen (*Wagner 1995*). Darüber hinaus ergibt sich der Vorteil, dass mechanische Module solcher Produktionsanlagen, wie z. B. ein Werkzeugmagazin in einfacher Weise vollständig vorinstalliert werden können. Dadurch kann der Zeitaufwand bei der Installation und der Inbetriebnahme der Gesamtanlage erheblich reduziert werden. Eine Vorinbetriebnahme einschließlich Funktionstest von einzelnen Maschinenmodulen ist allerdings nur eingeschränkt möglich, da hierzu jeweils eine eigene, vollständige Teststeuerung programmiert werden müsste. Dementsprechend können die Module einer Gesamtanlage erst im Zuge der Gesamtinbetriebnahme getestet werden.

Nach dem derzeitigen Entwicklungsstand werden bei dezentraler Installationstechnik die im Maschinenfeld verteilten, meist binären E/A-Module (Eingangs-

/Ausgangsmodule) durch einen Feldbus an eine zentrale SPS (speicherprogrammierbare Steuerung) angekoppelt, wobei sämtliche E/A-Informationen zyklisch zu dieser SPS übertragen werden müssen. Um dabei zu akzeptablen Reaktionszeiten zu kommen, müssen die Buslaufzeiten und SPS-Zykluszeiten für den gesamten E/A-Bereich gering gehalten werden, was zu einem verhältnismäßig hohen Hardwareaufwand für eine vergleichsweise geringe Anzahl zu verarbeitender Ereignisse führt.

Die bereits erwähnten E/A-Module zum Anschluss von Sensoren und Aktoren bieten bereits eine gute Basis zur Integration von Steuerungsfunktionalität und damit zur lokalen Datenverarbeitung. Durch diese Module ist bereits ein geeignetes Gehäuse mit entsprechenden Steckbuchsen, eine Stromversorgung sowie eine Platine mit einer Protokollauswertelogik vorhanden. Eine Erweiterung um einen kostengünstigen Prozessor zur Bearbeitung von Anwendungsprogrammen liegt daher nahe und wird bereits von einigen Herstellern angeboten (siehe Abschnitt 2.5).

In verschiedenen Bereichen des Maschinenbaus, in denen steuerungstechnische Fragestellungen zu lösen sind, sind dagegen Steuerungskonzepte, die auf verteilten, kooperierenden Steuerungen basieren, bereits etabliert. Nach diesem Konzept wird die Gesamtfunktionalität des Steuerungssystems durch einen Verbund von verteilten Modulsteuerungen abgebildet. Häufig wird dabei auch die notwendige Koordination der Modulsteuerungen durch deren kooperative Arbeitsweise anstatt durch spezielle, hierarchisch übergeordnete Koordinationssteuerungen sichergestellt. Als Beispiele seien hier Steuerungssysteme für Automobile und für Flugzeuge heraus gegriffen (siehe Abschnitt 2.3).

Eine verteilte, kooperierende Steuerungstechnik bietet auch im Werkzeugmaschinenbau vielfältige Potenziale, die im folgenden kurz umrissen sind. Eine eingehendere Betrachtung findet sich in Abschnitt 3.1.

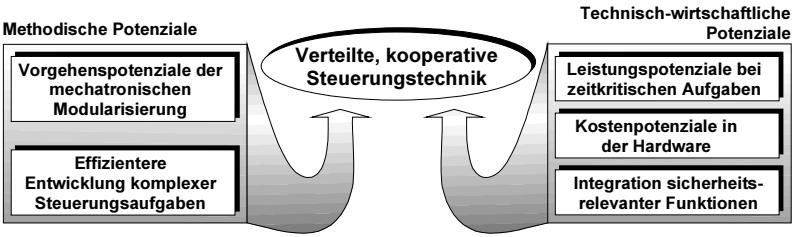

Bild 1-2: Übersicht über die Potenziale verteilter, kooperativer Steuerungstechnik im Bereich der Produktionsmaschinen.

Einerseits sind durch eine geeignete Entwicklungsmethode, die die Entwicklung der Steuerungsfunktionalität sauber von der Festlegung der Steuerungstopologie trennt, methodische Potenziale festzustellen. Ebenso bestehen Vorgehenspotenziale durch die Möglichkeit, vollständige, mechatronische Maschinenmodule zu entwickeln und so die Verflechtungen, die sich durch eine zentrale Steuerungstechnik ergeben, zu minimieren. Dabei bezeichnet der Begriff des mechatronischen Maschinenmoduls ein Modul, welches aus mechanischer, elektrischer und steuerungstechnischer Sicht ein vollständiges, in sich abgeschlossenes Teilsystem darstellt. Für die Steuerung interner Funktionen werden dabei keine externen Steuerungen benötigt. Mechatronische Module können zudem schon vor der Gesamtinbetriebnahme der Maschine getestet werden.

Andererseits bestehen durch die lokale Datenverarbeitung Leistungspotenziale bei zeitkritischen Aufgaben, die auch zur Kostensenkung in der steuerungstechnischen Hardware genutzt werden können. Zudem kann es durch die damit einher gehende Verschlankung der Zentralsteuerung ebenfalls zu Kosteneinsparungen kommen. Ein weiteres Potenzial liegt in der Integration bisher elektromechanisch realisierter, sicherheitsrelevanter Funktionen. Die Möglichkeit, beliebig Redundanz vorzusehen, stellt eine wichtige Voraussetzung für die steuerungstechnische Realisierung von Sicherheitsfunktionen dar, die bei verteilten, kooperativen Steuerungssystemen prinzipiell gegeben ist.

Verteilte, kooperative Steuerungen ist allerdings im Werkzeugmaschinenbau bisher aufgrund der Komplexität verteilter, kooperativer Funktionen nur schwer zu beherrschen. Der dazu nötige Entwicklungsaufwand kann hier anders als beispielsweise in der Automobilindustrie nicht auf hohe Stückzahlen umgelegt werden. Die im Werkzeugmaschinenbau üblichen Werkzeuge zur Steuerungsentwicklung sowie die aktuell verfügbaren Steuerungsmodule unterstützen speziell kooperative Ansätze nur sehr eingeschränkt. Daher können die oben aufgeführten Potenziale verteilter, kooperativer Steuerungen bisher im Werkzeugmaschinenbau kaum genutzt werden.

1.2 Zielsetzung und Einordnung der Arbeit

Die eingangs erläuterten Potenziale verteilter, kooperierender Steuerungskonzepte können nur mit einer auf die Anforderungen der Entwicklung maschinennaher Steuerungssoftware angepassten Entwicklungsmethodik und einer präzise darauf abgestimmten Steuerungstechnik nutzbar gemacht werden. Ziel der vorliegenden Arbeit ist es daher, eine solche Entwicklungsmethodik und eine entsprechende Steuerungstechnik zu entwickeln. Für die Entwicklungsmethodik sind folglich eine geeignete Vorgehensweise und eine anforderungsgerechte Mo-

dellierungstechnik zur Entwicklung verteilter Steuerungssoftware im Umfeld der Maschinenentwicklung zu erarbeiten. Daraus sind dann eine Steuerungsarchitektur und Konzepte für die Systemsoftware der Steuerungsmodule abzuleiten.

Zur Einordnung der Arbeit wird im folgenden der Betrachtungsbereich anhand der informationstechnischen Ebenen der rechnergeführten Produktion und anhand der bekannten Steuerungsarten genauer definiert.

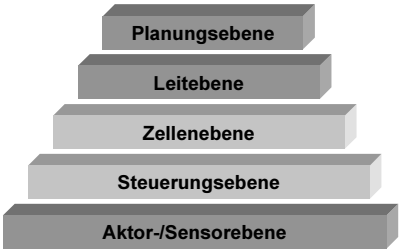

Bild 1-3: Informationstechnische Ebenen entsprechend Ottawa Report (1986)

Die rechnergeführte Produktion kann in verschiedene informationstechnische Ebenen eingeteilt werden (siehe Bild 1-3). Sowohl die Entwicklungsmethodik als auch die Steuerungstechnik sind diesem Schema entsprechend im Bereich der Steuerungsebene und in eingeschränkt der Zellenebene (im Bild hell hinterlegt) einzuordnen. Dabei ist die Steuerung einzelner Module der Steuerungsebene zuzurechnen, während Funktionen, die durch mehrere, kooperierende Module abgebildet werden, wie beispielsweise Funktionen des Werkzeughandlings oder der Auftragsdurchsetzung, auch Teil der Zellenebene sein können.

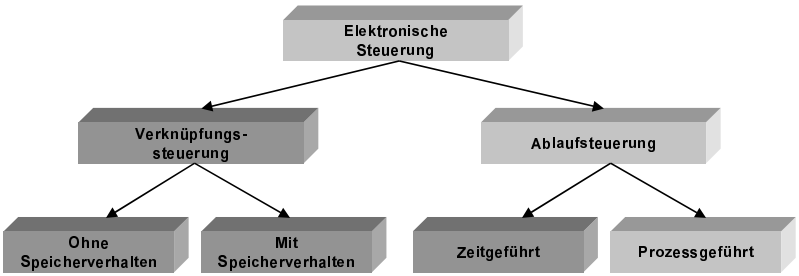

Bild 1-4: Einordnung der Arbeit anhand der Steuerungsarten nach DIN 19226-5 (1994).

Im Rahmen dieser Arbeit kommen insbesondere Funktionen der Maschinenperipherie in Betracht, da hier die Potenziale der verteilten, kooperativen Steuerungstechnik am deutlichsten zum tragen kommen. Solche Funktionen haben zumeist einen ablauforientierten Charakter. Dementsprechend betrifft die vorliegende Arbeit den Bereich der Ablaufsteuerungen. Durch die Beeinflussung des Steuerungsablaufes durch Sensorsignale kann im Sinne der *DIN 19226-5 (1994)* eine weitere Eingrenzung auf prozessgeführte Steuerungen vorgenommen werden (siehe hierzu Bild 1-4). Die NC-Technik, also die interpolierende Steuerung geregelter Achsen, wird dabei als in sich geschlossene Funktion innerhalb einer übergeordneten Ablaufsteuerung betrachtet und entsprechend hier nicht näher behandelt.

Hinsichtlich der betrachteten Maschinen liegt der Fokus dieser Arbeit auf Werkzeugmaschinen sowie auf Produktionseinrichtungen, die aus miteinander verketteten Werkzeugmaschinen bestehen. Die erarbeiteten Ergebnisse erscheinen prinzipiell auch in anderen Bereichen des Maschinenbaus einsetzbar, sofern dort zum Werkzeugmaschinenbau vergleichbare Anforderungen vorliegen und sich die geforderten Steuerungsfunktionen entsprechen. Die Übertragbarkeit wird im Rahmen dieser Arbeit aber nicht näher untersucht.

1.3 Vorgehen im Rahmen der Arbeit

Zur Erreichung der dargestellten Ziele wird im Rahmen dieser Arbeit entsprechend Bild 1-5 vorgegangen.

Zunächst wird in Kapitel 2 der Stand der Technik bezüglich des Einsatzes verteilter Steuerungstechnik im Bereich der Werkzeugmaschinen erarbeitet. Dabei wird einerseits die Situation im Werkzeugmaschinenbau eingehend beleuchtet und andererseits ein Überblick über den Einsatz in anderen Anwendungsbereichen des Maschinenbaus gegeben. Zudem wird die Einbettung der Steuerungsentwicklung in den Entwicklungsprozess untersucht. Vor dem Hintergrund eines möglichen Einsatzes im Bereich der Werkzeugmaschinen wird darüber hinaus der durch die vorliegende Arbeit abgedeckte Forschungsbedarf begründet.

Kapitel 3 dient der detaillierten Analyse der Potenziale verteilter, kooperativer Steuerungstechnik sowie der Ermittlung der Anforderungen an die Entwicklungsmethode und die Steuerungstechnik. Bezüglich der Anforderungen steht die Erschließung der analysierten Potenziale und ein wirtschaftlicher Einsatz verteilter, kooperativer Steuerungen im Mittelpunkt.

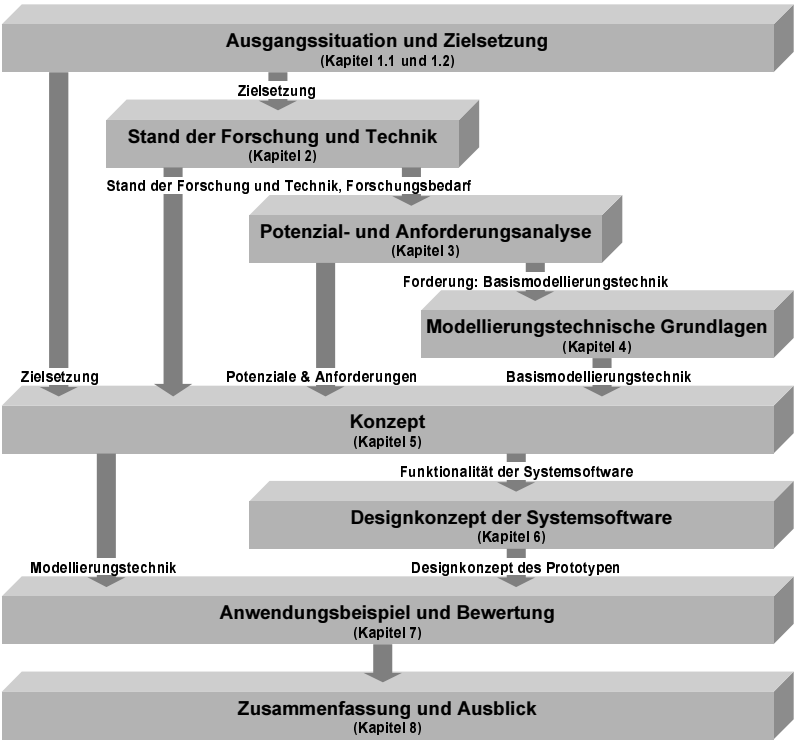

Bild 1-5: Vorgehen im Rahmen der Arbeit

In Kapitel 4 werden modellierungstechnischer Grundlagen, die zur Entwicklung des Konzepts in Kapitel 5 benötigt werden, gezielt erarbeitet. Dabei wird bereits eine Auswahl einer Modellierungstechnik als Basismodellierungstechnik für die Modellierung maschinennaher Abläufe durchgeführt.

Das Konzept für die verteilte, kooperative Steuerung maschinennaher Abläufe wird dann in Kapitel 5 vorgestellt. Dabei wird zunächst ein Grundkonzeption für den Einsatz verteilter, kooperativer Steuerungstechnik zur Steuerung maschinennaher Abläufe entwickelt. Ausgehend von einem Konzept für die Vorgehensweise bei der Entwicklung solcher Steuerungen erfolgt die Erarbeitung einer darauf abgestimmte Modellierungstechnik. Aufbauend darauf wird unter Berücksichtigung technischer Anforderungen ein Konzept für die Systemsoftware solcher Steuerungssysteme entwickelt. Darüber hinaus wird ein einfaches Verfahren zur Verteilung der Steuerungssoftware vorgestellt.

Darüber hinaus wird in Kapitel 6 ein Designkonzept für die Systemsoftware verteilter, kooperativer Steuerungen erarbeitet. Dieses Designkonzept dient einerseits als Basis für die prototypische Realisierung solcher Steuerungsmodule im Rahmen dieser Arbeit, kann aber andererseits auch als Leitlinie für eine kommerzielle Entwicklung verwendet werden.

Die praktische Einsetzbarkeit der entwickelten Methode sowie die prinzipielle Funktionsfähigkeit der technischen Konzepte werden in Kapitel 7 anhand einer beispielhaften Lösung einer Steuerungsaufgabe in einem Anwendungsbeispiel auf Basis einer prototypischen Realisierung des konzipierten Steuerungssystems nachgewiesen. Basierend auf den Erfahrungen bei der beispielhaften Lösung einer Steuerungsaufgabe wird dann eine Bewertung der vorgeschlagenen Konzepte verteilter, kooperierender Steuerungssysteme durchgeführt.

Kapitel 8 fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf künftige Arbeitsfelder.

2 Stand der Forschung und Technik

Das vorliegende Kapitel dient einerseits zur Erarbeitung des Stands der Technik bezüglich des Einsatzes verteilter Steuerungstechnik und andererseits zur Ableitung des Forschungsbedarfs für diese Arbeit. Dazu wird zunächst in Abschnitt 2.1 kurz auf das Vorgehen in der Entwicklung von Werkzeugmaschinen eingegangen, um später eine entsprechende Einordnung der Vorgehensschritte bei der Entwicklung verteilter, kooperativer Steuerungen zu ermöglichen.

Als weitere Basis werden in Abschnitt 2.2 grundlegende Steuerungsarchitekturkonzepte klassifiziert und dabei der Begriff „verteilte, kooperative Steuerungstechnik“ im Sinne dieser Arbeit präzisiert.

In Abschnitt 2.3 wird ein Überblick über den Einsatz verteilter, kooperativer Steuerungstechnik außerhalb des Werkzeugmaschinenbaus gegeben. Dem werden in Abschnitt 2.3.3 Ansätze dieser Technik im Werkzeugmaschinenbau gegenübergestellt. Abschnitt 2.5 stellt derzeit kommerziell verfügbare Elektronikkomponenten und Steuerungsmodule zusammen, die zum Aufbau verteilter, kooperativer Steuerungen von Werkzeugmaschinen eingesetzt werden können.

In Abschnitt 2.6 werden die Ergebnisse der Abschnitte 2.1 bis 2.5 kurz zusammengefasst und daraus der dieser Arbeit zugrundeliegende Forschungsbedarf abgeleitet.

2.1 Vorgehen in der Entwicklung von Werkzeugmaschinensteuerungen

Die grundsätzliche Vorgehensweise in der Projektierung automatisierter Anlagen, die die Entwicklung der Steuerungstechnik mit einschließt, wird in idealisierter Form in der Literatur (siehe z. B. *Lenschow 1991*) entsprechend Bild 2-1 beschrieben. Sie ist in Teilen auch in der Richtlinie *VDI 3683 (1986)* entsprechend standardisiert. In der Praxis allerdings zeigt sich, dass diese Vorgehensweise so nur bedingt eingesetzt wird (z. B. *Osmers 1998*, S.9). Dies liegt insbesondere darin begründet, dass zwischen der Mechanik- und der Softwareentwicklung intensive Wechselwirkungen bestehen, die häufig in der Phase der Projektierung/Konzeption (siehe Bild 2-1) noch nicht ausreichend berücksichtigt werden. Daher kann in der Praxis die strikte Trennung zwischen der Entwicklung der Steuerungstechnik und der Mechanik nicht aufrecht erhalten werden, was häufig zu einem erheblichen Softwarekorrekturaufwand in der Inbetriebnahmephase führt (*Reinhart u. a. 1998*). Als mögliche Gegenmaßnahme bietet sich hier der Übergang zu mechatronischen Maschinenmodulen mit bereits vollständig

getesteter Steuerungshard- und Software und damit einer verteilten, kooperativen Steuerungstechnik an (vgl. Abschnitt 3.1.1) (siehe hierzu auch *Reinhart u. a. 1999b*).

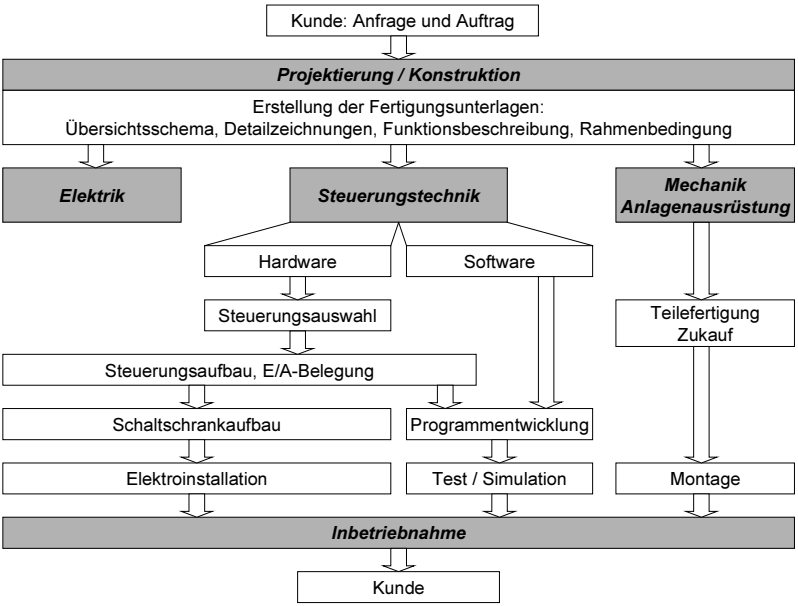

Bild 2-1: Vorgehensweise zur Entwicklung von automatisierter Anlagen nach einer auf VDI 3683 (1986) basierenden Detaillierung von Lenschow (1991).

Aus Bild 2-1 wird darüber hinaus noch deutlich, dass die Steuerungshardware bereits vor der Programmentwicklung festgelegt werden muss. Unter der bisher gültigen Prämisse einer zentralen Steuerungsarchitektur mag dies vertretbar sein, solange ausgeschlossen ist, dass die entstandene Steuerungssoftware später auf einer anderen, nicht kompatiblen Steuerung wiederverwendet werden soll.

Bei einer verteilten, kooperativen Steuerungstechnik aber kann entsprechend die Entwicklung der Steuerungsfunktionen erst nach der Festlegung der von der Modularisierung der Maschine abhängigen Steuerungstopologie erfolgen. Eine vollständige Steuerungsprogrammentwicklung ohne Kenntnis der späteren Verteilung ist nur sehr eingeschränkt möglich. Folglich stellt die bisher notwendige Sequenzialisierung dieser Vorgehensschritte ein Defizit hinsichtlich der Einsetz-

barkeit verteilter, kooperativer Steuerungskonzepte dar, da sie der Forderung nach schnellen Entwicklungsprozessen entgegen steht.

2.2 Grundlegende Architekturkonzepte

2.2.1 Zentrale Steuerung mit dezentraler Ein-/Ausgabe

In weiten Bereichen des Maschinenbaus herrscht bezüglich der für diese Arbeit relevanten Ablaufsteuerungen eine zentrale Steuerungsarchitektur mit dezentraler Ein- und Ausgabe entsprechend Bild 2-2 vor (siehe auch *Wagner & Bürgel 1997*). Dabei wird eine zentrale SPS mittels eines Feld- oder Ein-/Ausgangsbussystems und entsprechender Ein-/Ausgangsmodule mit der Sensorik und Aktorik verbunden. Hierfür kommen zum Beispiel Profibus (Process Field Bus, siehe *Bender 1990* bzw. *DIN EN 50170-2 1996*), Interbus S (*DIN EN 50254 1998, Baginski & Müller 1998*), CAN (Controller Area Network, siehe *Etschberger 1994*), AS-I (Aktuator/Sensor Interface, *Kriesel & Madelung 1999*) in Frage. *Färber 1994* gibt einen Überblick über die Charakteristika dieser und weiterer entsprechender Bussysteme, näheres findet sich auch in *Schnell (1999)*.

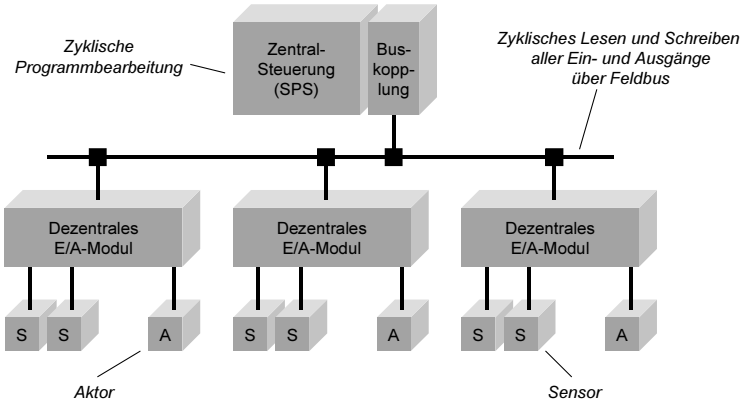

Bild 2-2: Zentrale Steuerung mit dezentraler Ein- und Ausgabe

Gegenüber einer Steuerungstechnik mit zentraler Ein-/Ausgabe liegen dabei die Vorteile insbesondere in Einsparungen sowohl hinsichtlich der Kabelkosten als auch hinsichtlich der Montageaufwendungen. Weitere Vorteile liegen in einer

vereinfachten Vormontage bereits installierter Module und in einer einfacheren Elektrodokumentation. (*Wagner & Bürgel 1997*).

Um diese Vorteile nutzen zu können, müssen die damit ausgerüsteten Maschinen eine Komplexität innerhalb bestimmter Grenzen aufweisen. Einfachste Werkstattmaschinen ohne bedeutende Peripheriefunktionalität und mit entsprechend wenigen Sensoren und Aktoren rechtfertigen einerseits nicht den Einsatz eines Bussystems. Statt dessen wird in diesen Fällen eine herkömmliche, zentrale Verdrahtung eingesetzt. Andererseits kommen komplexere Maschinen wie Transferstraßen nicht mit einer einzigen zentralen Steuerung aus. In diesem Fall wird meist auf ein Konzept entsprechend Abschnitt 2.2.2 übergegangen. Folglich kommen für das Konzept der zentralen Steuerung mit dezentraler Ein- und Ausgabe heute insbesondere Maschinen mittlerer Komplexität wie beispielsweise Fräsbearbeitungszentren in Frage (vgl. *VDW 1997a*).

2.2.2 Hierarchisch organisierter Steuerungsverbund

Ein ausschließlich hierarchisch organisierter Steuerungsverbund besteht aus zwei oder mehreren Ebenen von Steuerungsmodulen, wobei jedes Modul gegenüber Modulen einer niedrigeren Ebene als Client und gegenüber Modulen einer höheren Ebene als Server fungiert (siehe Bild 2-3).

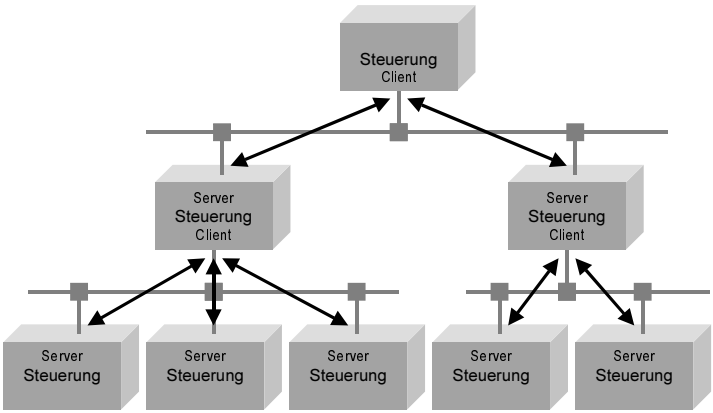

Bild 2-3: Ausschließlich hierarchisch organisierter Steuerungsverbund

Zwischen den Modulen einer Ebene bestehen dabei keine direkten Kommunikationsbeziehungen. Müssen Module einer Ebene koordiniert werden, so geschieht

dies grundsätzlich durch ein gemeinsames Clientmodul. Die zu koordinierenden Module müssen zur Abbildung der Interaktion eine entsprechend fein detaillierte Funktionsschnittstelle als Server anbieten, um dem koordinierenden Modul alle benötigten Synchronisationen zu ermöglichen. In Bild 2-4 sind dies die Funktionen A1 bis A3 des Steuerungsmoduls A und die Funktionen B1 und B2 des Steuerungsmoduls B. Am Beispiel einer Werkzeugwechselfunktion eines Fräsbearbeitungszentrums würde dies heißen, dass zum Entnehmen eines Werkzeugs aus der Spindel in den Steuerungsmodulen Werkzeugwechsler (A) und Bearbeitungseinheit (B) sich ergänzende Teilschritte (z. B. Lösen des Werkzeugs in der Spindel und Schließen des Werkzeuggreifers) nötig sind. Die Funktion zum Entnehmen eines Werkzeugs durch den Werkzeugwechsler wird dabei von einem koordinierenden Modul (K) auf der nächst höheren Ebene angeboten.

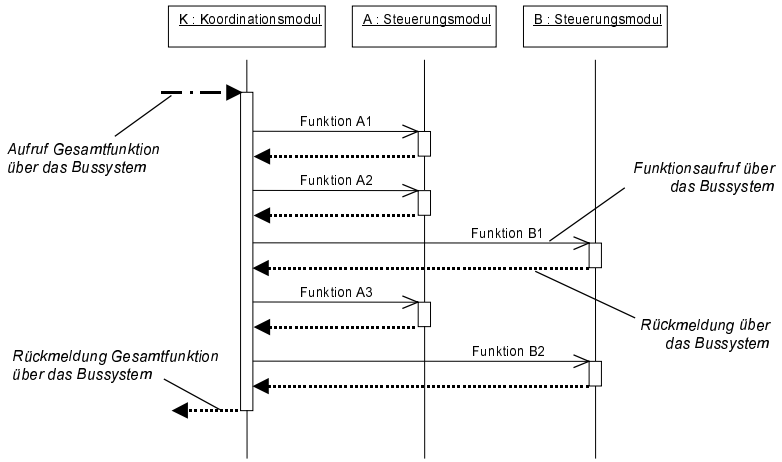

Bild 2-4: Typisches Sequenz-Diagramm bei einem ausschließlich hierarchisch organisierten Steuerungsverbund. Die verwendete Diagrammform ist an die Unified Modelling Language (Booch u. a. 1998) angelehnt (siehe auch Abschnitt 4.2.2).

Ausschließlich hierarchisch organisierte Steuerungsverbünde trifft man häufig bei Transferstraßen in Form der Kopfsteuerung an (z. B. *Waldner 1996*). Ihr Vorteil liegt in der einfachen Beherrschbarkeit bei Anwendungsfällen mit geringen Abhängigkeiten zwischen Modulen einer Ebene. Zudem lässt sich der Datenaustausch zwischen je zwei Ebenen sehr leicht und übersichtlich mit Bussystemen realisieren, da die angeschlossenen Module entweder als reiner Client oder als reiner Server fungieren.

Schwächen zeigt dieses Konzept bei Maschinen, bei denen zwischen einzelnen Modulen ein umfangreicher und eventuell sogar zeitkritischer Koordinations- und Synchronisationsbedarf besteht, da dann eine große Anzahl von Funktionen über das Bussystem zur Verfügung gestellt und angesteuert werden muss. Zusätzlich müssen zur Koordination notwendige Signale immer über den Umweg der koordinierenden Steuerung übertragen werden. Dies erhöht die Gesamtbelastung des Bussystems und setzt die mit dem Steuerungssystem erreichbare Reaktionsgeschwindigkeit herab.

2.2.3 Hierarchisch organisierter Steuerungsverbund mit direkter Synchronisation

Durch die Ergänzung um die Möglichkeit der direkten Synchronisation können die Schwächen des Konzepts nach Abschnitt 2.2.2 abgemildert werden. Funktionen eines Moduls können benötigte Informationen von anderen Modulen selbst einholen oder Ereignisse direkt dorthin weitermelden (Bild 2-5). Ein Beispiel für einen derartigen Ansatz findet sich in *Koch 1996*.

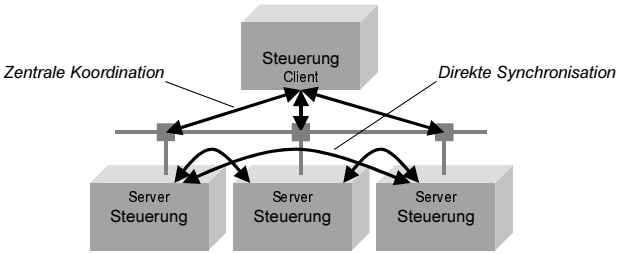

Bild 2-5: Hierarchisch organisierter Steuerungsverbund mit direkter Synchronisation.

Durch die Verlagerung von Synchronisationen in die dem Koordinationsmodul unterlagerten Steuerungsmodule kann mit einer geringeren Anzahl von Funktionen mit größerer Mächtigkeit gearbeitet werden, wie in Bild 2-6 an der zusammen gefassten Funktion A2/B1/A3 zu erkennen ist. Die Echtzeitproblematik kann so ebenfalls entschärft werden, da bei zeitkritischen Signalen der Umweg über das koordinierende Modul entfallen kann.

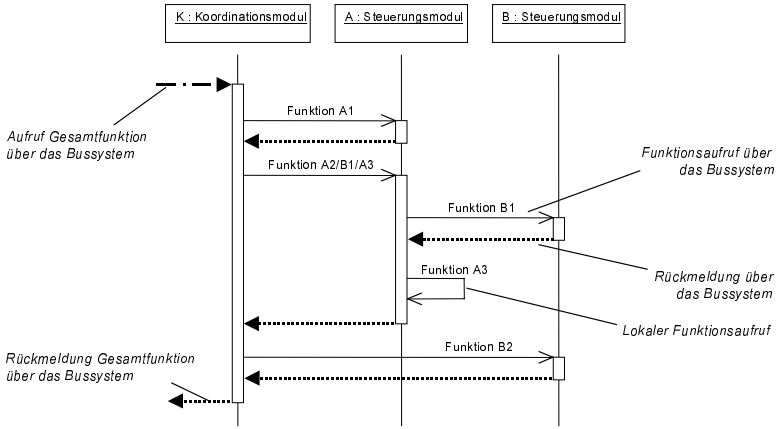

Bild 2-6: Typisches Sequenz-Diagramm bei einem hierarchisch organisierten Steuerungsverbund mit direkter Synchronisation. Vgl. hierzu Bild 2-4.

Problematisch ist bei diesem Konzept allerdings die Wahrung der Beherrschbarkeit und Übersichtlichkeit bei zunehmender Quervernetzung der Steuerungsmodule durch die Zunahme der Schnittstellen. Darüber hinaus wird in Abhängigkeit von der Koordinationsfunktionalität der Funktionsumfang der Servermodule erweitert, da diese nun Teile der Koordinationsfunktion übernehmen, was zunächst dem Gedanken einer sauberen Modularisierung widerspricht. Dieser Widerspruch ist ohne eine auf diese Technik zugeschnittene Methodik sowie entsprechende Entwicklungswerkzeuge nicht zu überbrücken.

2.2.4 Ausschließlich kooperativer Steuerungsverbund und Definition des verteilten, kooperativen Steuerungssystems

Gegenüber dem kombinierten Ansatz entsprechend Abschnitt 2.2.3 sind bei ausschließlich kooperativen Steuerungsverbünden prinzipiell keine speziellen Koordinationsmodule notwendig (siehe Bild 2-7).

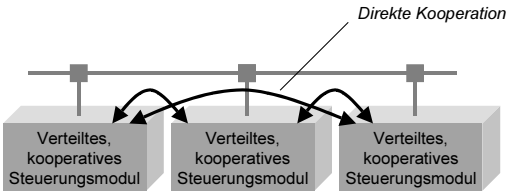

Bild 2-7: Ausschließlich kooperativer Steuerungsverbund.

Die Steuerungsaufgaben werden durch die beteiligten Steuerungsmodule auf Basis direkter Kooperation durchgeführt. Dabei ist der Kommunikationsaufwand minimal, wie in Bild 2-8 im Vergleich zu Bild 2-4 und Bild 2-6 mit nur fünf gegenüber zwölf bzw. zehn Kommunikationsschritten zu erkennen ist. Dadurch ist die Reaktionsgeschwindigkeit des Systems prinzipbedingt höher, als bei hierarchischen Steuerungsverbünden mit oder ohne direkter Synchronisation.

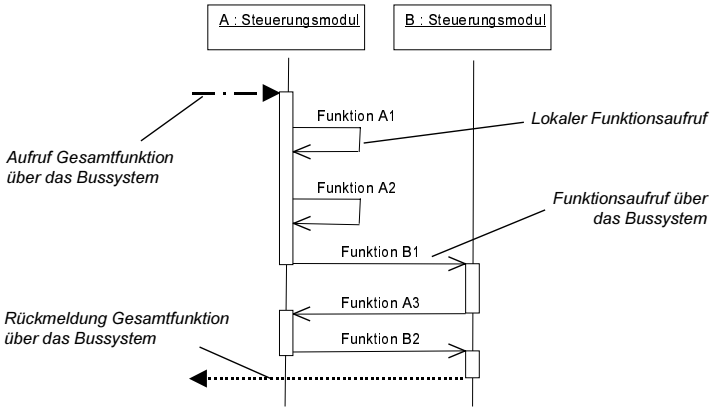

Bild 2-8: Sequenz-Diagramm bei einem ausschließlich kooperativen Steuerungsverbund.

Im Rahmen dieser Arbeit wird ein solcher **Steuerungsverbund ohne spezielle Koordinationsmodule und mit ausschließlich direkter Kooperation als verteiltes, kooperatives Steuerungssystem definiert** und soll als Grundlage der weiteren Arbeit dienen. Die darin zum Einsatz kommenden Steuerungsmodule werden auch als verteilte, kooperative Steuerungen bezeichnet.

Die bereits in Abschnitt 2.2.3 angesprochene Problematik hinsichtlich der Beherrschbarkeit, der Übersichtlichkeit sowie einer sauberen Modularisierung bestehen bei verteilten, kooperativen Steuerungssystemen selbstverständlich in zunächst noch ausgeprägterem Umfang. Sie steht damit der Nutzung der vielfältigen Potenziale dieses Architekturkonzepts im Wege. Daher stellt es einen zentralen Aspekt dieser Arbeit dar, hierfür Lösungsansätze zu entwickeln.

2.3 Einsatz verteilter, kooperativer oder hybrider Steuerungstechnik außerhalb des Werkzeugmaschinenbaus

Außerhalb des Werkzeugmaschinenbaus sind eine Reihe von Einsatzfällen verteilter, kooperativer Steuerungskonzepte bekannt, die hier diskutiert werden sollen. Damit können im weiteren Verlauf der Arbeit einerseits die Frage der Übertragbarkeit solcher Ansätze auf den Werkzeugmaschinenbau geklärt werden. Andererseits dienen diese bestehenden Erfahrungen in anderen Bereichen des Maschinenbaus als Referenz für die Analyse der Potenziale verteilten, kooperativen Steuerungen in Abschnitt 3.1. Dabei wurden gezielt einige Einsatzbereiche exemplarisch ausgewählt, die hinsichtlich der Anforderungen an den Entwicklungsprozess und an die Steuerungsfunktionalität Ähnlichkeiten zum Werkzeugmaschinenbau aufweisen. Als ähnlich betrachtet werden dabei insbesondere Einsatzbereiche, bei denen

- mechatronische Module hauptsächlich nach mechanischen und elektrotechnischen, jedoch nur untergeordnet nach softwaretechnischen Gesichtspunkten gebildet werden,
- auf einige diskrete Prozesssignale und Ereignisse innerhalb einer definierten, maximalen Zeit reagiert werden muss und
- zur Funktionserfüllung sowohl zustandsorientierte, reaktive Funktionen als auch ablauforientierte, aktive Funktionen gleichermaßen benötigt werden.

Als Beispiele werden im Folgenden Steuerungssysteme für Automobile und für Flugzeuge vorgestellt. Weitere Beispiele verteilter, kooperativer oder hybrider Konzepte finden sich unter anderem in der Kraftwerksleittechnik (z. B. *Meisel 1996*) oder in der Fördertechnik (z. B. *Gausemeier u. a. 1998*). Abschließend werden in Abschnitt 2.3.3 exemplarisch einige Forschungsansätze vorgestellt.

2.3.1 Verteilte, kooperative Steuerungen bei Automobilen

Die Steuerungstechnik von Automobilen wird von verteilten, kooperativen Ansätzen dominiert. Das Multimaster-Protokoll des ursprünglich für den Automobilsektor entwickelten Bussystems CAN (Controller Area Network, *Etschberger 1994*) unterstützt speziell auf kooperativer Verteilung basierende Steuerungsphilosophien. Zentrale Steuerungseinheiten mit hierarchisch untergeordneten, ohne Steuerungsfunktionalität versehene Busanschaltungen sind dagegen nicht üblich. Statt dessen finden sich normalerweise komplex vernetzte Verbünde von gleichwertigen Modulen z. B. aus dem Bereich des Antriebsstrangmanagements (*Bitzenberger & Schmidt 1997, Specks 1997*). Die Komponenten bilden dabei möglichst viel Funktionalität selbständig ab. Allerdings ist zur Beherrschung solcher komplex vernetzter Systeme deren systematischer Aufbau und eine geeignete Werkzeugunterstützung unverzichtbar. Entsprechend beschreiben beispielsweise *Bertram u. a. (1997*) auf den Automobilbau zugeschnittene Ansätze einer solchen Entwicklungsmethodik.

Diese Steuerungsphilosophie unterstützt die Gegebenheiten im Automobilbau, da hier für die Steuerung einzelner Aggregate (z. B. Motorsteuerung) extreme Echtzeitanforderungen vorliegen. Durch die Implementierung der Rechenleistung direkt am zu steuernden Aggregat ohne Sensor- und Aktordatenübertragung über ein Bussystem fallen dabei Buslaufzeiten weg und es kann dementsprechend mit Prozessoren wesentlich geringerer Leistungsfähigkeit gearbeitet werden, als dies bei einer zentralen Steuerung möglich wäre.

Der Datenaustausch zwischen den Aggregaten unterliegt dagegen aufgrund der durch die lokale Vorverarbeitung eintretenden Informationsverdichtung in der Regel deutlich geringeren Anforderungen, da der Umfang der zu übertragenden Daten wie auch die Echtzeitanforderungen wesentlich niedriger sind. Deshalb kann, abgesehen von einzelnen sicherheitsrelevanten Systemen wie z. B. ABS, auch auf entsprechend leistungsfähige und damit teuere Kommunikationssysteme verzichtet werden.

Ein weiterer Vorteil des verteilten Steuerungskonzepts ist die bessere Orientierung der Aggregate an einer mechatronischen Modularisierung. Dadurch kann die Steuerungssoftware eines Aggregats zusammen mit dem Aggregat selbst entwickelt und gepflegt werden (*Strauss 1997*). Über die so entstehenden Aggregatschnittstellen hinaus sind somit keine Detailinformationen über Interna des Aggregats zu dessen Ansteuerung von außen nötig. Damit kann die Entwicklung einzelner Aggregate besser voneinander entkoppelt werden. Zusätzlich werden dadurch Outsourcing- und Zulieferstrategien unterstützt. Dies geht soweit, dass komplette, mit ihrer Steuerungssoftware vorgetestete Aggregate geliefert werden,

die in der Montage nur noch an das Fahrzeugbussystem angeschlossen werden müssen.

2.3.2 Verteilte, kooperative Steuerungen bei Flugzeugen

Bei modernen Verkehrsflugzeugen wird heute intensiv von Rechnern zur Flugsteuerung und Fluglageregelung Gebrauch gemacht. Dabei kommt die „Fly-By-Wire"-Technologie (siehe z. B. *Aplin 1997, Fischer 1993*) zum Einsatz, was bedeutet, dass die Signale zur Ruderpositionierung über Bussysteme vom Flugsteuerungsrechner zum Aktor übertragen werden, anstatt dies mittels mechanischer Seilzüge zu bewerkstelligen.

Bei solchen Steuerungssystemen sind zwei Motivationen zum Einsatz verteilter Steuerungen zu erkennen. Zum einen spielen auch hier vollständig eigenständige, mechatronische Module mit allen Vorteilen für den Entwicklungs- und Produktionsprozess wie auch die Vorteile der lokalen Verarbeitung von Information eine entscheidende Rolle. Ein Beispiel eines solchen mechatronischen Moduls ist ein elektrohydraulischer Aktor zur Ruderpositionierung für die „Fly-By-Wire"-Technologie (*Ivantysynova 1996*). Zum anderen führen bei Flugzeugen insbesondere die aus Sicherheitsgründen bis zu fünffachen (teils auch diversitär ausgeführten) Redundanzen einzelner Komponenten zwangsläufig zu verteilten, kooperativen Steuerungssystemen (*Günzel 1995*).

2.3.3 Ansätze in der Forschung

Bereits seit längerem sind Konzepte zur Programmierung beziehungsweise Modellierung parallelisierter oder verteilter Rechnerarchitekturen aus der Informatik bekannt. Diese basieren meist auf speziellen Programmiersprachen wie beispielsweise Concurrent-C (*Tsujino u. a. 1984*), MRP (Mehrrechner-PEARL) (*DIN 66253 1988*) oder aber auf grafischen Modellierungstechniken (siehe unten).

Eine Reihe dieser Konzepte bezieht sich dabei auch auf Fragestellungen aus der Automatisierungstechnik, deren besonderes Merkmal die physikalisch bedingte Verteiltheit der Prozessschnittstellen darstellt. *Weber* (*1990, S. 7*) schreibt hierzu: „Es ist stets zu beachten, dass der originäre Grund für die Verteilung des Steuerungsprogramms nicht ein sicherlich ebenfalls erwünschter Lastausgleich unter den beteiligten Automatisierungsrechnern ist, sondern die durch die Konfiguration vorgegebene räumliche Verteilung der einzulesenden bzw. auszugebenden Prozessvariablen." Dabei wurde auch die Notwendigkeit einer monolithischen Programmierung – d. h. eine Programmierung ohne Berücksichtigung einer kon-

kreten Softwareverteilung – herausgestellt. In *Weber* (*1990)* werden hierzu beispielsweise auf der Programmiersprache C (*Kernighan & Ritchie 1978*) aufbauende Konzepte einschließlich eines automatisierten Softwarelokalisierungsverfahrens sogenannter Programmatome vorgestellt.

Andere Ansätze mit Bezug zur Automatisierungstechnik gehen ebenfalls von verteilten Systemen aus, wie beispielsweise die in *Broy u. a. (1999* – Werkzeug AUTOFOCUS), *Selic u. a. (1994,* Modellierungstechnik der Methode ROOM) oder *Harel (1990* – Kommerzielle Umsetzung im Werkzeug STATEMATE) eingesetzten, grafischen Modellierungstechniken. Zu diesen Modellierungstechniken sind jeweils Steuerungsplattformen vorhanden, die ein verteiltes Abarbeiten der mit der Modellierungstechnik erstellten Softwareelemente in verschiedenen Prozessen, gegebenenfalls auf verschiedenen Steuerungsmodulen, ermöglichen.

Dabei lässt sich jedoch feststellen, dass in diesen Ansätzen mit dem Einsatz von zustandsgrafenorientierten Techniken ein verhaltensorientierter Ansatz dominiert. Dies kommt allerdings der im Werkzeugmaschinenbau verbreiteten, ablauforientierten Denkweise nicht entgegen. Zudem ist es mit diesen Techniken erforderlich, für mögliche Auftrennungen von Funktionalitäten zum Zwecke der Verteilung auf verschiedene Steuerungsmodule entsprechende Schnittstellen vorzusehen. Diese sind aber aus einer ausschließlich an der Funktionalität orientierten Softwaremodularisierung nicht zu rechtfertigen, widersprechen also dem Gedanken der monolithischen Programmierung. Letztere Feststellung gilt auch für andere, nicht mit Zustandsgrafen arbeitende Techniken wie z. B. IPANEMA (*Honekamp 1998*).

Im Forschungsprojekt DeKOS (Dezentrale, kooperierende, offene Systeme, *Bregulla u. a. 2000*) werden für eine Vielzahl intelligenter Feldgeräte diverse Funktionen standardisiert. Dabei liegt der Fokus allerdings auf Bedien- und Parametrierfunktionen. Eine Technik zur Nutzung solcher Feldgeräte als frei programmierbare Steuerungsmodule im Sinne dieser Arbeit ist dabei nicht vorgesehen.

2.4 Ansätze einer verteilten, kooperativen Steuerungstechnik im Werkzeugmaschinenbau

Nachdem im vorher gehenden Abschnitt der Einsatz verteilter, kooperativer Steuerungstechnik außerhalb des Werkzeugmaschinenbaus beleuchtet wurde, soll dem hier deren Einsatz bei Werkzeugmaschinen gegenüber gestellt werden. Dabei wird auf die Ergebnisse einer vom *iwb* durchgeführten Befragung bei einer Reihe von namhaften Werkzeugmaschinenherstellern im Rahmen des VDW-Projekts „Dezentralisierte Steuerungstechnik“ zurückgegriffen (*VDW 2000*).

2.4.1 Verteilte Steuerungen bei Zuliefermodulen mit einfachsten Schnittstellenanforderungen

In der vom iwb durchgeführten Befragung konnten vielfach Zuliefermodule gefunden werden, die über eine eigene Steuerung - üblicherweise auf SPS-Basis - verfügen. Diese Zuliefermodule stellen zumeist in sich weitgehend autark arbeitende, mechatronische Module dar, deren Kommunikationsbedarf zur Maschinensteuerung sich in der Regel auf wenige Bits beschränkt. Beispiele hierfür sind Kühl-/Schmierstoff-Kombinationen, Ladeportale oder Palettenspeicher. In diesen Fällen wurde die Schnittstelle als binäre Parallelschnittstelle – entsprechend Bild 2-9 - oder als ein entsprechendes, bustechnisches Äquivalent gebildet. Prinzipiell kann ein solches Konzept als eine erste Form der verteilten, kooperativen Steuerung angesehen werden.

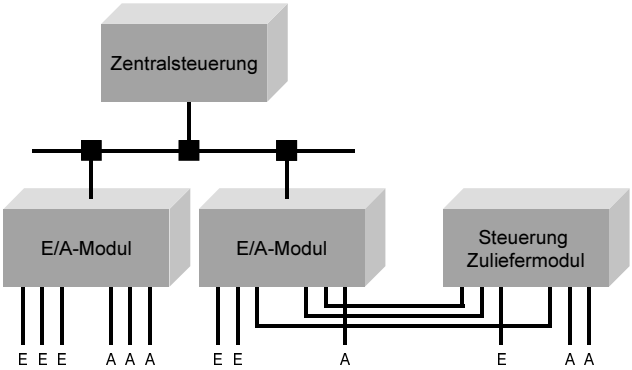

Bild 2-9: Über eine binäre Parallelschnittstelle angebundenes, einfaches Steuerungsmodul.

Die nach diesem Konzept gesteuerten Module verfügen meist über eine integrierte Bedienerschnittstelle einfachster Bauart, z. B. ein Operator Panel mit einer ein- oder mehrzeiligen LCD-Anzeige, über die hauptsächlich Zustands- und Diagnosemeldungen angezeigt werden. Die Diagnoseschnittstelle zur Maschinensteuerung ist in diesen Fällen auf eine einzige Sammelfehlermeldung reduziert, die dann am Bedienfeld der Maschinensteuerung angezeigt wird. Diese Fehlermeldung verweist dann den Bediener auf die Diagnoseanzeige am fehlerbehafteten Modul. Ein Durchgriff von der Maschinensteuerung über die Schnittstelle in die Modulsteuerung in der Inbetriebnahme ist ebenfalls nicht möglich.

Eine derartige Lösung kann dementsprechend nur bei einfachen Modulen, die nur geringe steuerungstechnische Wechselwirkungen mit der Maschinensteuerung aufweisen, deren Schnittstellen kaum Änderungen unterworfen sind und die einen untergeordneten Einfluss auf die Gesamtverfügbarkeit der Maschine haben, in Betracht kommen.

2.4.2 Verteilte Steuerungen bei komplexen, verketteten Werkzeugmaschinen

Bei komplexen, verketteten Werkzeugmaschinen wie z. B. Transferstraßen oder insbesondere Transferpressen sind bereits heute mehrere, teilweise sehr leistungsfähige Steuerungen im Einsatz, wie die Befragung im Kreise der VDW-Mitgliedsfirmen ergab (*VDW 2000*). Dabei können komplexe Bustopologien – entsprechend Bild 2-10 - mit bis zu ca. 12 Bussträngen entstehen. Dies wird nötig, um bei der Vielzahl an E/A-Punkten noch akzeptable Bus- und Steuerungszykluszeiten zu erreichen.

Neben der Hardware- und Kostenproblematik stellt auch die Handhabung eines solchen Steuerungssystems in der Entwicklung und Inbetriebnahme ein Problem dar.

Einerseits muss die Funktionalität eines solchen Steuerungssystems hardwareorientiert modularisiert werden, da an einer Steuerungsmodulgrenze grundsätzlich zu Kommunikationszwecken eine Schnittstelle programmiert und damit eine funktionale Schnittstelle eingeführt werden muss (Bild 2-11). Bei einer Vielzahl von Funktionen (z. B. Schmierung bei Transferpressen) ist eine solche Schnittstelle aus funktionalen Gründen aber nicht zu rechtfertigen. Eine hardwareunabhängige, rein funktionsorientierte Strukturierung der Software ist also nicht möglich.

Diese Problematik ist bei immer wiederkehrenden Steuerungsaufgaben insofern heute beherrschbar, da über die Jahre hinweg normalerweise eine Hard- und Softwarestrukturierung erarbeitet wurde, die sowohl aus Hardwaregesichtspunkten, als auch aus Gesichtspunkten der funktionalen Strukturierung der Software die gestellten Anforderungen zu erfüllen vermag. Jede Änderung dieser Struktur in der Hard- oder Software stellt allerdings einen immensen Aufwand dar.

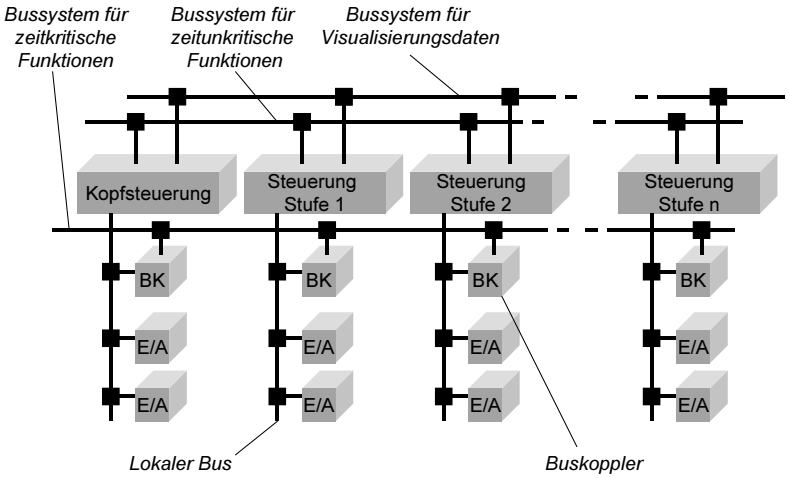

Bild 2-10: Steuerungssystem bei komplexen Werkzeugmaschinen am Beispiel einer Transferpresse (vereinfacht).

Andererseits ist es in der Inbetriebnahme eines solchen Systems unbedingt erforderlich, dass das Steuerungssystem von jedem beliebigen Punkt aus einen transparenten Durchgriff auf die Daten, Programme etc. anderer Steuerungen zulässt. Dies ist heute noch nicht in allen Fällen gegeben und wird bei zunehmender Dezentralisierung von Steuerungsfunktionen verstärkt Probleme aufwerfen.

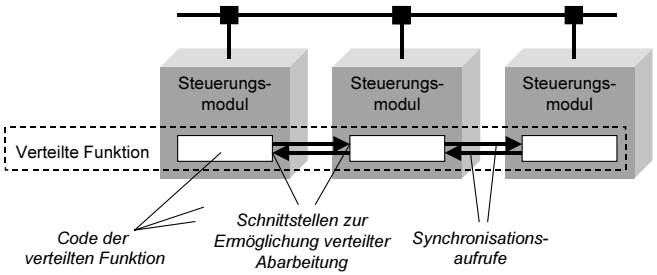

Bild 2-11: Zu Kommunikationszwecken eingeführte, funktionale Schnittstellen

2.4.3 Ansätze in der Forschung

Hinsichtlich der aktuellen Forschungsansätze zur verteilten, kooperativen beziehungsweise hybriden Werkzeugmaschinensteuerungen werden hier exemplarisch die Ergebnisse der Forschungsprojekte OSACA und HÜMNOS herangezogen. Sie repräsentieren die Entwicklung solcher sogenannter offener Steuerungen für den Bereich der Werkzeugmaschinen in idealer Weise. Andere Arbeiten (z. B. *OMAC 1994*) führen in etwa zu vergleichbaren Aussagen.

Die Kommunikationsplattform OSACA (Bild 2-12) ermöglicht es, auf einfache Weise Steuerungen zu modularisieren und diese Module auch auf mehrere Hardwareplattformen zu verteilen (vgl. *Pritschow & Sperling 1997*, *Daniel 1996, Weck u. a. 1993, Pritschow u. a. 1996*). Betrachtet man dabei ausschließlich die Sicht auf die beteiligte Hardware, so kann dabei durchaus ein kooperatives Steuerungssystem entstehen. Die Sicht auf die Struktur der Architekturobjekte zeigt aber, dass bedingt durch das Konzept von OSACA üblicherweise streng hierarchische Verbünde von Architekturobjekten vorliegen, da in der Regel zwischen Architekturobjekten eindeutige, einfache Client-/Serverbeziehungen bestehen. Entsprechend werden in HÜMNOS (*Schäfer 1997, HÜMNOS 1998*) solche hierarchisch übergeordneten Architekturobjekte häufig Koordinatoren genannt. Ringstrukturen von Architekturobjekten sind dagegen nicht üblich.

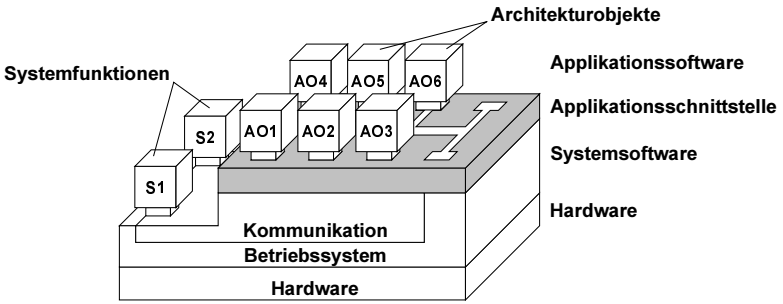

Bild 2-12: Steuerungsplattform OSACA (nach Daniel 1996)

Prinzipiell wird durch die Kommunikationsplattform OSACA auch in der Sicht auf die Struktur der Architekturobjekte eine verteilte, kooperative Steuerungstechnik ermöglicht. Dazu muss auf der Ebene der Architekturobjekte auf die eindeutigen Client-/Serverbeziehungen verzichtet werden. Statt dessen können durch eine beiderseitige Festlegung von Serverkommunikationsobjekten zwi-

schen Architekturobjekten echte, kooperative Beziehungen aufgebaut werden (Bild 2-13).

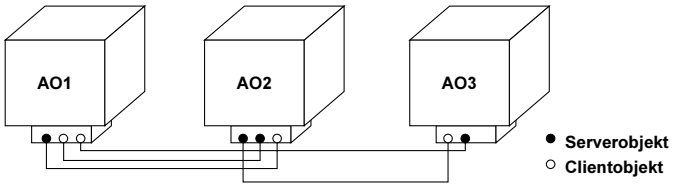

Bild 2-13: Beispiel für eine kooperative Beziehung zwischen drei OSACA-Architekturobjekten durch eine beiderseitige Festlegung von Serverkommunikationsobjekten.

Allerdings führt dieses Vorgehen sehr leicht zu einem Verlust der Beherrschbarkeit des Steuerungssystems. Einerseits bedarf es zur Programmierung solcher Architekturobjekte intensiver Kenntnisse der Programmierung von Multitaskinganwendungen auf Basis von C++ (zu C++ siehe *Stroustrup 1997*). Andererseits reicht die in OSACA übliche Beschreibung der Architekturobjekte über Templates für Serverkommunikationsobjekte nicht mehr aus, um ihr Verhalten sicher zu beschreiben. Letzteres kann dazu führen, dass bestimmte Konfigurationen solcher Architekturobjekte zu Verklemmungen des Steuerungssystems führen, ohne dass dies aus den Beschreibungen der Architekturobjekte zu erkennen wäre.

Das Projekt HÜMNOS hat für wichtige Applikationsmodule von Werkzeugmaschinensteuerungen, insbesondere aus dem Bereich der Zellenfunktionalität (siehe hierzu *Groha 1988* und *Glas 1993*) wie z. B. Werkzeugmanagement, Diagnose und MDE/BDE standardisierte Strukturen, Funktionsbereiche und Modulschnittstellen geschaffen (*HÜMNOS 1998*). Diese Module sind aber darauf angewiesen, dass ihnen weitere, maschinennähere Steuerungsmodule zur Verfügung stehen. Die Schnittstellen dieser Module auf Applikationsseite konnten auf der Ebene der Serverkommunikationsobjekte allgemeingültig festgelegt werden. Deren Modulstruktur und die Schnittstelle zu der durch den jeweiligen Funktionsbereich betroffenen Hardware kann aufgrund der vielfältigen mechanischen Lösungen jedoch nicht mit vertretbarem Aufwand vollständig vereinheitlicht werden.

2.5 Kommerziell verfügbare Lösungen zum Aufbau verteilter, kooperativer Steuerungen

Die Firma Siemens bietet mit dem Gerät ET200X BM147/CPU (*Siemens 1999b*) eine Steuerungshardwarekomponente an, die sich prinzipiell zum Aufbau verteilter, kooperativer Steuerungssysteme eignet. Es handelt sich hierbei um eine Kleinsteuerung, die in ein flexibles, feldfähiges E/A-Modul integriert ist. Es wurde bereits angekündigt, dass hierfür die Funktionalität des "Datenquerverkehrs" eingeführt wird. Dadurch wird die direkte Kommunikation solcher Steuerungen untereinander ermöglicht und damit eine wesentliche, technische Voraussetzung für den Aufbau verteilter, kooperierender Steuerungssysteme erfüllt.

Andere Anbieter von SPSen und Bussystemen wie beispielsweise Phoenix Contact oder Groupe Schneider gehen ebenfalls den Weg einer zunehmend dezentraleren Steuerungsfunktionalität. Phoenix Contact bietet beispielsweise Chips an, mit denen sehr einfach Feldgeräte mit eigenem, frei programmierbaren Prozessor aufgebaut werden können (*Jasperneite 1997*). Groupe Schneider bietet auf Feldbusstationen aufsteckbare Klein-SPSen an, die innerhalb eines Netzes miteinander kommunizieren können (*Groupe Schneider 1998*).

Das Local Operating Network (LON) unterstützt Netzwerke von Komponenten mit lokaler Steuerungsfunktionalität. Dabei basiert LON auf einer Chip-Familie der Firma Echelon, die einerseits das Protokollhandling durchführt und andererseits freie Rechenkapazität für die Implementierung von Anwendungsprogrammen (C-Programmierung) zur Verfügung stellt. Zusätzlich sind LON-Chips mit einer vielfältig parametrierbaren E/A-Schnittstelle ausgerüstet. LON-Netze sind prinzipiell als Multi-Master-Systeme aufgebaut. Damit bietet LON alle wesentlichen Voraussetzungen zum Aufbau verteilter, kooperativer Steuerungssysteme (s. *Dietrich u. a. 1999*). Spezielle, für den Einsatz in Maschinen entwickelte LON-Module werden bereits angeboten, die aber bislang wenig Verbreitung gefunden haben. Es sind aber Einsatzfälle wie z. B. bei Lackieranlagen (*Herkommer 1999*) bekannt.

Vergleichbare, integrierte Chips (Protokollverarbeitung und frei programmierbare Anwendung) mit verhältnismäßig hoher Leistungsfähigkeit sind auch für CAN zu sehr günstigen Preisen (unter 10 Dollar) erhältlich (*Gigou 1998*). Auch von CAN-Modulen sind Einsatzfälle mit lokaler Steuerungsfunktionalität im Maschinenbau bekannt (z. B. *Leindl 1997*).

Auch die Firma Jetter bietet eine Steuerungshardware an, die auf Basis geswitchter Ethernet/TCP/IP-Netze (Transmission Control Protocol / Internet Protocol) aufsetzen (*Jetter 1999*). Dabei kann mit mehreren Steuerungsmodulen grundsätzlich auch eine verteilte, kooperative Steuerung im weiteren Sinne auf-

gebaut werden. Allerdings ist nach diesem Konzept ein Steuerungsmodul lediglich ein Steuerungsrechner, der keine eigenen Ein- und Ausgänge besitzt. Letztere werden über das Bussystem und E/A-Module, die keine eigene Steuerungsfunktionalität abbilden können, an den technischen Prozess angeschlossen. Es kann dabei also nicht von einem verteilten, kooperativen Steuerungssystem im Sinne dieser Arbeit gesprochen werden, da die Vorteile der schnellen, lokalen Reaktionen, die bei direkt am Steuerungsmodul angeschlossener Sensorik und Aktorik möglich sind, nicht genutzt werden können.

2.6 Zusammenfassung und Ableitung des Forschungsbedarfs

Zusammenfassend kann festgestellt werden, dass auf Verteilung und Kooperation basierende Steuerungskonzepte sich in verschiedenen Bereichen des Maschinenbaus wie z. B. im Automobilbau bereits durchgesetzt haben. Auch im Bereich des Werkzeugmaschinenbaus ist ein Trend zur Dezentralisierung erkennbar. Dies wird einerseits durch die inzwischen weit verbreitete, dezentrale Installationstechnik (siehe Abschnitt 1.1) und andererseits durch die Arbeiten zu offenen Steuerungsarchitekturen belegt. Getragen wird dieser Trend durch die rasante Entwicklung in der Mikroelektronik, die heute eine Vielzahl von Schaltkreisen zum Aufbau solcher Steuerungssysteme kostengünstig anbietet. Erste darauf basierende Steuerungsmodule, die grundsätzlich einen Einsatz im Werkzeugmaschinenbau erlauben, sind bereits am Markt verfügbar.

Im Werkzeugmaschinenbau können bereits Ansätze einer verteilten, kooperativen Steuerungstechnik festgestellt werden. Meist handelt es sich dabei aber um spezifische, zum Teil sehr einfache Verknüpfungen zwischen Modulen auf der Ebene binärer Signale. Vereinzelt werden komplexe Steuerungsaufgaben auch heute schon mit verteilten Softwarestrukturen gelöst. In beiden Fällen wird aber keine allgemeingültige Methodik zur Entwicklung solcher Funktionen angewendet, sondern auf starre Schnittstellenstrukturen sowie in arbeitsintensiver Weise auf das Know-how erfahrener Programmierer zurückgegriffen.

Der Verbreitung verteilter, kooperativer Techniken im Werkzeugmaschinenbau steht aber die Frage der Beherrschbarkeit in der Entwicklung, der Inbetriebnahme und im Betrieb entgegen. Insbesondere muss sich dabei die Steuerungsentwicklung in die Rahmenbedingungen der Werkzeugmaschinenentwicklung einordnen. Dies hat zur Folge, dass sich einerseits der Aufwand der Steuerungsentwicklung nicht auf hohe Stückzahlen umlegen lässt, wie dies beispielsweise im Automobilbau möglich ist, und andererseits die verfügbare Zeit zur Umsetzung der aus der Mechanikentwicklung stammenden Funktionsanforderungen extrem knapp bemessen ist.

In der aktuellen, zentral geprägten Steuerungstechnik wird der Problematik des Zeitdrucks und des Aufwands in der Steuerungsentwicklung begegnet, indem mit einer spezialisierten Steuerungshardware, meist einer SPS, und darauf abgestimmten Entwicklungswerkzeugen gearbeitet wird. Diese Werkzeuge bieten problemorientierte Modellierungstechniken an, während die zugehörige Steuerung bereits eine Vielzahl von unterstützenden Funktionen zur Bearbeitung von Steuerungsaufgaben mitbringt.

Eine vergleichbare, steuerungstechnische Systemlandschaft für verteilte, kooperative Steuerungen existiert derzeit aber noch nicht. Soll mit der bestehenden Technik eine verteilte, kooperative Steuerungslösung aufgebaut werden, muss bereits zu einem frühen Zeitpunkt die genaue Steuerungs- und Verteilungsstruktur festgelegt werden. Die Bedienung der damit festgelegten Kommunikationsschnittstellen muss bei der anschließend erfolgenden Programmierung der Funktionen im gesamten Code berücksichtigt werden. Damit entstehen einerseits extrem starre Strukturen, andererseits ist die Kommunikationsprogrammierung aufwendig und fehlerträchtig.

Zusätzlich variiert die geforderte Maschinenfunktionalität von Kunde zu Kunde stark, was zu immer neuen Modulkombinationen führt. Daher besteht in einem verteilten System häufig Anlass, die Softwareverteilung zu modifizieren, was jedoch bei heutigen Techniken nicht sinnvoll handhabbar ist.

Um die Potenziale verteilter, kooperativer Steuerungskonzepte im Werkzeugmaschinenbau besser nutzen zu können, ist es daher naheliegend, Methoden und Werkzeuge zu entwickeln, die auf Basis darauf abgestimmter Steuerungsmodule und –funktionen die Entwicklung verteilter, kooperativer Maschinensteuerungen unterstützen. Eine solche steuerungstechnische Systemlandschaft muss über den bei der zentralen Steuerungstechnik bereits bekannten Leistungsumfang hinaus dem Entwickler einen effizienten Umgang mit der Verteilung ermöglichen. Dies bedeutet insbesondere, dass der Steuerungsentwickler einerseits von der aufwendigen und fehlerträchtigen Aufgabe der Kommunikationsprogrammierung entlastet und andererseits eine flexible Modularisierung unterstützt werden muss.

Der dieser Arbeit zugrunde liegende Forschungsbedarf ist es daher, ausgehend von einer genaueren Analyse der Potenziale verteilter, kooperativer Steuerungstechnik und der Anforderungen für deren Einsatz im Werkzeugmaschinenbau ein Konzept für eine derartige, steuerungstechnische Systemlandschaft zu erarbeiten. Entsprechende Steuerungsmodule einschließlich der zur verteilten, kooperativen Abarbeitung benötigten Systemsoftware sind darauf aufbauend prototypisch zu realisieren. Die Eignung dieses Konzepts ist darüber hinaus anhand eines Anwendungsbeispiels nachzuweisen und zu bewerten. Darüber hinaus ist für die

Steuerungsmodule eine Systemsoftware zu entwickeln, die die Abarbeitung der modellierten Steuerungssoftware unterstützt

Die Entwicklung von Modellierungswerkzeugen für die vorzuschlagenden Modellierungstechniken ist dagegen aus wissenschaftlicher Sicht nicht notwendig und daher nicht Inhalt dieser Arbeit. Die Modellierung des Anwendungsbeispiels kann jederzeit auch ohne Werkzeugunterstützung erfolgen. Auf der Grundlage der hier zu erarbeitenden Konzepte können die benötigten Werkzeuge im Rahmen einer Produktentwicklung später jederzeit implementiert werden.

3 Potenzial- und Anforderungsanalyse

Ziel dieser Arbeit ist es, die Potenziale einer verteilten, kooperativen Steuerungstechnik für den Werkzeugmaschinenbau nutzbar zu machen. Daher werden im vorliegenden Kapitel zunächst die Potenziale einer solchen Technik (vgl. Bild 1-2) eingehender analysiert (Abschnitt 3.1). Dem werden in Abschnitt 3.2 Anforderungen an die Steuerungstechnik im Werkzeugmaschinenbau gegenübergestellt, die in Zusammenhang zu einem Übergang auf eine verteilte, kooperative Steuerungstechnik stehen. Abschnitt 3.2.4 fasst die durch den Einsatz verteilter, kooperativer Steuerungstechnik nutzbaren Potenziale sowie die dabei zu berücksichtigenden Anforderungen kurz zusammen. Dabei wird zudem auf die durch diese Arbeit zu erschließenden Potenziale fokussiert.

3.1 Potenziale der verteilten, kooperativen Steuerungstechnik

Bei der Analyse der Potenziale einer verteilten, kooperativen Steuerungstechnik muss der Betrachtungsbereich über die direkt bewertbaren, technisch-wirtschaftlichen Potenziale hinaus noch weiter gefasst werden, da nur so ein umfassendes Bild entstehen kann. Dabei sind insbesondere die Aspekte einer effizienten Entwicklung von Werkzeugmaschinen, einer zügigen und aufwandsarmen Inbetriebnahme und einer guten Beherrschbarkeit im Betrieb zu betrachten. Hinzu kommen Aspekte, die die Wandlungsfähigkeit von Werkzeugmaschinenherstellern und –anwendern stärken. Hierunter ist beispielsweise die Fähigkeit eines Werkzeugmaschinenherstellers zur kurzfristigen Veränderung seiner Zulieferstrategie oder eines Werkzeugmaschinenanwenders zur Umkonfiguration seiner Fertigungssysteme zu verstehen. Die hiermit in Zusammenhang stehenden Potenziale können in Potenziale der mechatronischen Modularisierung (Abschnitt 3.1.1) und in Potenziale bei der Umsetzung komplexer Steuerungsaufgaben (Abschnitt 3.1.2) untergliedert werden.

Die technisch-wirtschaftlichen Potenziale umfassen Leistungspotenziale bei zeitkritischen Anwendungen (Abschnitt 3.1.3), Kostenpotenziale in der Hardware (Abschnitt 3.1.4) und Potenziale bei der Integration sicherheitsrelevanter Funktionen (Abschnitt 3.1.5).

3.1.1 Potenziale der mechatronischen Modularisierung

Durch eine verteilte Steuerungstechnik wird eine mechatronische Modularisierung der Maschinen – d. h. die Modularisierung in vollständige, Mechanik, Elektrik, Elektronik und Software umfassende Maschinenmodule - möglich. Da-

durch können viele Module im Zusammenspiel mit der ihnen zugeordneten Steuerung schon vor einer Gesamtinbetriebnahme und mit einer einfachen Teststeuerung zur Abbildung des Verhaltens angrenzender Module einem **vollständigen Modultest** unterzogen werden. So können Fehler - z. B. aufgrund einer nicht zur Mechanik passenden Softwareversion - schon früher bzw. sicherer entdeckt und die Gesamtinbetriebnahmezeiten verkürzt werden.

Die Bildung vollständiger, mechatronischer Module unterstützt darüber hinaus die Wandlungsfähigkeit von Zulieferstrategien, da die Software, die in der Modulsteuerung bereits enthalten ist, zweifelsfrei dem Verantwortungsbereich des Zulieferers zuzurechnen ist. Somit wird ein **einfacheres Outsourcing** möglich. Der Systemintegrator wird von der Zuordnung verschiedener Softwareversionen zu den jeweiligen Mechanikvarianten entlastet. Dem Zulieferer können dadurch größere Freiheiten in der Optimierung der Module eingeräumt werden. Der Systemintegrator hat dann nur noch die koordinierenden Anteile der Steuerungssoftware dem Modul hinzuzufügen.

Darüber hinaus kann durch eine derartige Modularisierung die **Austauschbarkeit von Modulen** bei Werkzeugmaschinen – wie er in der Automobilindustrie bereits vereinzelt angedacht wird – unterstützt werden. Ein solcher Austausch kann bei identischen mechanischen und softwaretechnischen Schnittstellen verhältnismäßig einfach und selbst dann realisiert werden, wenn neues und altes Modul intern nicht identisch sind.

3.1.2 Potenziale bei der Umsetzung komplexer Steuerungsaufgaben

Komplexe Steuerungsaufgaben können auch heute nicht ausschließlich zentral oder hierarchisch gelöst werden. In diesen Fällen wird zumeist mit aufwendigen Schnittstellenkonzepten ein kooperatives Konzept aufgebaut (siehe auch Abschnitt 2.4.2). Zur Betrachtung der Potenziale bei der Umsetzung komplexer Steuerungsaufgaben soll hier davon ausgegangen werden, dass sowohl eine Methode als auch eine entsprechende Werkzeugunterstützung im Sinne dieser Arbeit für die Entwicklung verteilter, kooperativer Steuerungssysteme vorliegt. Müssen dabei die angesprochenen Kommunikationsschnittstellen nicht mehr durch den Steuerungsentwickler programmiert werden, sondern werden aufgrund von Steuerungstopologieinformationen durch ein Entwicklungswerkzeug eingefügt, führt dies zu einer **vereinfachten Programmierung**.

Eine Entwicklungsmethode, die eine Unabhängigkeit der funktionalen von der hardwareorientierten Strukturierung der Steuerungssoftware bietet, ermöglicht eine **frühzeitige** und detaillierte **Softwareentwicklung** schon bevor eine konkrete Steuerungstopologie festgelegt ist. Damit kann die Softwareentwicklung im

Sinne eines Simultaneous Engineering früher beginnen und weitgehend zur Mechanik- und zur Elektrokonstruktion parallelisiert werden. Dadurch kann eine Verkürzung von Projektlaufzeiten erreicht werden.

Ein weiteres Kostensenkungspotenzial ergibt sich, wenn durch ein schnelles und sicheres Vorgehen auch schon bei heutigen Problemstellungen für eine verteilte, kooperative Steuerungstechnik eine **zielsichere Dimensionierung** der Leistungsfähigkeit der Elemente des Steuerungssystems möglich wird.

3.1.3 Leistungspotenziale bei zeitkritischen Aufgaben

Die Verarbeitung von Prozesssignalen auf einer lokalen, parallel an die Sensorik und Aktorik angeschlossenen Kleinsteuerung – integriert in ein dezentrales E/A-Modul – bewirkt den Wegfall von Kommunikationszeiten zu einer zentralen Steuerung. Zusätzlich muss diese Kleinsteuerung nur einen geringen Funktionsumfang realisieren. Dadurch können mit einem solchen System bei einem niedrigen Hardwareaufwand mit **schnellen, lokalen Reaktionen** deutlich bessere Reaktionszeiten erreicht werden, als dies bei einer zentralen Steuerung mit dezentraler E/A-Anbindung möglich wäre.

Durch die lokale Verarbeitung der Sensor- und Aktorinformationen wird eine drastische Reduzierung der über das Kommunikationssystem zu transportierenden Daten erzielt. Durch diese **Entlastung des Kommunikationssystems** können auch bei Steuerungsaufgaben, an denen mehrere solcher verteilter Steuerungen beteiligt sind, bessere Reaktionszeiten erreicht werden.

Eine **direkte Kooperation** zwischen verteilten Steuerungen ermöglicht dabei nochmals verbesserte Reaktionszeiten, da nur ein einziger Kommunikationsschritt zur Datenübertragung nötig ist (s. Abschnitt 2.2.4).

3.1.4 Kostenpotenziale in der Hardware

Durch die Ausschöpfung der oben beschriebenen Leistungspotenziale kann bei gleichgebliebenem Leistungsbedarf die Steuerungsaufgabe mit einer kostengünstigen Hardwareausstattung – Low-Cost-Prozessoren auf den E/A-Modulen – realisiert werden. Durch die Verlagerung von Funktionalität in verteilte Steuerungen verringern sich der Leistungsbedarf und damit die Kosten einer zentralen Steuerung drastisch. Der dort zu realisierende Funktionsumfang kann einerseits weitestgehend oder gar völlig auf die verteilten Kleinsteuerungen ausgelagert werden und hat andererseits wesentlich verringerte Reaktionszeitanforderungen zu erfüllen. Dabei kann dies zu einem **Wegfall der Zentralsteuerung** führen

oder es kann zumindest der verbleibende Funktionsumfang auf eine vorhandene Nicht-Echtzeit-Plattform wie z. B. einen Bedienfeld-PC verlagert werden.

Darüber hinaus kann aufgrund des geringeren Datenvolumens und der geringeren Reaktionszeitanforderungen der Kommunikation auf **günstigere Bussysteme** umgestiegen werden. Dies gilt insbesondere bei komplexen Steuerungsaufgaben, wenn hier die Zahl der verwendeten Busstränge und –koppler verringert werden kann.

Weitere Potenziale bestehen durch die fortschreitende Miniaturisierung und die damit verbesserte Unterbringungsmöglichkeit von Steuerungsmodulen im Maschinenfeld. Durch eine entsprechend konsequente **Feldinstallation** der Module kann ein weiterer Schritt hin zur "schaltschranklosen Maschine" gegangen werden, wodurch sich weitere Einsparungseffekte ergeben.

3.1.5 Integration von sicherheitsrelevanten Funktionen

Die Integration der Sicherheitstechnik in die Maschinensteuerung beinhaltet ein großes Kosteneinsparungspotenzial gegenüber der bisher üblichen, parallel verdrahteten, elektromechanischen Sicherheitstechnik, die einen hohen Installations- und Materialaufwand verursacht. Zudem ermöglicht der direkte Zugriff auf Signale und Zustände der Sicherheitsfunktionen eine vereinfachte Diagnose. Darüber hinaus können umfangreichere Sicherheitsfunktionen, die Maschinen- und Betriebszustände berücksichtigen und entsprechend mit situationsangepassten Schutzfunktionen arbeiten, realisiert werden (*Zeller 1999*).

Soll dies mit heutigen Mitteln erreicht werden, kommen meist spezielle Sicherheitssteuerungen, die intern redundant aufgebaut sind und zur Sicherstellung der Funktion zahlreiche Kreuzvergleichs-Operationen durchführen, zum Einsatz (z. B. *Thomas 1999*) Solche Steuerungen sind allerdings in der Regel sehr teuer, weshalb ihr Einsatz nur unter speziellen Umständen gerechtfertigt ist, wenn beispielsweise sehr komplexe Sicherheitsfunktionen zu realisieren sind. Zudem kommt es aus Kostengründen meist nicht in Betracht, die gesamte Steuerungsfunktionalität mit einer Sicherheitssteuerung zu realisieren. Andererseits ist die parallele Nutzung von Standard- und Sicherheitssteuerung unter Kosten- als auch unter Beherrschbarkeitsgesichtspunkten in vielen Fällen ebenfalls nicht erfolgversprechend.

Demgegenüber wird durch die **Nutzung vorhandener** hardwaretechnischer **Redundanz** bei einer verteilten, kooperativen Steuerungstechnik eine redundante, eventuell diversitäre Implementierung sicherheitsgerichteter Funktionen er-

möglicht (*Meier & Englberger 1999*). Damit kann der Einsatz von teueren Sicherheitssteuerungen vermieden werden.

In einem solchen System kann des weiteren die sicherheitsorientierte und damit aufwendige Programmbearbeitung auf die tatsächlich sicherheitsrelevanten Funktionen beschränkt werden. Damit kann eine exzellente **Skalierbarkeit** der Steuerungslösung hinsichtlich der Sicherheitsanforderungen erreicht werden.

Die Integration sicherheitsrelevanter Funktionen ist hier mit aufgenommen, um einen umfassenden Überblick über die potenziale der verteilten, kooperativen Steuerungstechnik zu geben. Da zu deren Nutzbarmachung allerdings noch ein erheblicher, über den Rahmen dieser Arbeit weit hinausgehender Forschungsbedarf besteht, wird dieser Bereich aus den weiteren Betrachtungen ausgenommen (vgl. hierzu auch Abschnitt 3.2.4).

3.2 Anforderungen

In diesem Abschnitt werden die wesentlichsten Anforderungen zusammen getragen, die hinsichtlich einer verteilten, kooperativen Steuerungstechnik seitens des Werkzeugmaschinenbaus gestellt werden. Diese Anforderungen beziehen sich auf eine effiziente Entwicklung und eine geeignete Modellierungstechnik, eine hohe Beherrschbarkeit in der Inbetriebnahme und im Betrieb sowie eine umfassende Betrachtung aller relevanten Steuerungsfunktionen von Werkzeugmaschinen.

3.2.1 Effiziente Entwicklung

Grundsätzlich besteht im Werkzeugmaschinenbau die Anforderung nach **kürzeren Entwicklungszeiten** (*Siegler 1998, Reinhart u. a. 1999b*). Aufgrund der erweiterten Möglichkeiten der Software steigt der Aufwand für die Entwicklung der Steuerungssoftware bei Werkzeugmaschinen kontinuierlich an Daher müssen Strategien gefunden werden, die diesen Aufwand minimieren, bzw. Vorgehensweisen erarbeitet werden, die die Entwicklungszeit allgemein verkürzen. Dies kann unter anderem durch eine Parallelisierung von Entwicklungstätigkeiten erreicht werden. Dazu muss allein mit Hilfe der Informationen aus den vorgegebenen Spezifikationen der Steuerungsfunktionalität bereits die Steuerungssoftware aus funktionaler Sicht möglichst vollständig entwickelt werden können. Die Anpassung an die jeweilige Steuerungstopologie muss bei geringstem Aufwand zu jedem beliebigen, späteren Zeitpunkt noch möglich sein.

Daher ergibt sich für die Entwicklung von verteilten, kooperativen Steuerungen die Anforderung nach einer optimalen Entkopplung von anderen, eventuell erst später im Entwicklungsprozess durchführbaren Tätigkeiten wie beispielsweise der Installationsplanung, um bereits frühzeitig mit der Steuerungsentwicklung beginnen zu können. Durch eine **Entkopplung der Entwicklungstätigkeiten** verringern sich auch die Fehlerrisiken, da Installationsänderungen dann keine umfangreiche Umgestaltung der Steuerungssoftware zur Folge haben.

Ein Kernproblem der Entwicklung von verteilten, kooperativen Steuerungssystemen ist zunächst die durch die Kommunikations- und Synchronisationsfunktionalität hinzukommende Komplexität. Um den Entwicklungsaufwand dadurch nicht unnötig zu steigern, muss diese Komplexität durch eine geeignete Entwicklungsmethode sowie darauf abgestimmte, Routinetätigkeiten vermeidende Entwicklungswerkzeuge beherrschbar gemacht und möglichst weitgehend verborgen werden. Eine solche **Kapselung der Komplexität** bedeutet insbesondere, dass der Entwickler nicht mit Systeminterna der verteilten, kooperativen Steuerungen konfrontiert wird.

Daraus leitet sich einerseits ab, dass insgesamt eine einfache Programmierung ermöglicht und dabei der Entwickler insbesondere von der aufwendigen und fehlerträchtigen Aufgabe der Programmierung der Kommunikation und Synchronisation entlastet werden muss (vgl. z. B. auch *Weber 1990, S. 7).* Die Kommunikations- und Synchronisationsfunktionalität muss deshalb durch entsprechende Entwicklungswerkzeuge in Abhängigkeit von der Steuerungstopologie automatisch eingefügt werden. Diese Anforderung soll im Rahmen dieser Arbeit als **transparente Kommunikation** bezeichnet werden, da Kommunikations- und Synchronisationsfunktionen zwar implizit vorhanden, für den Programmierer aber nicht sichtbar sind.

Die Software einer verteilten, kooperativen Steuerung setzt sich aus Ablauffunktionen, Kommunikations- und Synchronisationsfunktionen sowie verteilungsabhängigen Schnittstellen zusammen (vgl. z. B. Abschnitt 2.4.2). Dadurch wird die Wiederverwendung solcher Software erheblich erschwert, da die Kommunikations- und Synchronisationsfunktionen sowie die verteilungsabhängigen Schnittstellen sehr stark von der Zuordnung von Sensoren und Aktoren zur Steuerungshardware abhängen. Deshalb muss für die verteilte, kooperative Steuerungstechnik die **Wiederverwendbarkeit** der Ablauffunktionalität unabhängig von Änderungen der Steuerungstopologie gefordert werden.

3.2.2 Geeignete Modellierungstechnik

Für eine effiziente Softwareentwicklung ist eine gute Verständlichkeit der Funktionsweise der Software von höchster Bedeutung. Daher wird für verteilte, kooperative Steuerungen der Einsatz einer **grafischen Modellierungstechnik** gefordert. Sie muss insbesondere auch geeignet sein, einen modularen Aufbau der Steuerungssoftware zu unterstützen.

Bei der Auswahl einer geeigneten Modellierungstechnik ist aber zu beachten, dass aus Akzeptanzgründen im Maschinenbau bereits **etablierte Sprachen** und Modellierungstechniken möglichst weiter verwendet werden können. Dadurch können bestehende, grafische Modellierungswerkzeuge um die Verteilungsfunktionalität erweitert und somit auch zukünftig im Einsatz bleiben. Dabei sollen die im Werkzeugmaschinenbau zu bewältigenden und für eine Verteilung in Frage kommenden Aufgaben der Steuerung von Abläufen in der Peripherie und des Handlings von Werkzeugen und Werkstücken abgebildet werden können.

Um den Umgang mit der Steuerungssoftware weiter zu vereinfachen, wird darüber hinaus gefordert, dass für die anfallenden Entwicklungstätigkeiten bei der Entwicklung verteilter, kooperativer Steuerungssysteme jeweils **aufgabenorientierte Sichten** vorgesehen werden. Dies bedeutet, dass für jede Entwicklungstätigkeit eine grafische Darstellungsform gefunden wird, die möglichst nur diejenigen Informationen enthält, die relevant sind. Dies sind Informationen, die für die jeweilige Tätigkeit als Eingangsinformation benötigt werden, oder Informationen, die durch die Entwicklung generiert werden (siehe hierzu auch Abschnitt 4.2).

3.2.3 Beherrschbarkeit in der Inbetriebnahme und im Betrieb

Um eine effiziente Inbetriebnahme und eine hohe technische Verfügbarkeit im Betrieb zu erreichen, ist die Beherrschbarkeit der Steuerung von zentraler Bedeutung. Es müssen an die verteilte, kooperative Steuerungstechnik zumindest die gleichen Anforderungen gestellt werden, wie sie für die zentrale Steuerungstechnik gelten.

Zunächst muss die Softwareinstallation zügig, aufwandsarm und mit einem geringen Fehlerrisiko durchgeführt werden können. Dazu werden Funktionen für ein einfach beherrschbares **Softwaremanagement** benötigt, mit deren Hilfe die Steuerungssoftware von zentraler Stelle aus weitgehend automatisch in die jeweiligen Steuerungsmodule geladen werden kann. Dazu müssen die einzelnen Elemente der Steuerungssoftware über das Bussystem auf die dafür vorgesehenen Steuerungsmodule übertragen und dort zur Ausführung gebracht werden.

Darüber hinaus wird allgemein im Bereich der Steuerungstechnik eine umfassende **Diagnostizierbarkeit** gefordert. Dazu muss mit einem Visualisierungs- oder einem Diagnosewerkzeug auf Signal- und Funktionszustände sämtlicher im System befindlichen Steuerungsmodule zugegriffen werden können. Darüber hinaus werden Funktionen zur Analyse des Verhaltens der Software sowie der gesteuerten Maschine benötigt. Bezogen auf eine verteilte, kooperative Steuerungstechnik bedeutet dies insbesondere, dass zeitliche und logische Querbezüge zwischen verteilten Signalen und Funktionen unabhängig von deren Verteilung nachvollzogen werden können. Hinzu kommen Funktionen, die der Detektion von Fehlern im Steuerungssystem wie beispielsweise einer fehlerhaften Busverbindung dienen. Neben der Unterstützung der Ursachenfindung müssen dabei auch die Auswirkungen solcher Fehler auf die Steuerungsfunktionalität erkennbar werden, so dass nach einer Ursachenbeseitigung die Inbetriebnahme oder der Betrieb zügig fortgesetzt werden kann.

Einen weiteren Aspekt bezüglich der Beherrschbarkeit stellt das Verhalten des Steuerungssystems bei einem partiellen Ausfall von Steuerungsmodulen oder Kommunikationsverbindungen dar. Dabei wird gefordert, dass dies nicht den Ausfall der gesamten Steuerung zur Folge hat, das Steuerungssystem als ganzes sich also durch **Robustheit** gegenüber partiellen Ausfällen auszeichnet.

3.2.4 Vollständiger Funktionsumfang zur Steuerung von Werkzeugmaschinen

Grundsätzlich muss für eine vollständige Betrachtung der Steuerung von Werkzeugmaschinen neben der in dieser Arbeit behandelten **Ablauffunktionalität** auch die **Geometrieverarbeitung** und die **Sicherheitstechnik** betrachtet werden.

Hinsichtlich der Geometrieverarbeitung soll im Rahmen dieser Arbeit davon ausgegangen werden, dass die NC-Funktionalität, also die interpolierende Steuerung geregelter Achsen als in sich geschlossene und von außen beauftragbare Funktionalität betrachtet werden kann. Ein solcher Achsverbund kann zwar selbst wieder aus einer Koordinations- und mehreren untergeordneten, verteilten Achsreglerfunktionen aufgebaut sein, tritt aber gegenüber der umgebenden Steuerungstechnik lediglich als beauftragbare Funktionalität auf. Dies bedeutet, dass der Achsverbund von beliebigen Koordinationsfunktionen beauftragt werden kann, eine bestimmte Bewegung, Bewegungsfolge oder ein Bearbeitungsprogramm auszuführen. Bei Beendigung dieses Auftrags meldet dies die Koordinationsfunktion des Achsverbunds an das beauftragende Modul zurück.

Die Integration sicherheitsrelevanter Funktionen in ein verteiltes, kooperatives Steuerungssystem führt zu einer Reihe von komplexen und höchst spezifischen,

technischen Anforderungen, die zur Erfüllung der Sicherheitsanforderungen zu konkretisieren und umzusetzen sind (vgl. *Meier & Englberger 1999*). Da in dieser Arbeit jedoch erst ein grundsätzliches Konzept zur Nutzung verteilter, kooperativer Steuerungen erarbeitet werden soll, würde die Integration der Sicherheitstechnik den Rahmen dieser Arbeit bei weitem sprengen und wird daher ausgeklammert. Für eine spätere Integration muss das hier erarbeitete Konzept gegebenenfalls entsprechend erweitert werden.

3.3 Zusammenfassung

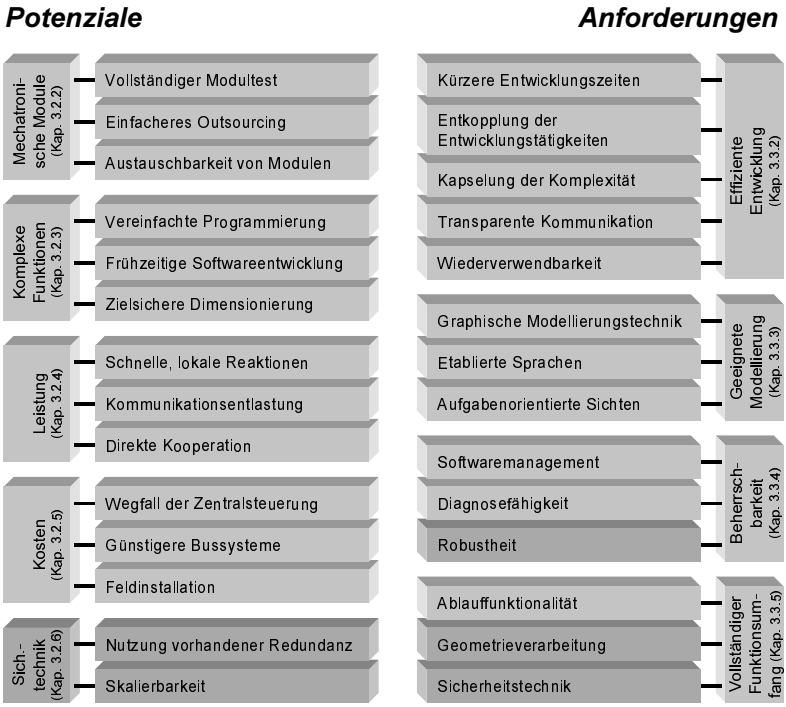

Bild 3-1: Zusammenstellung der Potenziale von und Anforderungen an eine verteilte, kooperative Steuerungstechnik. Nicht weiter verfolgte Aspekte sind dunkel hinterlegt.

In Bild 3-1 werden die analysierten Potenziale von und Anforderungen an eine verteilte Steuerungstechnik zusammengestellt. Dabei sind die in Abschnitt 3.2.4 aus dem im weiteren Verlauf dieser Arbeit zu betrachtenden Funktionsumfang ausgeklammerten Funktionalitäten im Bild dunkel hinterlegt. Da dies auch die Integration der Sicherheitstechnik betrifft, werden im Rahmen dieser Arbeit auch die Potenziale verteilter, kooperativer Steuerungen hinsichtlich der Integration der Sicherheitstechnik nicht weiter berücksichtigt. Ebenfalls wird die Frage der Robustheit nicht eingehender behandelt, da dies im Wesentlichen im Rahmen einer späteren Produktentwicklung erfolgen muss.

4 Modellierungstechnische Grundlagen

Für die Entwicklung des Konzepts (Kapitel 5) sollen im vorliegenden Kapitel die modellierungstechnischen Grundlagen behandelt werden, wobei im Rahmen dieser Arbeit unter Modellierung die eindeutige Festlegung des gewünschten Steuerungsverhaltens verstanden werden soll. Sie umfasst damit den gesamten kreativen Prozess von der Konzeption der Funktionalität bis hin zu deren Ausdetaillierung zu einem lauffähigen Steuerungsprogramm. Die dazu verwendeten Sprachmittel sollen hier in ihrer Gesamtheit als Modellierungstechnik bezeichnet werden.

In Abschnitt 4.1 werden zunächst bestehende Techniken zur Modellierung maschinennaher Abläufe vorgestellt und darüber hinaus weitere, durch die Forschung vorgeschlagene Modellierungstechniken behandelt. Damit soll es im Rahmen des Konzepts ermöglicht werden, für die im Werkzeugmaschinenbau etablierten Modellierungstechniken eine gemeinsame, verteilbare Basismodellierungstechnik zu erarbeiten.

Im Bereich der allgemeinen Softwareentwicklung wie auch bei der Modellierung von Echtzeitanwendungen haben objektorientierte Modellierungstechniken mittlerweile einen hohen Verbreitungsgrad gefunden. Verschiedene Arbeiten behandeln auch deren Einsatz im Maschinenbau (z. B. *Schelberg 1994, Kieß 1995, Awad u. a. 1996*). Daher wird in Abschnitt 4.2 eine Auswahl solcher objektorientierter Modellierungstechniken vorgestellt.

Ziel dieses Abschnitts ist es, einerseits deren Einsetzbarkeit als Basismodellierungstechnik zu überprüfen und andererseits zur Modellierung der Systemsoftware verwendete Modellierungstechnik einzuführen. In Abschnitt 4.3 werden die vorgestellten Techniken bewertet und für das folgende Kapitel 5 ausgewählt.

4.1 Techniken zur Modellierung maschinennaher Abläufe

4.1.1 Modellierungstechniken der DIN-IEC 1131-3

Im Bereich der SPS-Technik hat die *DIN IEC 1131-3 (1992)* mittlerweile Verbreitung gefunden. Sie definiert zur Modellierung von Steuerungsfunktionen fünf Sprachen, die im folgenden kurz vorgestellt werden.

- Anweisungsliste (AWL) stellt eine textbasierte Sprache auf Niveau von Assemblerbefehlen dar.

- Strukturierter Text (ST) ist eine höhere, ebenfalls textuelle Programmiersprache, die sich stark an der Programmiersprache Pascal orientiert.

Da diese Sprachen keine grafische Repräsentation besitzen, kommen sie hier zunächst nicht weiter in Betracht.

- Kontaktplan (KOP) formuliert Verknüpfungen in einer Form, die im Zusammenhang mit elektromechanischen Verknüpfungssteuerungen entstanden ist. Dementsprechend stellt sie im Kern eine grafische Repräsentation boolescher Ausdrücke dar.

- Funktionsbausteinsprache (FBS) ist ebenfalls eine grafische, an Schaltplänen orientierte Modellierungstechnik, deren Symbolik aber aus der Elektronik stammt. Gegenüber dem Kontaktplan können dementsprechend auch komplexere Verknüpfungsglieder wie Zähler etc. dargestellt werden.

- Ablaufsprache (AS) ist eine zu Schrittketten gleichwertige Modellierungstechnik. Daher wird hier auf den Abschnitt 4.1.4 verwiesen.

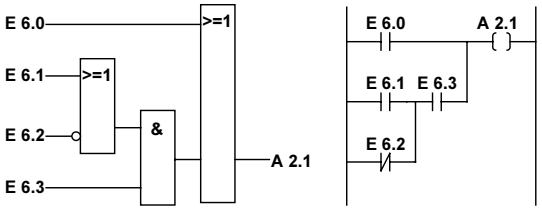

Bild 4-1: Funktionsbausteinsprache (FBS) und Kontaktplan(KOP) entsprechend DIN IEC 1131-3 im Überblick.

Bild 4-1 zeigt einen Überblick über Funktionsbausteinsprache und Kontaktplan aus der DIN IEC 1131-3. Es handelt sich dabei um zwei verschiedene, grafische Notationen zur Modellierung boolescher Ausdrücke. Die Steuerung komplexerer Abläufe ist allerdings trotz der grafischen Repräsentation aufgrund der starken Spezialisierung auf die Verknüpfungsmodellierung nur schwer möglich.

4.1.2 Kontrollstrukturen

Nach *Balzert* (*1996, S. 238*) „legen Kontrollstrukturen innerhalb eines Algorithmus fest, in welcher Reihenfolge, ob und wie oft Anweisungen ausgeführt werden sollen. Die strukturierte Programmierung erlaubt nur solche Kontrollstrukturen, die genau einen Ein- und einen Ausgang haben. Man nennt solche Kontrollstrukturen daher lineare Kontrollstrukturen. Es lassen sich vier verschiedene Typen unterscheiden: die Sequenz, die Auswahl, die Wiederholung und der Aufruf. Alle Typen lassen sich beliebig miteinander kombinieren und ineinander schachteln. Es gibt vier grafische Darstellungsformen: Struktogramme, auch Nassi-Shneiderman-Diagramme genannt, Jackson-Diagramme, Warnier-Orr-Diagramme und Programmablaufpläne (PAP).

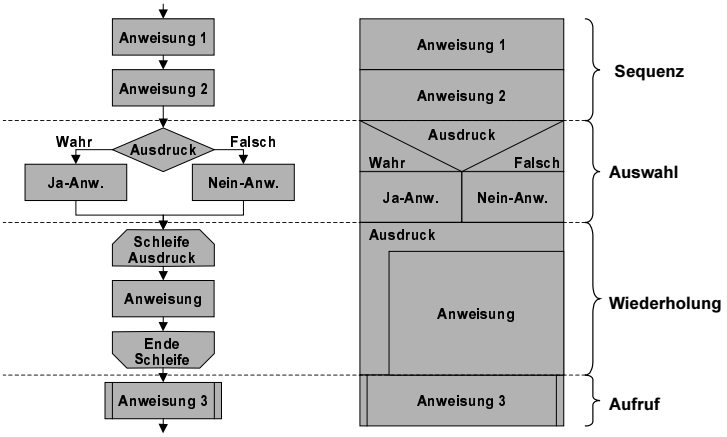

Bild 4-2: Typen von Kontrollstrukturen im PAP nach DIN 66001 (1983) und im Struktogramm nach DIN 66261 (1985).

Bild 4-2 zeigt diese vier grundlegenden Typen von Kontrollstrukturen in den Darstellungsformen des Struktogramms (siehe hierzu *Nassi & Shneiderman 1973* oder *DIN 66261 1985*) und des Programmablaufplans (siehe *DIN 66001 1983*). Jackson-Diagramme (*Jackson 1975*) und Warnier-Orr-Diagramme (*Warnier 1979, Orr 1977* und *DIN EN 28631*) werden aufgrund ihrer geringen Verbreitung nicht weiter behandelt.

Sollen Kontrollstrukturen für die Modellierung ereignisorientierter Steuerungssoftware angewandt werden, tritt das Problem auf, dass sich sowohl nebenläufige Abläufe als auch das Warten auf Ereignisse aus dem gesteuerten Prozess nicht

explizit darstellen lassen. Dies kann jedoch durch entsprechende Erweiterungen geschehen, wie sie beispielsweise in der Programmierumgebung der Firma Jetter (*Jetter 1999*) eingesetzt werden. In Bild 4-3 sind diese Erweiterungen am Beispiel eines Ausschnitts einer Steuerungsfunktion dargestellt. Derart erweiterte, auf Kontrollstrukturen basierende Modellierungstechniken sind folglich gut für die Modellierung von maschinennahen Abläufen geeignet.

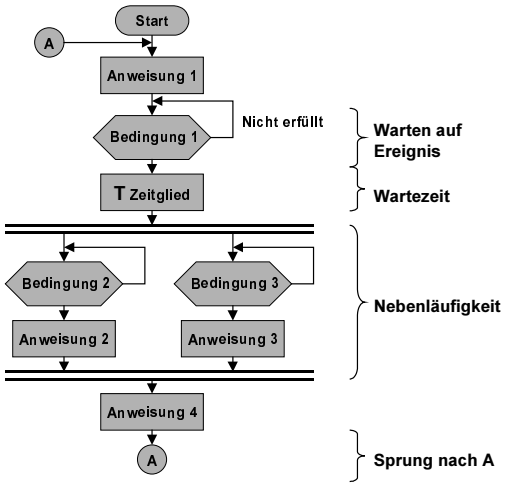

Bild 4-3: Darstellung einer Funktion mittels Programmablaufplan entsprechend Jetter 1999.

4.1.3 Zustandsgrafen

Zur Modellierung des Verhaltens von Maschinen und deren Komponenten eignen sich auch Zustandsgrafen. Grundsätzlich bestehen sie aus Zuständen und sogenannten Transitionen, die den Übergang zwischen solchen Zuständen beschreiben. Eine Komponente wie beispielsweise ein Werkzeugwechslergreifer befindet sich immer in genau einem Zustand. Er ist also immer auf, zu, öffnet sich oder schließt sich. Eine Zustandsänderung wird durch das Ausführen einer Transition erreicht, was durch ein Prozessereignis wie ein Sensorsignal oder einen Funktionsaufruf durch eine übergeordnete Funktion bewirkt werden kann.

Im Bereich der Automatentheorie begegnet man auch dem Begriff des Zustandsautomaten, der ein System bezeichnet, welches sich durch einen Zustandsgrafen

beschreiben lässt. Der Zustand eines Systems beinhaltet dabei implizit die Informationen, die sich aus den bisherigen Eingaben ergeben haben und die benötigt werden, um die Reaktion des Systems auf noch folgende Eingaben zu bestimmen (*Balzert 1996*, S. 270). Bezogen auf die maschinennahe Steuerungstechnik sind dabei unter Eingaben sowohl Ereignisse aus dem Prozess, als auch Aufrufe aus überlagerten, koordinierenden Steuerungsfunktionen zu verstehen.

In der Literatur sind Mealy- und Moore-Automaten sowie Zustandsautomaten nach Harel bekannt. Da die auch unter dem Namen Statecharts bekannten Harel-Zustandsautomaten (*Harel 1987*) eine Erweiterung der Mealy- und Moore-Automaten darstellen, soll hier bezüglich letzterer auf die Literatur verwiesen werden (z. B. *Hopcroft & Ullman 1979*). Die im Zusammenhang der Steuerungstechnik wichtigsten Konzepterweiterungen betreffen dabei nebenläufige Zustände und hierarchische Zustandsautomaten, was zu einer wesentlich verbesserten Übersichtlichkeit führt.

Bild 4-4 zeigt einen solchen Harel-Automaten am Beispiel eines Ausschnitts einer Werkzeugwechselfunktion, von der ein Werkzeuggreifer und das Werkzeugspannsystem in der Spindel jeweils durch einen nebenläufigen Grafen modelliert sind. Im Grafen des Werkzeugspannsystems wird dabei ein Beispiel hierarchischer Zustände - „Öffnet“ und „Schließt“ - gegeben. Für eine detailliertere Betrachtung sei hier auf die Literatur verwiesen (*Harel 1987*).

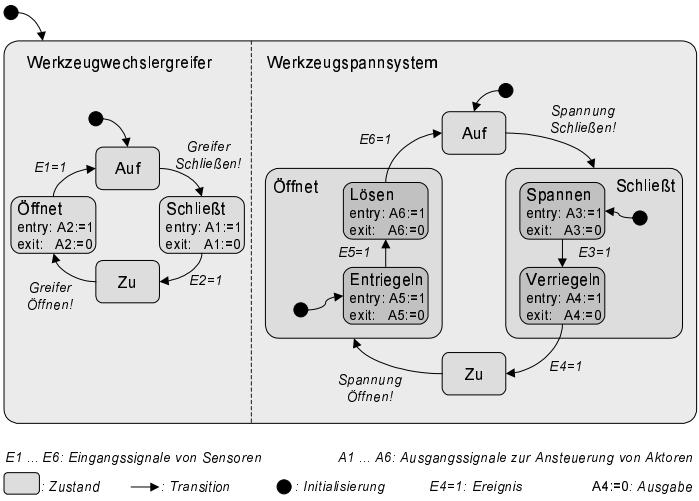

Bild 4-4: Beispielhafte Modellierung eines Ausschnitts einer Werkzeugwechselfunktion mittels eines Zustandsgrafen nach Harel (1987).

Im Bereich der Steuerung maschinennaher Abläufe werden an Zustandsgrafen orientierte Modellierungstechniken in zahlreichen Forschungsarbeiten (*Fleckenstein 1987, Brandl u. a. 1996, Storr u. a. 1997, Reinhart u. a. 1999a*) und vereinzelt auch in kommerziellen Entwicklungswerkzeugen eingesetzt. Prominentester Vertreter eines zustandsgrafenbasierten Entwicklungswerkzeugs für diesen Anwendungsbereich ist das von der Firma Siemens angebotene Werkzeug HiGraph (*Rath 1995*). Es beruht abgesehen vom Hierarchiekonzept, der grafischen Notation und einigen spezifischen, weiterführenden Vereinbarungen auf einer den Harel-Automaten ähnlichen Modellierungstechnik. Das Tool Statemate (*Harel 1990*) basiert dagegen direkt auf Harel-Automaten und wird bei der Programmierung reaktiver Systeme häufig angewendet (z. B. *Eckrich 1997*), hat aber bisher im Werkzeugmaschinenbau keine Verbreitung gefunden.

Nach *Balzert* (*1996*, S. 291) eignen sich Zustandsautomaten zur Modellierung von Systemen und Geräten, deren Verhalten von der bis dahin durchlaufenen Historie abhängt, wie dies auch in Bild 4-4 zu erkennen ist. Andererseits können mit Zustandsgrafen zwar parallele Prozesse durch nebenläufige Zustandsgrafen modelliert werden. Die dabei häufig benötigten Synchronisationen können aber in der grafischen Repräsentation nicht explizit dargestellt werden. Dies widerspricht dem im Maschinenbau in erster Linie durch Abläufe getrieben Vorgehen. So wird ein Mechanikkonstrukteur die Funktion eines Werkzeugwechslers durch die Abfolge seiner Bewegungen beschreiben. Abläufe sind bei nebenläufigen Zustandsgrafen zwar in deren kollektivem Verhalten implizit enthalten, sind aber nicht expliziter Gegenstand der Modellierung. Ansätze, die den Zustandsgrafen spezielle Synchronisationskonzepte hinzufügen (z. B. *Herrscher 1981*), gehen damit bereits deutlich in Richtung der Petrinetze (siehe folgender Abschnitt).

4.1.4 Petrinetze

Ebenfalls kommen zur Steuerung maschinennaher Abläufe Petrinetze, die auf eine Dissertation von Carl Adam Petri zurückgehen (*Petri 1962*), in Betracht (*Abel 1990, Möhrle 1989, Sabbah 1995, Wagner 1997)*. Nach Balzert (*1996*, S. 318) eignen sich Petrinetze besonders gut zur Modellierung von Systemen mit kooperierenden Prozessen, weshalb sie hier näher betrachtet werden sollen. Für einen noch tieferen, theoretischen Hintergrund muss allerdings hier auf die umfangreiche Literatur zum Thema Petrinetze verwiesen werden (z. B. *Budde & Nieters 1992, Luttenberger & Cramer 1992, Peterson 1981, Moßig & Stäble 1995)*.

Ein Petrinetz ist ein gerichteter Graf, der aus zwei verschiedenen Sorten von Knoten besteht, den Stellen und Transitionen. Dabei entspricht eine Stelle einer

Zwischenablage von Informationen, eine Transition beschreibt die Verarbeitung von Informationen (*Balzert 1996*, S. 298). Üblicherweise werden Stellen – manchmal auch Zustände genannt – durch Kreise, Transitionen durch Balken oder Rechtecke dargestellt. Stellen und Transitionen werden in abwechselnder Reihenfolge durch sogenannte Kanten - dargestellt durch Pfeile – miteinander verbunden, entlang derer Objekte – sogenannte Marken – über Transitionen von Stelle zu Stelle weitergeschaltet werden. Dabei gilt allgemein als Bedingung, dass die Stellen im Vorbereich einer Transition für den Schaltvorgang ausreichend Marken bereitstellen können, die Stellen im Nachbereich einer Transition ausreichend Marken aufnehmen können und eine zusätzliche Transitionsbedingung erfüllt ist. Bild 4-5 gibt einen Überblick über diese Grundelemente zum Aufbau von Petrinetzen.

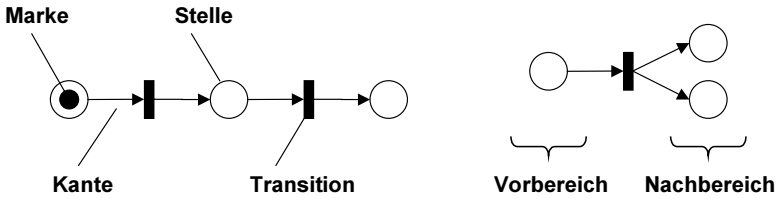

Bild 4-5: Grundelemente zum Aufbau von Petrinetzen.

Bei Petrinetzen wird eine Vielzahl verschiedener Netzklassen unterschieden, von denen die im Rahmen dieser Arbeit relevantesten kurz beschrieben werden sollen. Diese Netzklassen unterscheiden sich dabei im wesentlichen durch die Kapazitäten von Stellen, das Gewicht der einzelnen Kanten und die Unterscheidbarkeit von Markentypen. Dabei bedeutet die Kapazität einer Stelle, wie viele Marken sie aufzunehmen in der Lage ist und das Gewicht von Kanten, wie viele Marken bei einem Schaltvorgang transportiert werden. Dabei steigt mit zunehmender Komplexität der Netzklasse auch deren Mächtigkeit an. Über die im Folgenden genannten Netzklassen hinaus gibt es noch weitere, mächtigere Netzklassen, die aufgrund ihrer Komplexität hier nicht behandelt werden sollen. Daneben existieren zahlreiche anwendungsspezifische Erweiterungen die hier ebenfalls nicht diskutiert werden.

Die einfachste Netzklasse stellen die **Bedingungs-/Ereignisnetze**, kurz **B/E-Netze**, dar. Hier sind Kantengewichte und Stellenkapazitäten grundsätzlich gleich eins. Eine Unterscheidung verschiedener Marken findet nicht statt. Damit entspricht das B/E-Netz der Ablaufsprache aus *DIN IEC 1131-3* beziehungsweise

der Modellierungstechnik der Schrittketten, wie sie aus einigen Entwicklungsumgebungen für SPSen (z. B. *Siemens 1999a*, *Helmey 1995*) bekannt sind.

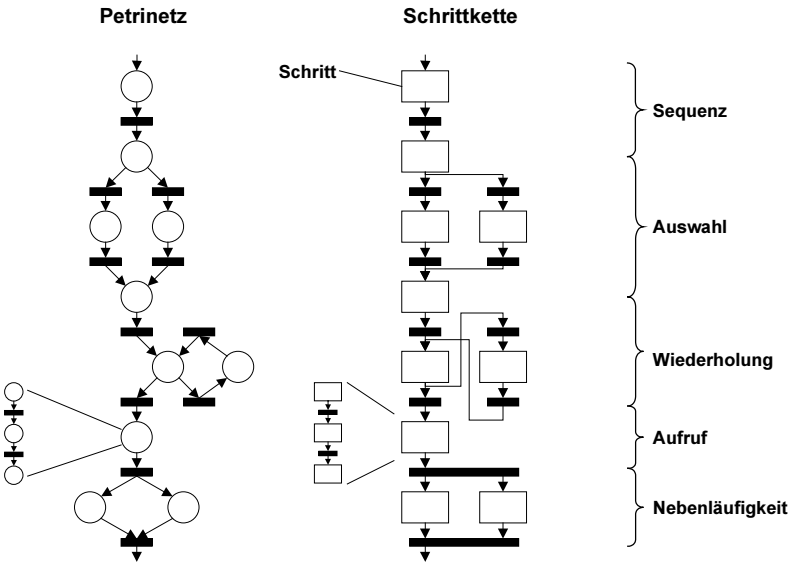

Bild 4-6: Bedingungs-/Ereignisnetze im Vergleich zu Schrittketten.

Bild 4-6 zeigt B/E-Netze im Vergleich zu Schrittketten. Dabei ist beispielhaft die Abbildung der aus Abschnitt 4.1.2 bereits bekannten Typen von Kontrollstrukturen dargestellt. Die herausragende Eigenschaft der Petrinetze - und damit auch der Schrittketten - ist dabei die Möglichkeit, nebenläufige, kooperierende Prozesse explizit modellieren zu können. Synchronisationen zwischen nebenläufigen Systemen können dabei durch eine geeignete Netzstruktur erzwungen werden (*Balzert 1996*, S.319).

In Entwicklungswerkzeugen für Schrittketten wie auch in Forschungsarbeiten (z. B. *VDW 1997b)* sind dabei üblicherweise Möglichkeiten vorgesehen, die benötigten Prozesseingänge zu lesen und entsprechend Prozessausgänge zu manipulieren, um so die Schrittkette beziehungsweise das Petrinetz zur Steuerung von Abläufen einsetzen zu können. Dabei wird in der Transitionsbedingung ein boolescher Ausdruck ausgewertet, in dem Eingangssignale, Variablen- und Timerstände berücksichtigt werden. Häufig findet sich hier eine grafische Repräsentation der Transitionsbedingung in Form von Kontaktplan (KOP, siehe Abschnitt

4.1.1). Dies ist insbesondere deshalb günstig, da die Ursache einer nicht gesetzten Transitionsbedingung und damit der Grund für eventuelle Maschinenstillstände in der Kontaktplannotation sofort erkennbar ist (siehe Bild 4-7). Die Manipulation von Prozessausgängen, Variablen und Timern wird dagegen in den Schritten, häufig in Form von Anweisungsliste (AWL, siehe Abschnitt 4.1.1) modelliert.

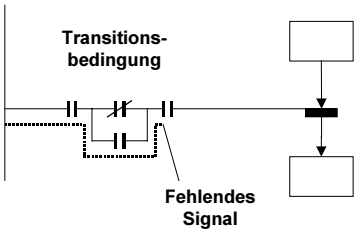

Bild 4-7: Fehlendes Sensorsignal in einer Transitionsbedingung nach der Kontaktplannotation.

Durch B/E-Netze können auch Zustandsgrafen, wie sie beispielsweise das Tool HiGraph (siehe Abschnitt 4.1.3) verwendet, dargestellt werden. Bild 4-8 zeigt dies an einem einfachen Beispiel. Damit eignen sich B/E-Netze zusätzlich zur Abbildung von Schrittketten auch zur Abbildung der im Maschinenbau gebräuchlichen Zustandsgrafen.

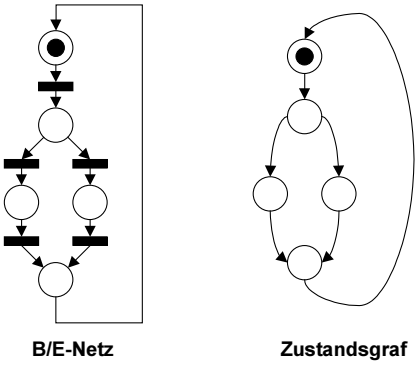

Bild 4-8: Als Bedingungs-/Ereignisnetz dargestellter Zustandsgraf

Bei der Netzklasse der **Stellen-/Transitionennetze**, kurz **S/T-Netze**, sind dagegen die Stellenkapazitäten und die Kantengewichte ganzzahlig frei modellierbar. Dadurch können komplexere Sachverhalte mit wenigen Netzelementen modelliert werden. Ein Beispiel hierfür wird in Bild 4-9 angegeben.

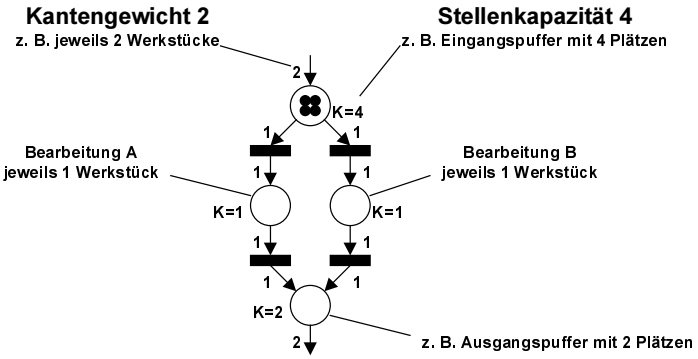

Bild 4-9: Beispiel eines Stellen-/Transitionennetzes.

Im S/T-Netz sind alle Marken identisch, können also nicht voneinander unterschieden werden. Bei der Netzklasse der **colorierten Petrinetze** (**CPN**) werden Markentypen eingeführt, die häufig auch durch farbige Marken repräsentiert werden, was zur Bezeichnung coloriertes Petrinetz führt. Markenfarben eignen sich z. B. sehr gut, um Materialflüsse zu modellieren, indem die Markenfarben als Werkstücktypen interpretiert werden.

In colorierten Petrinetzen wird an den Kanten das Kantengewicht abhängig von der Markenfarbe angegeben. Es wird also festgelegt, wie viele Marken einer bestimmten Farbe bei einem Schaltvorgang über die Kante transferiert werden. Ebenso ist die Kapazität der Stellen farbspezifisch angegeben. Darüber hinaus können, wie in *Glüer & Schmidt (1988)* beschrieben, sogenannte Farbgruppen definiert werden, so dass durch eine entsprechende Kantenanschrift eine bestimmte Anzahl von Marken mit Zugehörigkeit zu einer bestimmten Farbgruppe transferiert werden. Analog dazu wird die Kapazität der Stellen farbgruppenspezifisch angegeben. Bild 4-10 zeigt die Elemente der colorierten Petrinetze.

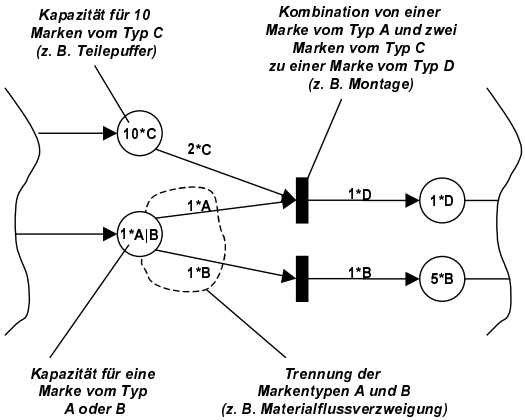

Bild 4-10: Elemente colorierter Petrinetze.

In der Literatur findet sich eine Vielzahl von meist sehr theoretisch gehaltenen Ansätzen, bei denen colorierte Petrinetze zur Steuerung von Produktionssystemen vorgeschlagen werden. Hierbei liegt der Fokus häufig auf Flexiblen Fertigungssystemen oder auf Montageanlagen (z. B. *Colombo 1998*, *Glüer & Schmidt 1988, Kasturia u. a. 1988*). Die Verwendung von Petrinetzen wird dabei häufig mit deren Eignung zur Modellierung nebenläufiger, synchronisierter Prozesse begründet. Die Netzklasse der colorierten Petrinetze wiederum wird dabei in der Regel aufgrund ihrer Eignung zur Modellierung von Materialflüssen eingesetzt.

In Bild 4-11 ist eine weitere, auf die Semantik der colorierten Petrinetze zurückführbare, sehr anschauliche und in eine Entwicklungsmethode für Montageanlagen von Feldmann und Cuiper (siehe z. B. *Reinhart & Cuiper 1999* und *Feldmann 1997*) eingebettete Modellierungstechnik dargestellt. Dabei zeigt Bild 4-11 einen Ausschnitt aus einem Steuerungsnetz zusammen mit seiner Entsprechung in Form von colorierten Petrinetzen. Hierbei werden Netze für den Montagevorgang und die Betriebsmittel miteinander verknüpft und so in anwendungsorientierter Weise Steuerungsnetze für Montageanlagen modelliert.

Zusammenfassend kann festgestellt werden, dass colorierte Petrinetze die Mächtigkeit besitzen, alle für die Modellierung maschinennaher Abläufe relevanten Sachverhalte abzubilden.

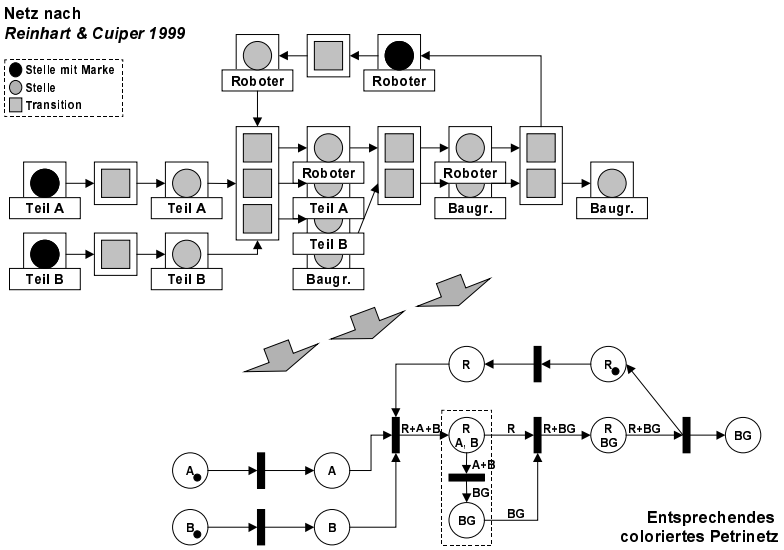

Bild 4-11: Petrinetz nach Reinhart & Cuiper (1999) zur Modellierung eines Montageablaufs und dessen Abbildung auf ein coloriertes Petrinetz. Im vorliegenden Beispiel sind Kantengewichte und Stellenkapazitäten eins und deshalb nicht angegeben.

4.2 Objektorientierte Modellierungstechniken

Multisichtorientierte Modellierungstechniken vereinen verschiedene, zur Lösung eines Modellierungsproblems benötigte, sichtspezifische Modellierungstechniken zu einer integrierten Modellierungssprache. Dabei sind die Einzelsichten derart in Beziehung zueinander gesetzt, dass sich daraus ein eindeutiges, redundanzfreies Modell ableiten lässt. Heute werden dabei hauptsächlich sogenannte objektorientierte Modellierungstechniken eingesetzt. Deshalb werden zunächst in Abschnitt 4.2.1 einige der wichtigsten Begriffe der objektorientierten Modellierungstechnik kurz erläutert.

Unter der Vielzahl solcher Modellierungstechniken sind in dieser Arbeit drei Techniken mit besonderer Relevanz dargestellt. Abschnitt 4.2.2 beschreibt die UML (Unified Modeling Language), die aus den verbreitetsten, objektorientierten Techniken hervorgegangen ist. Sie gilt heute als die wichtigste und aktuellste objektorientierte Modellierungstechnik.

Eine weitere Modellierungstechnik, die für diese Arbeit relevant ist, ist die Notation der Methode ROOM (Real Time Object Oriented Modeling), die speziell für die Modellierung von verteilten, reaktiven Systemen im Echtzeitbereich entwickelt wurde. ROOM konnte bereits im Forschungsbereich zur Modellierung von Maschinensteuerungen eingesetzt werden und wird deshalb in Abschnitt 4.2.3 näher betrachtet.

Elemente von ROOM finden sich auch in UML-RT (Real Time), einer Erweiterung von UML für die Spezifikation echtzeitorientierter Software. Sie wird daher als dritte Modellierungstechnik in Abschnitt 4.2.4 behandelt.

4.2.1 Objektorientierte Begriffswelt

Die grundlegenden Begriffe der Objektorientierung sind die des **Objektes**, der **Klasse** und der **Vererbung**.

- **Objekt**: Objekte sind Konstrukte aus zusammengehörenden Daten (auch als **Attribute** oder **Member-Variablen** bezeichnet) und darauf operierenden Funktionen (auch als **Methoden, Operationen** oder **Member-Funktionen** bezeichnet). Die Daten eines Objekts können nur mittels seiner Methoden, nicht aber durch direkten Zugriff manipuliert werden. Dieses Verbergen von Informationen wird auch als **Datenkapselung** bezeichnet. Die Funktionalität eines Systems ergibt sich durch das Zusammenwirken von Objekten.

- **Klasse**: Objekte, die ein gemeinsames Verhalten aufweisen, gehören zur gleichen Klasse. Eine Klasse ist die allgemeine Spezifikation einer beliebigen Anzahl von Objekten, welche auch als **Instanzen** der Klasse bezeichnet werden.

- **Vererbung**: Eine Klasse kann von einer anderen Klasse Attribute und Methoden erben und sich somit deren Eigenschaften zu eigen machen, ohne sie neu entwickeln zu müssen. Durch Hinzufügen weiterer Daten und Methoden entstehen so neue Klassen, sogenannte **abgeleitete Klassen**. Die Klasse, die Ausgangspunkt einer Vererbung ist, nennt man **Basisklasse**.

Für ein detaillierteres Verständnis der allgemeinen Grundlagen objektorientierter Modellierung und Softwareentwicklung wird auf die vielfältige Fachliteratur zu diesem Thema verwiesen (z. B. *Rumbaugh u. a. 1991*, *Jakobson u. a. 1992*, *Coad & Yourdon 1991a/b*, *Booch 1994, Shlaer & Mellor 1992*).

4.2.2 Unified Modeling Language (UML)

Die Unified Modeling Language *(Booch u. a. 1998)* vereinigt wesentliche Elemente mehrerer verschiedener objektorientierter Modellierungstechniken, wie beispielsweise *Rumbaugh u. a. (1991)* und *Booch (1994).* Sie stellt heute einen Quasi-Standard im Bereich Objektorientierte Analyse und Design dar. Die Elemente der UML-Notation, welche im Rahmen dieser Arbeit Verwendung finden, werden hier kurz vorgestellt.

Die UML erlaubt es, ein zu modellierendes System unter verschiedenen Blickwinkeln, sogenannten Entwicklungssichten, zu betrachten. Dazu stehen unterschiedliche Arten von Diagrammen zur Darstellung der einzelnen Sichten zur Verfügung.

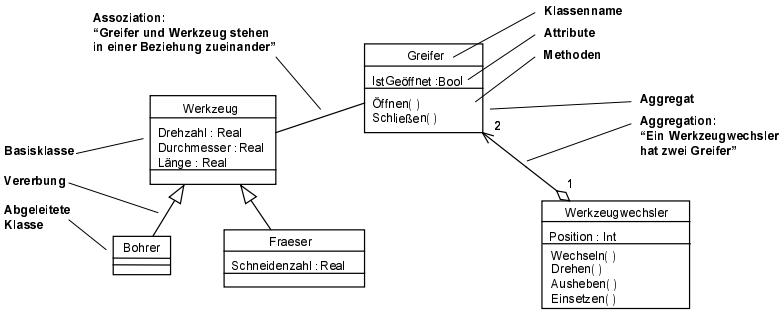

Bild 4-12: Klassendiagramm der UML und dessen wichtigste Elemente.

- **Klassendiagramme**: In Klassendiagrammen können die Klassen mit ihren Attributen und Methoden sowie Beziehungen von Klassen untereinander dargestellt werden. Bild 4-12 gibt ein Beispiel mit den gebräuchlichsten Beziehungsarten.

- **Interaktionsdiagramme:** Durch die Interaktionsdiagramme wird das Zusammenwirken von Objekten über den Austausch von Botschaften veranschaulicht. Es werden dabei zwei Darstellungsformen dieser Diagramme unterschieden. Während die **Sequenzdiagramme** stärker die zeitliche Abfolge von Ereignissen und Operationen betonen, stellen die **Kollaborationsdiagramme** die Objektstruktur stärker in den Vordergrund. Beide Diagrammarten sind jedoch äquivalent zueinander und können ineinander überführt werden (vgl. Bild 4-13).

- **Komponentendiagramme:** Sie dokumentieren, welche Klassen in welchen Dateien implementiert werden und bieten eine Übersicht über die Verzeichnisstruktur in der Implementierungsphase.
- **Zustandsdiagramme:** Die Zustandsdiagramme in der UML entsprechen im Wesentlichen den Zustandsgrafen nach Harel (siehe Abschnitt 4.1.3), wobei durch die Zustandsdiagramme das Verhalten von Klassen spezifiziert wird. Dabei kann der Zustand von Objekten der Klasse durch den Aufruf von Methoden manipuliert werden.
- **Aktivitätsdiagramme:** Die Aktivitätsdiagramme der UML dienen der Modellierung der internen Abläufe von Funktionen. Ihre Notation und ihre Modellierungsmöglichkeiten sind in etwa vergleichbar mit den Ablaufdiagrammen nach *Jetter* (*1999*, siehe Abschnitt 4.1.2). Allerdings ist demgegenüber bei Aktivitätsdiagrammen nicht vorgesehen, auf externe Ereignisse zu reagieren.
- **Use Case Diagramme (Anwendungsfalldiagramme):** Use Case Diagramme dienen der Beschreibung von Arbeitssituationen bei der Anwendung eines Systems.

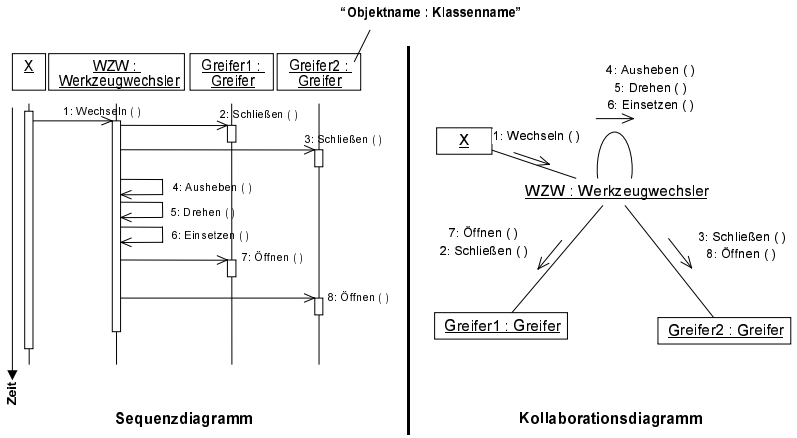

Bild 4-13: Sequenz- und Kollaborationsdiagramm der UML.

Des weiteren existieren noch **Implementierungsdiagramme**, die an dieser Stelle nicht weiter erläutert werden. Die UML sieht darüber hinaus noch das Konzept der **Stereotypen** vor, mit denen spezielle Klassentypen definiert werden können.

Damit kann für bestimmte Anwendungsfelder die Notation der UML entsprechend erweitert werden. Für eine ausführlichere Darstellung der Notation sei auf die Literatur zur UML verwiesen (*Booch u. a. 1998, Oestereich 1997*).

Damit lässt sich feststellen, dass die UML eine außerordentlich umfassende Modellierungstechnik ist. Sie ist aber in keiner Weise auf die Bedürfnisse der Entwicklung maschinennaher Abläufe spezialisiert, sondern orientiert sich an allgemeinen Problemstellungen der Softwaretechnik.

4.2.3 Real Time Object Oriented Modeling (ROOM)

Speziell für den Einsatz bei reaktiven Systemen ist die auf objektorientierten Prinzipien basierende Modellierungstechnik ROOM zugeschnitten (*Selic u. a. 1994,* S.7). Der amerikanische Begriff real time wird dabei im Sinne reaktiver Systeme verwandt und lässt sich daher nicht wörtlich mit dem Begriff Echtzeit übersetzen. Folglich deckt die ROOM-Notation die Modellierung von Echtzeitsystemen nur hinsichtlich des reaktiven Systemcharakters ab. Im Zusammenhang mit dieser Arbeit wichtige Elemente dieser Modellierungstechnik sollen daher im Folgenden kurz erläutert werden. Bild 4-14 zeigt ein einfachen Beispiel eines in ROOM modellierten Systems.

- **Aktor-Klassen**: ROOM definiert als zentrales Element sogenannte Aktor-Klassen, die zur Abbildung von Softwarekomponenten verwendet werden. Dabei sind Instanzen dieser Aktor-Klassen zueinander nebenläufig, d. h. sie werden durch ein unterlagertes Multitasking-Betriebssystem jeweils mit eigener Rechenzeit ausgestattet. Aktor-Klassen können nach ROOM ineinander geschachtelt werden. Damit können komplexere Aktor-Klassen durch miteinander verknüpfte, einfachere Aktor-Klassen aufgebaut werden.

- **Protokoll-Klassen**: Die Kommunikation zwischen den Aktorobjekten geschieht über entsprechende Schnittstellen, die mittels der Protokoll-Klassen definiert werden. Dabei wird die Schnittstelle einer Aktor-Klasse durch Ports, über die Nachrichten entsprechend der Protokoll-Klasse versendet werden können, beschrieben. Der jeweils zugehörige Empfänger-Port wird zum Sender-Port als konjugiert bezeichnet.

- **Zustandsdiagramme**: Zu den Aktor-Klassen wird mit Hilfe von eng an die Zustandsgrafen von Harel (siehe Abschnitt 4.1.3) angelehnten Zustandsdiagrammen jeweils ein Verhalten beschrieben. Dabei werden Transitionen durch eingehende Nachrichten ausgelöst, Manipulationen interner Daten oder das Versenden von Nachrichten geschieht beim Schalten einer Transition oder beim Verlassen von beziehungsweise beim Eintreten in Zustände.

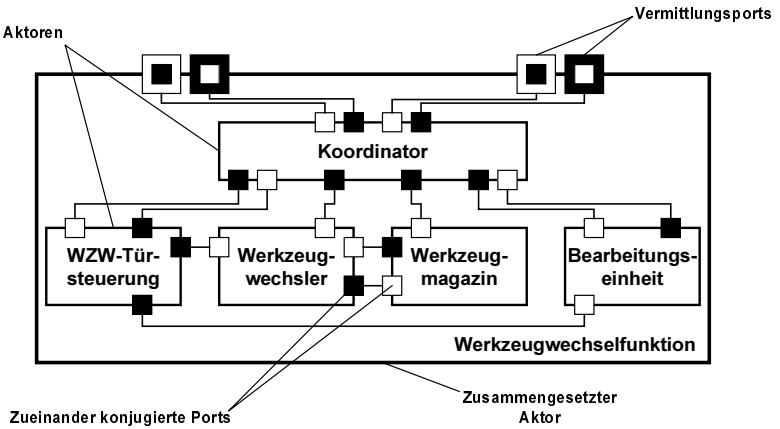

Bild 4-14: Elemente von ROOM am Beispiel eines entsprechend modellierten Systems.

ROOM beinhaltet darüber hinaus noch ausgefeilte Vererbungsmechanismen sowie verschiedene Konstrukte zur Erweiterung der Modellierungsmöglichkeiten. Insbesondere ist diesbezüglich die Möglichkeit der Replikation von Ports und Aktorobjekten zu nennen. So können an einen replizierbaren Port mehrere entsprechende Ports anderer Aktorobjekte angeschlossen werden, wie dies in Bild 4-15 dargestellt ist. Dabei werden implizit mehrere gleichartige Ports erzeugt, über die dann jeweils identische Nachrichten versendet werden, was zu einer erheblich verbesserten Flexibilität in der Modellierung führt. Ebenso kann innerhalb einer Aktorklasse eine in ihr enthaltene Komponente durch Replikation nach Bedarf vervielfältigt werden. Für ein detaillierteres Verständnis von ROOM wird hier auf die Literatur (*Selic u. a. 1994*) verwiesen.

Zu ROOM gibt es entsprechende Softwarewerkzeuge, die es ermöglichen, ein in ROOM modelliertes System direkt auf eine Laufzeitumgebung zu übertragen. Dazu wird eine sogenannte ROOM Virtual Machine eingesetzt, die die notwendige Funktionalität zur Abbildung der Aktorobjekte sowie zur Kommunikationsabwicklung bereitstellt. Damit erfüllt sie für ein in ROOM modelliertes System eine vergleichbare Funktion, wie die in dieser Arbeit entwickelte Systemsoftware für ein verteiltes, kooperatives Steuerungssystem.

Um die angesprochene Übertragung in eine Laufzeitumgebung zu ermöglichen, wird in ROOM in einer üblichen Programmiersprache (meist C++) die benötigte Funktionalität bis ins Detail spezifiziert. Eine eigene Implementierungsphase außerhalb des ROOM-Werkzeugs ist damit nicht notwendig.

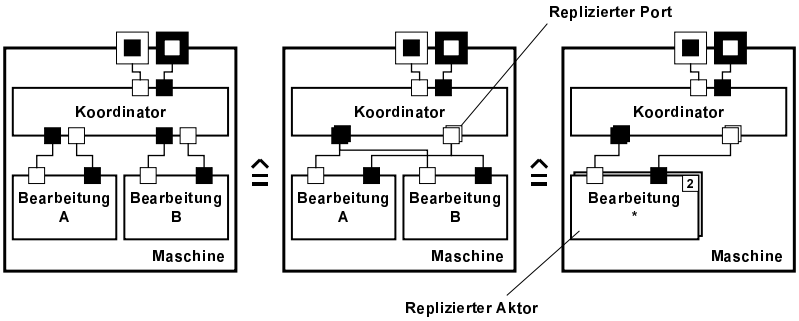

Bild 4-15: Replikation von Ports und Aktoren in ROOM.

ROOM bietet damit eine Reihe interessanter Modellierungskonzepte für die Steuerungstechnik im Werkzeugmaschinenbau. Allerdings wird einerseits die Modellierung von Abläufen nicht explizit unterstützt. Andererseits müssen für eine spätere Verteilung auch hier entsprechende Schnittstellen in der funktionalen Modellierung explizit vorgesehen werden.

4.2.4 UML für reaktive Systeme (UML-RT)

Neuere Entwicklungen im Bereich der UML greifen die Entwicklung von Software für reaktive Systeme auf. Bei UML-RT (RT: Real Time) wurde das Konzept der Stereotypen (siehe Abschnitt 4.2.2) der UML dazu verwendet, die Elemente von ROOM zu modellieren. Dementsprechend stellt UML-RT eine Einbettung der Konzepte von ROOM in die UML dar.

Aktuelle Entwicklungswerkzeuge (z. B. Rational Rose-RT, *Rational 1999*) für UML-RT bilden darüber hinaus auch die Funktionalität von Implementierungswerkzeugen ab. Es sind für verschiedene Zielsysteme daran anschließbare ROOM Virtual Machines verfügbar, so dass mit einem derartigen Entwicklungswerkzeug die Entwicklung von reaktiven Systemen von der Spezifikation bis hin zur Implementierung und zum Test durchgängig in einem Tool durchgeführt werden kann.

4.3 Zusammenfassung und Auswahl

In den vorangegangenen Abschnitten wurden im Maschinenbau anwendbare, gebräuchliche Modellierungstechniken vorgestellt und ihre jeweiligen Einsatzmöglichkeiten kurz diskutiert. Zudem wurden die aktuellsten Modellierungstech-

niken aus dem Bereich der objektorientierten Modellierung vorgestellt. Wie bereits erläutert, soll im Rahmen dieser Arbeit eine Basismodellierungstechnik zum Einsatz kommen, die es ermöglicht, die im Maschinenbau gebräuchlichen, grafischen Modellierungstechniken abzubilden. Dies kann grundsätzlich auf verschiedene Art erreicht werden.

- Einerseits kann eine Modellierungstechnik zum Einsatz kommen, deren Mächtigkeit ausreicht, um alle Elemente der im Maschinenbau gebräuchlichen Modellierungstechniken in einfacher Weise abzubilden.

- Andererseits ist es möglich, eine Modellierungstechnik mit geringerer Mächtigkeit zu verwenden, indem komplexere Elemente der gebräuchlichen Modellierungstechniken jeweils aus mehreren Elementen der Basismodellierungstechnik nachgebildet werden.

Aufgrund der Forderung nach einer guten Diagnostizierbarkeit besteht die Notwendigkeit, das Geschehen in der Steuerungssoftware direkt visualisieren und manipulieren zu können. Bei der erstgenannten Möglichkeit wird jedes Element der spezifischen Modellierungstechnik genau auf ein Element der Basismodellierungstechnik abgebildet. Dadurch werden die für die Visualisierung notwendigen Umkehrabbildungen wie auch die zur Manipulation benötigten Zugriffsmechanismen erheblich vereinfacht. Folglich empfiehlt es sich hier auf die erstgenannte Möglichkeit zurückzugreifen..

Um die in der Praxis gebräuchlichen Modellierungstechniken abbilden zu können, kommen daher nur Petrinetze in Frage, da durch sie sowohl Zustandsgrafen als auch Kontrollstrukturen entsprechend Abschnitt 4.1 abbildbar sind. Damit stellt sich die Frage, welche der beschriebenen Netzklassen Verwendung finden soll. Zwar genügt für die Abbildung heute in der Praxis verbreiteter Modellierungstechniken das B/E-Netz, andererseits eröffnet die Verwendung von colorierten Petrinetzen beispielsweise für die Modellierung von Materialflussaufgaben wertvolle Spielräume. Da das B/E-Netz auch als ein vereinfachtes CPN betrachtet werden kann, ist mit CPN jeder im B/E-Netz darstellbare Sachverhalt ebenfalls modellierbar. Durch eine geschickte Modellierung der Systemsoftware verteilter, kooperativer Steuerungen kann zudem der für CPN typische, erhöhte Rechenzeitaufwand bei B/E-Sequenzen vermieden werden (siehe dazu Abschnitt 6.2.2).

Aufgrund dieser Überlegungen werden für die Konzeptdetaillierung und für die prototypische Realisierung der Systemsoftware colorierte Petrinetze als Basismodellierungstechnik festgelegt. Die Ausführungen der folgenden Kapitel beschränken sich allerdings aus Gründen der Einfachheit und Verständlichkeit der Darstellung im Wesentlichen auf den Sprachumfang der B/E-Netze. In Abschnitt

6.2.2 wird dann kurz auf die für die Abbildung von CPN benötigten Klassen eingegangen.

Durch diese Festlegungen scheiden die bisher bekannten objektorientierten Modellierungstechniken als alleinige Modellierungstechnik aus, da hier colorierte Petrinetze zur Modellierung von Abläufen nicht vorgesehen sind. Gleichwohl erscheint es aber denkbar und für die Zukunft auch lohnend, eine Erweiterung der UML mit Hilfe von Stereotypen, vergleichbar mit der Erweiterung zu UML-RT, zur Abbildung von Petrinetzen durchzuführen. Im Rahmen der Ziele dieser Arbeit ist eine derartige, formale Erweiterung jedoch nicht notwendig.

Ebenso ist die Übertragung objektorientierter Gedanken wie die Vererbung auf die im folgenden Kapitel zu detaillierende Modellierungstechnik grundsätzlich möglich, steht aber nicht im Mittelpunkt der Arbeit. Entsprechend sollen die Konzepte aus ROOM zur Abbildung von Komponenten durch Aktor-Klassen aufgrund ihrer guten Handhabbarkeit als gedankliche Leitlinie für das Konzept dienen.

Darüber hinaus wurde die UML für die Entwicklung der Systemsoftware im Rahmen der prototypischen Realisierung herangezogen. Folglich findet sie auch in Kapitel 6 bei der Erläuterung des vorgeschlagenen Designkonzepts der Systemsoftware Anwendung.

5 Konzept

Im vorliegenden Kapitel soll das Konzept für die verteilte, kooperative Steuerung maschinennaher Abläufe entwickelt werden. Dazu soll zunächst die Grundkonzeption für das Zusammenspiel der Komponenten einer verteilten, kooperativen Steuerungstechnik vorgestellt werden (Abschnitt 5.1).

Zur effizienten Entwicklung dezentraler Steuerungssysteme wird eine spezielle Entwicklungsmethode benötigt (siehe Abschnitt 3.2). Sie umfasst ein Vorgehen (Abschnitt 5.2), eine in Entwicklungssichten eingeteilte Basismodellierungstechnik (Abschnitt 5.3) und ein Konzept für die Planung der Softwareverteilung (Abschnitt 5.4).

Darüber hinaus werden in Abschnitt 5.5 die zur Unterstützung der Entwicklungsmethode beziehungsweise darauf aufbauender Entwicklungswerkzeuge benötigten Funktionen für die Systemsoftware der verteilten, kooperativen Steuerungsmodule behandelt.

5.1 Grundkonzeption

Die in Abschnitt 3.1 analysierten Potenziale verteilter, kooperativer Steuerungen sollen unter Berücksichtigung der in Abschnitt 3.2 erarbeiteten Anforderungen durch das Zusammenspiel der in Bild 5-1 dargestellten Komponenten nutzbar gemacht werden.

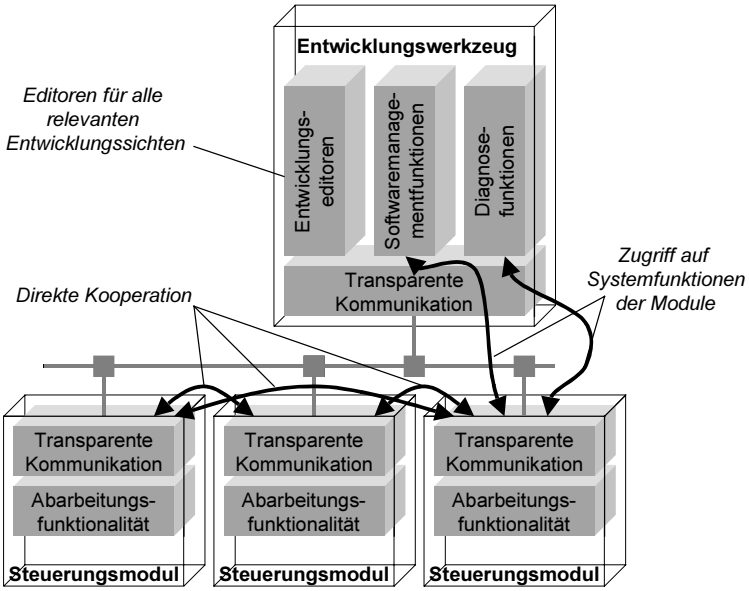

Bild 5-1: Zusammenspiel der Komponenten der verteilten, kooperativen Steuerungstechnik

Für die relevanten Entwicklungssichten werden in einem Entwicklungswerkzeug jeweils spezialisierte Entwicklungseditoren vorgesehen, mit deren Hilfe die Steuerungssoftware spezifiziert und programmiert wird. Dabei wird auf Funktionen der Systemsoftware zurückgegriffen, die eine Abarbeitung der verteilten Steuerungsfunktionen und die dazu benötigte Kommunikation und Synchronisation unterstützen. Diese Funktionen werden durch die Spezifikation der Steuerungssoftware in den Editoren implizit parametrisiert. Weitere Werkzeugfunktionen dienen dem Softwaremanagement und der Diagnose.

Die Abarbeitung der programmierten Funktionalität geschieht durch die verteilten, kooperativen Steuerungsmodule, die durch heute im Werkzeugmaschinenbau übliche Feldbusse wie beispielsweise Profibus (*DIN EN 50170-2 1996, Bender 1990*) oder aber auch Standardkommunikationssysteme wie Ethernet (*Furrer 1998*) vernetzt sind. Einen Überblick über entsprechende Bussysteme geben z. B. *Färber (1994)* oder *Schnell (1999)*. Die Steuerungsmodule stellen Funktionalität zur Abarbeitung der Steuerungssoftware, zur transparenten Kommunikation sowie zum Softwaremanagement und zur Diagnose zur Verfügung, auf die wieder-

um das Entwicklungswerkzeug aufsetzt. Diese Funktionen werden in Abschnitt 5.3 weiter detailliert.

5.2 Vorgehen

Hinsichtlich der Vorgehensweise wurde in Abschnitt 3.2.1 gefordert, eine weitreichende Parallelisierung zwischen Software- und Mechanikentwicklung zu ermöglichen. Dazu muss frühzeitig damit begonnen werden können, die geforderte Funktionalität der Software möglichst detailliert zu modellieren. Die Modellierung der Funktionen muss dabei ohne Kenntnis der Topologie der Steuerungshardware allein aus den Funktionsanforderungen möglich sein, um eine spätere Festlegung der Topologie durch die mechanische und die installationstechnische Modularisierung zu ermöglichen (siehe auch Abschnitt 2.1). Daraus lässt sich ableiten, dass das Vorgehensmodell die Festlegung der Verteilung der Software erst als dritten, unabhängigen Schritt nach der Modellierung der Funktionalität sowie der Modellierung der Steuerungstopologie vorsehen muss. Das sich daraus ergebende Vorgehen ist in Bild 5-2 dargestellt (siehe auch *Reinhart & Meier 2000)*.

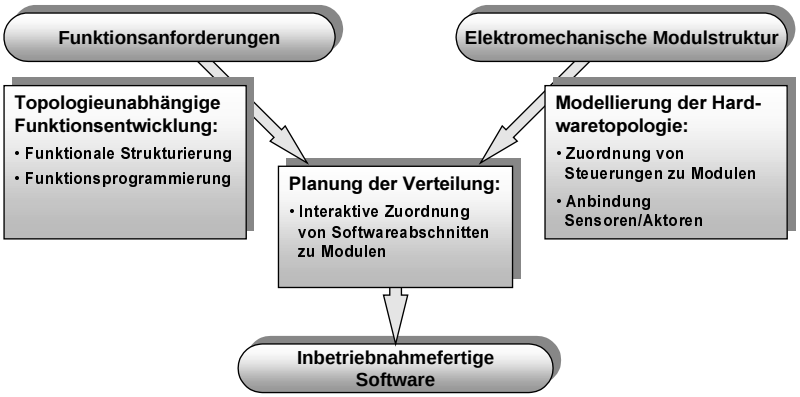

Bild 5-2: Vorgehen zur Softwareentwicklung für verteilte, kooperative Steuerungssysteme

Entsprechend diesem Vorgehen wird die Entwicklung der Steuerungsfunktionalität zum frühestmöglichen Zeitpunkt in der Gesamtentwicklung der Maschine durchgeführt. Dies findet also nach der in Zusammenarbeit zwischen mechanischer Konstruktion und Softwareentwicklung erarbeiteten Definition der Anfor-

derungen an die Steuerungsfunktionalität statt. Darauf aufbauend werden in der **topologieunabhängigen Funktionsentwicklung** Funktionsbereiche und Funktionsschnittstellen definiert und anschließend die Funktionen ausprogrammiert. Parallel dazu wird auf Basis der durch die mechanische Konstruktion und die Installationstechnik bestimmten, elektromechanischen Modularisierung die **Modellierung der Hardwaretopologie** durchgeführt. Dabei werden den Modulen oder Modulgruppen Steuerungen zugeordnet und durch Bussysteme verbunden. Den Steuerungen werden dabei auch die Sensoren und Aktoren zugeordnet. Erst im Anschluss an diese beiden Schritte wird die **Planung der Verteilung** der Steuerungssoftware auf die Steuerungshardware durchgeführt. Dies geschieht unter Berücksichtigung der E/A-Verteilung, der geforderten Reaktionszeiten und der Auslastung der Steuerungen im Vergleich zu deren Leistungsdaten. Dabei zerfallen die Koordinationsfunktionen in lokale Teilfunktionen, die sich selbsttätig mittels der Funktionen der transparenten Kommunikation koordinieren und synchronisieren.

5.3 Modellierungstechnik

In den folgenden Abschnitten wird die zur Modellierung verteilter, kooperativer Steuerungen eingesetzte Basismodellierungstechnik vorgestellt. Zunächst wird dazu auf die Modellierung von Abläufen, die funktionale Sicht, eingegangen. Diese Sicht wird dann schrittweise um die Modellierung von Prozessanbindungs- und Hilfsfunktionen sowie von Reaktionszeitanforderungen ergänzt und auf dieser Basis ein Konzept zur Modellierung von Modulschnittstellen eingeführt. Abschließend wird dann auf die topologische sowie die distributive Sicht eingegangen.

5.3.1 Entwicklungssichten und Entwicklungswerkzeug

Für die Modellierungstechnik für verteilte, kooperative Steuerungssysteme lassen sich dem Vorgehen entsprechend drei wesentliche Entwicklungssichten definieren. Es handelt sich dabei um die Entwicklungssichten der Funktionalität, der Steuerungstopologie sowie der Verteilung, analog zu den wesentlichen Schritten des in Abschnitt 5.2 vorgestellten Vorgehens. Die Arbeit mit mehreren solchen Entwicklungssichten kann durch ein entsprechendes, multisichtorientiertes Entwicklungswerkzeug unterstützt werden (Bild 5-3). Dabei gewährleistet das Entwicklungswerkzeug die Konsistenz der in den jeweiligen Sichten modellierten Sachverhalte durch die Abbildung in einem gemeinsamen Datenmodell, das zusätzlich auch für die Softwaremanagement- und Diagnosefunktionen benötigt wird.

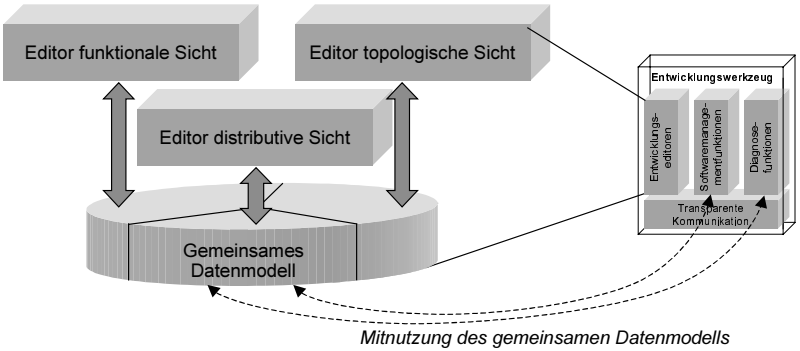

Bild 5-3: Sichtenspezifische Editoren des Entwicklungswerkzeugs

Die **funktionale Sicht** der Modellierungstechnik dient dazu, die Funktionen des Steuerungssystems in einer weitgehend grafischen Modellierungssprache zu strukturieren und auszuprogrammieren. Innerhalb dieser Sicht darf die spätere Verteilung der Software keine Rolle spielen. Dementsprechend dürfen hier auch keine Funktionen zur Abwicklung der verteilungsbedingten Kommunikation und Synchronisation erscheinen. Die Software wird in dieser Sicht ausschließlich nach funktionalen Gesichtspunkten strukturiert, unterscheidet sich in diesem Aspekt also kaum von bekannten, für rein zentrale Steuerungen ausgelegten Modellierungstechniken.

Darüber hinaus sind innerhalb der funktionalen Sicht auch die durch die zu realisierende Maschinenfunktion gestellten Anforderungen an die Reaktionszeit des Steuerungssystems zu modellieren. Dadurch werden einerseits solche Anforderungen dokumentiert und andererseits können später zielgerichtet Maßnahmen zu deren Einhaltung, beispielsweise durch eine entsprechende Verteilung, ergriffen werden. Die Modellierung geforderter Reaktionszeiten ist bisher außer in einfachen Tabellen, die die jeweilige Verzögerung eines Ausgangssignals gegenüber der auslösenden Veränderung des Eingangssignals angibt, nicht üblich (*Balzert 1996*).

In der Entwicklungssicht zur Festlegung der Steuerungstopologie, der **topologischen Sicht**, sind die verwendeten Steuerungen, die Struktur des Bussystems sowie die Verteilung der Ein- und Ausgänge abzubilden. Zusätzlich werden in dieser Sicht Informationen über die Leistungsfähigkeit der Steuerungsmodule und der Bussysteme benötigt, um eine hinsichtlich der Reaktionsgeschwindigkeit zielgerichtete Softwareverteilung zu ermöglichen. Solche Informationen können durch ein Entwicklungswerkzeug gewissermaßen als Katalogdaten zu den Steue-

rungsmodulen und Bussystemen vorgehalten werden, so dass hier kein zusätzlicher Modellierungsaufwand entsteht.

Die Entwicklungssicht zur Planung der Softwareverteilung, die **distributive Sicht**, dient schließlich zur Festlegung bzw. zur Darstellung der Verteilung der Steuerungssoftware auf die Steuerungshardware. Dabei werden die grafisch spezifizierten Softwareelemente der funktionalen Sicht den Steuerungsmodulen der topologischen Sicht zugeordnet.

5.3.2 Funktionale Sicht

Wie bereits in Abschnitt 4.3 herausgearbeitet, sollen bei der Detaillierung der Modellierungstechnik für die funktionale Sicht colorierte Petrinetze zum Einsatz kommen. Zur Abgrenzung von den bisher dargestellten Netzen soll im Folgenden eine gegenüber Abschnitt 4.1.4 leicht modifizierte, grafische Darstellung verwendet werden. Die dabei verwendeten Basiselemente sind anhand eines einfachen Modellierungsbeispiels in Bild 5-4 zusammengestellt.

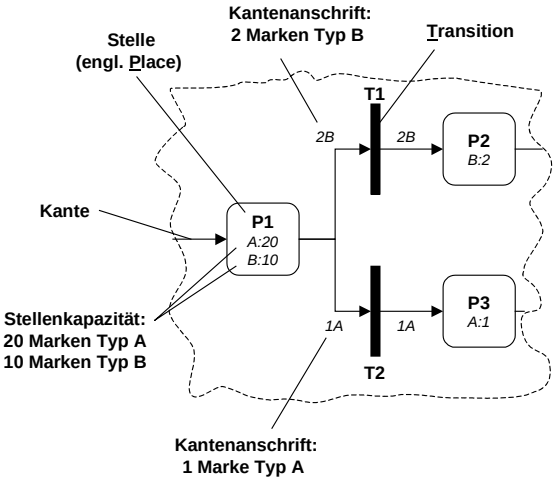

Bild 5-4: Basiselemente der Modellierungstechnik.

Wie bereits in Abschnitt 4.3 erläutert, sollen die folgenden Ausführungen für eine einfachere und schneller verständliche Darstellung auf den reduzierten Umfang der B/E-Netze beschränkt werden. Die folgenden Ausführungen gelten trotz

dieser Einschränkung auch für colorierte Petrinetze, da jederzeit entsprechende Kantenanschriften oder Stellenkapazitäten hinzugefügt werden könnten. Um maschinennahe Abläufe im Sinne dieser Arbeit vollständig und effizient modellieren zu können, werden im Folgenden sukzessive Erweiterungen und spezifische Modellierungskonstrukte eingeführt.

Zunächst sind dabei Vereinbarungen zur Modellierung von Schreib- und Leseoperationen für den Eingangs- und Ausgangsbereich der Steuerung zu treffen. Analog zu Schrittketten- und einigen Zustandsgrafentechniken wird dabei für die Stelle ein **Entry**- und ein **Exit-Bereich** definiert. In diesen Bereichen werden Anweisungen zum Lesen von Sensorsignalen, Variablen (s. u.) oder Timern (s. u.), zu deren Verknüpfung und zur Ausgabe von Konstanten oder Verknüpfungsergebnissen Aktorsignale, Variable oder Timertriggerimpulse eingetragen. Bild 5-5 zeigt eine mögliche, grafische Darstellung dieser Bereiche. Die darin enthaltenen Anweisungen können in jeder geeigneten Programmiersprache eingetragen werden – also wie hier in Pseudocode oder aber auch beispielsweise in Anweisungsliste (AWL, siehe Abschnitt 4.1.1).

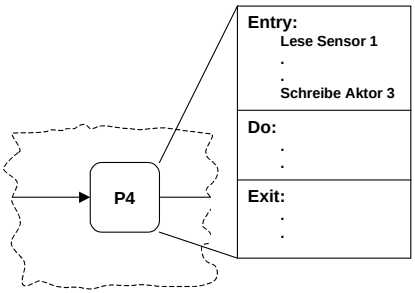

Bild 5-5: Abbildung von Aktionen in der Basismodellierungstechnik in Entry-, Do- und Exit-Sektionen der Stellen.

Der Entry-Bereich wird dabei beim Eintreten einer Marke in die Stelle durchlaufen, der Exit-Bereich beim Verlassen der Stelle durch eine Marke. Zusätzlich wird ein **Do-Bereich** definiert, in dem eine zyklische Aktion definiert werden kann. Dadurch könnte beispielsweise ein Regelalgorithmus ein- beziehungsweise ausgeschaltet werden, sobald die Stelle markiert beziehungsweise demarkiert wird. Handelt es sich bei der Stelle um eine komplexe Stelle mit CPN-Funktionalität, so muss hier zusätzlich definiert werden, welche Markierungssituation oder Markierungsänderung zum Durchlaufen dieser Bereiche führt. Hier

können je nach Anwendungsfall verschiedene Standardeinstellungen definiert werden. Darauf weiter einzugehen würde hier jedoch zu weit führen.

Als eine weitere Ergänzung soll das Konstrukt einer Transitionsbedingungsstelle **TEP** (Transition Expression Place) eingeführt werden. Eine TEP ist aus Sicht der Petrinetz-Theorie keine Stelle im eigentlichen Sinne, sondern dient lediglich zur Feststellung des aktuellen Wahrheitswerts der Transitionsbedingung der angeschlossenen Transition. Dementsprechend verändert die TEP ihre Markierung entsprechend des booleschen Ausdrucks, den sie repräsentiert. Das Schalten der angeschlossenen Transition bewirkt folglich auch nicht die Demarkierung der TEP, ebenso wie die TEP ohne das Schalten einer Transition ihre Markierung ändert. Bild 5-6 zeigt die grafische Notation der TEP. Der boolesche Ausdruck der Transitionsbedingung kann dabei beispielsweise in Kontaktplannotation (KOP, vgl. Abschnitt 4.1.1) dargestellt werden, was, wie in Abschnitt 4.1.4 bereits erläutert, im Diagnosefall zu einer grafisch leicht erfassbaren Visualisierung führt.

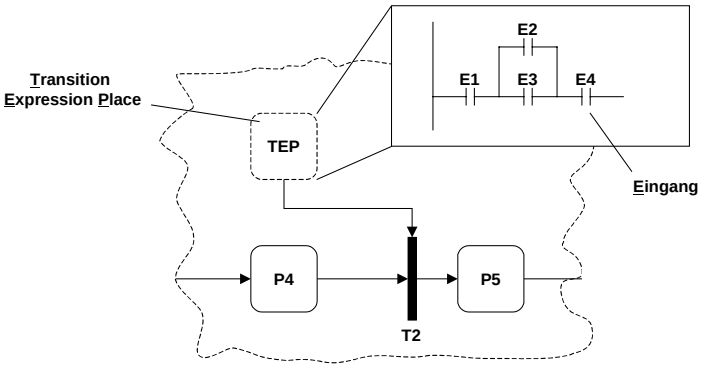

Bild 5-6: Ergänzung der Basismodellierungstechnik um eine Transitionsbedingungsstelle und Darstellung einer Transitionsbedingung beispielsweise in Kontaktplannotation.

Dieses Konstrukt wird aus zwei Gründen eingeführt. Einerseits wird dadurch die grafische Visualisierung des Netzzustandes einfacher verständlich. Das Netz schaltet damit grundsätzlich und ohne Verzögerung weiter, wenn alle Stellen – einschließlich der TEP – im Vorbereich ausreichend markiert sind und alle Stellen im Nachbereich ausreichend Kapazität verfügen. Andererseits wird es damit möglich, die Funktionalität der Systemsoftware einfacher zu strukturieren, was jedoch erst in Abschnitt 6.1.1 eingehender erläutert wird.

Zur Verbesserung der Strukturierbarkeit der modellierten Steuerungssoftware wird das Konstrukt der **funktionalen Module** eingeführt. Damit können Teilnetze, die zur Steuerung einzelner, mechanischer Komponenten benötigt werden, abgegrenzt werden, wie dies in Bild 5-7 dargestellt ist. In diesen Modulen werden weiterhin die benötigten Eingangs- und Ausgangssignale symbolisch in Form von sogenannten **Signalports** (siehe auch hierzu Bild 5-7) angegeben. Dabei werden Signalports für lesende Zugriffe mit einem moduleinwärts zeigenden, gefüllten Halbkreis dargestellt, während für schreibende Zugriffe dieser Halbkreis modulauswärts zeigt. Die Signalports können dann später in einem entsprechenden Entwicklungswerkzeug mit den real vorhandenen Ein- und Ausgängen der topologischen Sicht sowie entsprechenden globalen Variablen und Timern verknüpft werden. Die Signalports werden innerhalb des funktionalen Moduls durch gestrichelte Linien mit den zugreifenden Stellen verbunden. Diese Linien können allerdings in der funktionalen Sicht auch weggelassen werden, da sie genaugenommen nur eine grafische Repräsentation der in den Entry-, Do- und Exitfunktionen ohnehin vorliegenden Informationen darstellen.

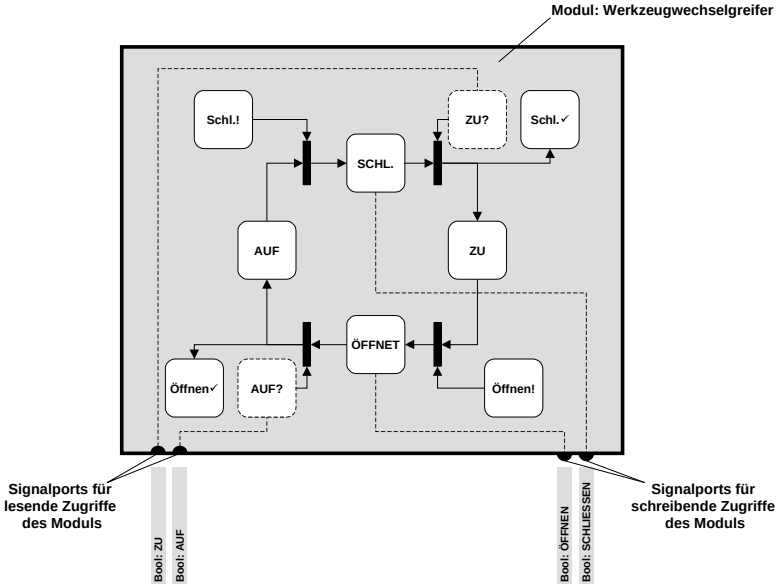

Bild 5-7: Bildung von Modulen in der Basismodellierungstechnik der funktionalen Sicht und Einführung von Signalports.

Um solche funktionalen Module miteinander verbinden zu können, müssen noch spezielle Konstrukte zur Modellierung von Schnittstellen ergänzt werden. Sie stellen gewissermaßen Anschlussstellen für die Verlängerung von Kanten über Modulschnittstellen dar. Dementsprechend sind sie nur ein grafisches Hilfsmittel zur Vereinfachung der Modellierung, nicht aber eine substantielle Erweiterung der Petrinetznotation. Diese Konstrukte sollen künftig als **Markenports** bezeichnet und durch Pfeilspitzen an den Modulrändern dargestellt werden, wie dies Bild 5-8 zeigt. Sie sind in etwa vergleichbar mit den Ports aus ROOM, wobei die zugehörige Kantenanschrift der Protokolldefinition entspricht (vgl. Abschnitt 4.2.3). Die Pfeilspitzen weisen immer in Richtung des über die Kante abzuwickelnden Markenflusses. Markenports auf Seiten des beauftragenden funktionalen Moduls (Client) sind gefüllt und auf Seiten des beauftragten funktionalen Moduls (Server) nicht gefüllt dargestellt. Als Modellierungsregel müssen hier immer ausgefüllte Pfeilspitzen mit nicht ausgefüllten Pfeilspitzen verbunden werden. Ebenso müssen immer moduleinwärts gerichtete Pfeilspitzen mit modulauswärts gerichteten Pfeilspitzen verbunden werden.

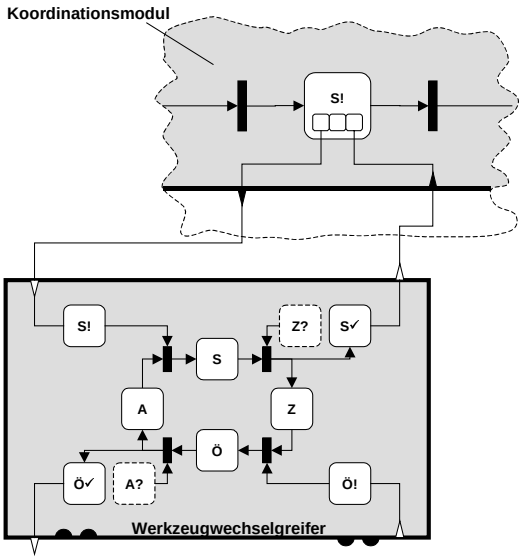

Bild 5-8: Verknüpfung von funktionalen Modulen in der Basismodellierungstechnik der funktionalen Sicht am Beispiel des Werkzeugwechselgreifers aus Bild 5-7.

Um eine einfache Möglichkeit zur Modellierung der Beauftragung eines funktional unterlagerten Moduls zu schaffen, wurde als grafische Vereinfachung eine sogenannte **Client-Stelle** geschaffen. Diese stellt aber ebenfalls lediglich als eine grafische Vereinfachung dar, deren Bedeutung in Bild 5-9 dargestellt ist.

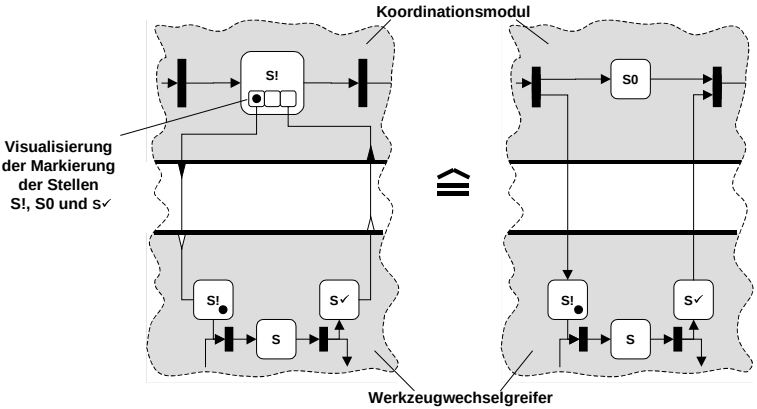

Bild 5-9: Interne Abbildung der Clientstelle auf bereits bekannte Elemente colorierter Petrinetze.

In der in Abschnitt 4.2.3 dargestellten Modellierungstechnik ROOM ist mit der Replizierbarkeit von Ports und Komponenten ein weiteres, für den praktischen Einsatz sehr gut geeignetes Modellierungskonstrukt enthalten, welches ebenfalls auf die Basismodellierungstechnik für verteilte, kooperative Steuerungen übertragen werden soll. Bild 5-10 zeigt die **Replizierbarkeit von Markenports** und funktionalen Modulen, die wiederum lediglich eine grafische Vereinfachung eines komplexeren Sachverhalts in der CPN-Notation darstellt. Im Gegensatz zu ROOM wird hier allerdings keine dynamische Veränderung der Zahl der Replikationen zur Laufzeit des Steuerungsprogramms vorgesehen. Dies erscheint im Maschinenbau deshalb im Allgemeinen nicht relevant, da die Zahl der funktionalen Module meist durch die zur Laufzeit nicht veränderliche Zahl der real vorhandenen Maschinenkomponenten bestimmt ist.

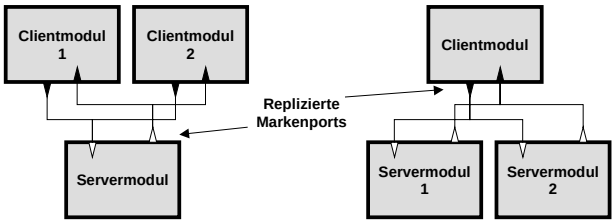

Bild 5-10: Konkurrierender Zweifachzugriff auf ein Servermodul und parallele Einbindung zweier Servermodule durch ein Clientmodul.

Zusätzlich zu diesen Modellierungselementen müssen zur Spezifikation der funktionalen Sicht auch **Reaktionszeitanforderungen** geeignet modelliert werden können. Dazu wird die Modellierungstechnik der funktionalen Sicht um ein entsprechendes Element erweitert. Danach kann für jeden beliebigen Markenpfad, dargestellt durch eine punktierte Linie, eine Reaktionszeitanforderung in das Petrinetz eingetragen werden, wie dies in Bild 5-11 dargestellt ist.

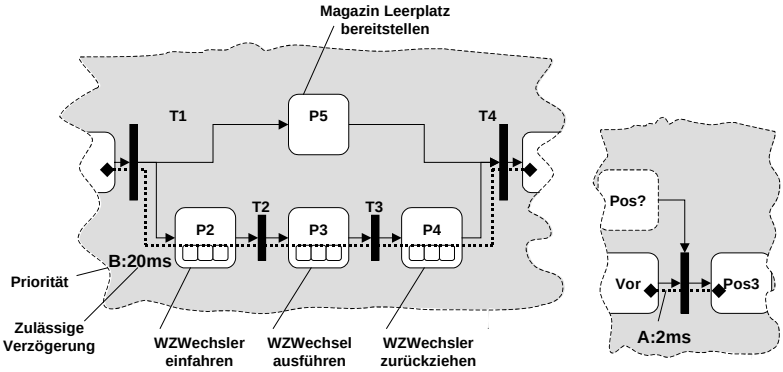

Bild 5-11: Modellierung von Reaktionszeitanforderungen unterschiedlicher Priorität entlang von Pfaden in der funktionalen Sicht.

Die modellierten Reaktionszeitanforderungen beschreiben die zulässige Summe von Verzögerungszeiten, die sich aufgrund von Kommunikations- und Prozessorzeiten entlang des Pfades bei schaltfähigen Transitionen einschließlich der dabei notwendigen Zeiten für den Zugriff auf Ein- und Ausgangssignale für den Markenfluss ergeben darf. Dabei kann die Reaktionszeitanforderung in Prioritätsklassen eingestuft werden. Bei der höchsten Priorität (Priorität A) muss diese

kritische Reaktionszeit unabdingbar eingehalten werden, andere Prioritätsstufen können anwendungsspezifisch definiert werden. Sie sind dann nicht mehr unbedingte Muss-Anforderungen, sondern dienen ausschließlich als Eingangsinformation für eine möglichst anforderungsgerechte Verteilung und als Bewertungsreferenz für Zeitmessungen in der Inbetriebnahme.

5.3.3 Topologische Sicht

Die Modellierung der Steuerungstopologie umfasst die Abbildung der Steuerungsmodule, die Zuordnung der Sensorik und Aktorik zu den Steuerungsmodulen sowie die Verbindungen der Steuerungssysteme untereinander mittels Bussystemen. Dabei müssen beliebige Busstrukturen möglich sein. Bild 5-12 zeigt dies anhand einer einfachen Beispieltopologie, in der die Module Werkzeugwechsler (WZW), Bearbeitungseinheit (BAE) und Werkzeugmagazin (WZMag) mittels zweier Bussysteme miteinander verbunden sind, um in Kooperation die Werkzeugwechselfunktion eines Fräsbearbeitungszentrums abzubilden. Zusätzlich ist in diesem Steuerungssystem ein Bedienfeld, auf dem beispielsweise ein Visualisierungswerkzeug laufen kann, vorgesehen.

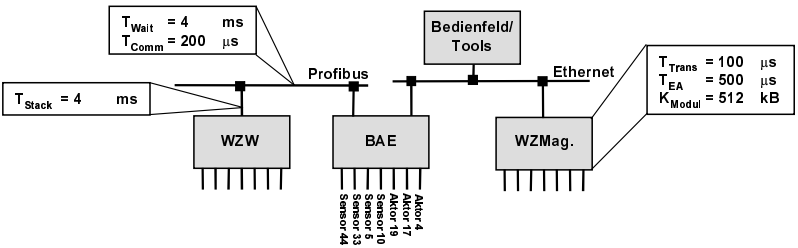

Bild 5-12: Modellierungstechnik der topologischen Sicht am Beispiel der am Werkzeugwechsel beteiligten Module eines Bearbeitungszentrums.

Zusätzlich zur Topologieinformation wird mit einfachen Kennzahlen die Leistungsfähigkeit der Steuerungen und der Bussysteme modelliert. Dadurch soll es ermöglicht werden, bei der späteren Verteilung der Steuerungssoftware auf die Steuerungsmodule eine hinsichtlich der Reaktionszeiten sowie der Steuerungsauslastung anforderungsgerechte Lösung zu finden. Deshalb muss anhand der vorgesehenen Kennzahlen einerseits eine grobe Abschätzung erreichbarer Kommunikationszeiten insbesondere für kritische Reaktionszeitanforderungen möglich sein. Andererseits muss die Auslastung der Steuerungsmodule überschlägig bestimmt werden können.

Hinsichtlich der erreichbaren Reaktionszeiten werden zur groben Abschätzung durch entsprechende Funktionen eines Entwicklungswerkzeugs die sechs folgenden Kennzahlen vorgeschlagen:

- **Stackzeit (T_{Stack}):** Die Stackzeit ist diejenige Verzögerungszeit, die ein hochpriores Telegramm (ein über das Bussystem zu versendendes Datenpaket) benötigt, um von der Anwendungsebene über den Kommunikationsstack bis zum möglichen, physikalischen Versand auf dem Bussystem zu gelangen beziehungsweise vom Bussystem zur Anwendungsebene transferiert zu werden. Diese Zeiten sind zwar nicht identisch, da aber immer beide Verzögerungen anfallen, kann hier vereinfachend der Durchschnittswert beider Zeiten verwendet werden. Entsprechend fällt die Stackzeit pro Übertragungsstrecke zweifach an.

- **Maximale Wartezeit am Bus (T_{Wait}):** Ein versandfertiges Telegramm kann abhängig vom jeweiligen Buszugriffsverfahren des jeweils verwendeten Bussystems unter Umständen nicht sofort versendet werden. Beispielsweise kann bei Bussystemen, die das Schreibrecht auf dem Bus mit dem Tokenumlaufverfahren (z. B. Profibus, s. *Bender 1990*) vergeben, die Token Rotation Time eingesetzt werden.

- **Busbelegung pro Telegramm (T_{Comm}):** Zum Versand eines Telegramms wird das Bussystem für eine von der Telegrammlänge und der Übertragungsrate des Bussystems abhängige Zeit belegt. Zur Bestimmung dieser Zeit kann von einer für den Markentransfer durchschnittlichen Telegrammlänge ausgegangen werden. Bei den heute üblichen Übertragungsraten schneller Bussysteme hat diese Verzögerungszeit allerdings meist einen verhältnismäßig geringen Einfluss.

- **Transitionszeit (T_{Trans}):** Weitere Verzögerungen entstehen durch die Dauer des Schaltvorgangs von Transitionen. Dabei wird diejenige Zeit angegeben, die zum Transfer der Marken des Vorbereichs in den Nachbereich benötigt wird. Diese Zeit hängt genaugenommen von der Anzahl der Kanten und der Marken sowie den zu berücksichtigenden Farbgruppen ab. Eine solche Betrachtung würde jedoch zu weit führen. Statt dessen soll hier ein Richtwert genügen, der sich an einer Transition geringer bis mittlerer Komplexität orientiert. Dies ist sinnvoll, da kritische Reaktionszeitanforderungen meist auf Synchronisationsaufgaben, weniger jedoch auf Materialflussfunktionen angewendet werden.

- **E/A-Zugriffszeit (T_{EA}):** Zusätzlich muss bei der Abschätzung der Reaktionszeit die Verzögerung bei der Ein- bzw. Ausgabe von Prozesssignalen berück-

sichtigt werden. Auch hier soll zur Vereinfachung nicht zwischen Ein- und Ausgangssignalen unterschieden werden.

- **Speicherkapazität (K_{Modul}):** Zur Abschätzung der Auslastung der Steuerungsmodule wird lediglich die Speicherauslastung als Maß herangezogen. Daher genügt es, hierfür die für die Abbildung der Petrinetze zur Verfügung stehende Speicherkapazität der Steuerungsmodule anzugeben.

Es sei hier darauf hingewiesen, dass eine präzise Berechnung der Reaktionszeiten in bestimmten Steuerungssituationen anhand der hier vorgeschlagenen Kennzahlen weder möglich ist, noch angestrebt wird. Dazu müsste das Verhalten des Steuerungssystems und der gesteuerten Maschine ebenfalls detailliert beschrieben werden. Dies erscheint aus Aufwandsgründen aber nicht sinnvoll. Auch ist es damit nicht möglich, sichere Obergrenzen für Reaktionszeiten bei Worst-Case-Annahmen zu ermitteln. Die Schätzfunktion, die anhand dieser Kennzahlen in Abschnitt 5.4.3 vorgeschlagen wird, dient zum Erkennen von Verteilungen, die zur Erfüllung der Reaktionszeitanforderungen ungeeignet sind, was eine wesentliche Unterstützung in der Entwicklung verteilter, kooperativer Steuerungssysteme darstellt.

5.3.4 Distributive Sicht

Durch die Modellierung der distributiven Sicht wird die Verteilung der in der funktionalen Sicht festgelegten Stellen und Transitionen auf die einzelnen, in der topologischen Sicht modellierten Steuerungsmodule festgelegt, wie dies in Abschnitt 5.4 beschrieben wird. Die distributive Sicht muss es dem Steuerungsentwickler dabei ermöglichen, eine geeignete Verteilung zu modellieren und bestehende Verteilungen zu beurteilen. Dazu muss für den Steuerungsentwickler qualitativ erkennbar sein, wie sich eine Verteilungsentscheidung auf die Eigenschaften des Steuerungssystems und insbesondere die Reaktionszeiten auswirkt. Deshalb müssen Sachverhalte, die zu Kommunikationserfordernissen führen, grafisch klar in der distributiven Sicht erkennbar sein.

Zunächst aber müssen durch den Steuerungsentwickler die Verbindungen zwischen den Signalports der funktionalen Sicht an die in der topologischen Sicht modellierten Sensoren und Aktoren beziehungsweise an globale Variablen oder Timer angebunden werden, wie dies in Bild 5-13 dargestellt ist. Dabei erstreckt sich das funktionale Modul über alle an der Funktionalität in Form von Ein- und Ausgängen, Variablen und Timern sowie Markenports unterlagerter, funktionaler Module beteiligten Steuerungsmodule. Verbindungen zwischen diesen Elementen und Signal- oder Markenports des aktuell zur Verteilung anstehenden, funktionalen Moduls müssen dabei stets innerhalb eines Steuerungsmoduls verlaufen.

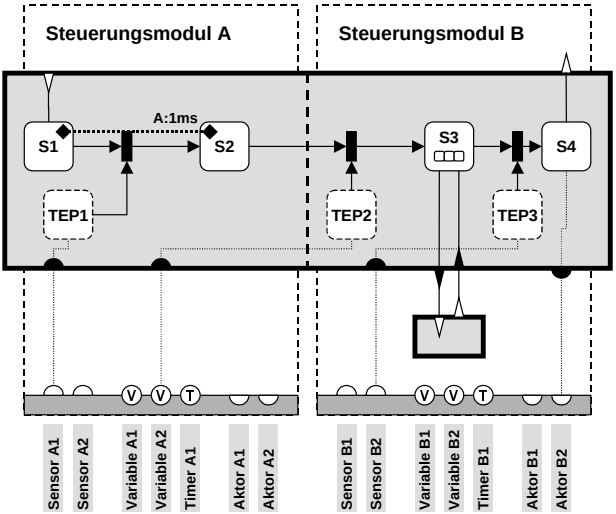

Bild 5-13: Elemente der Modellierung in der distributiven Sicht.

Durch den Zusammenhang zwischen Signalports und Stellen kann damit eine sinnvolle Verteilung gefunden werden. Dazu muss die Zahl der Verknüpfungen zwischen Elementen, die auf verschiedenen Steuerungsmodulen lokalisiert sind, möglichst gering gehalten werden. Dies gilt neben den Verbindungen zwischen Stellen und Signalports auch für die Kanten des verteilten Petrinetzes, die schließlich ebenfalls über die Kommunikation abgebildet werden müssen, falls sie Netzknoten auf unterschiedlichen Steuerungsmodulen miteinander verbinden. Dabei müssen die in der funktionalen Sicht modellierten Reaktionszeitanforderungen ebenfalls mit berücksichtigt werden (siehe auch hierzu Bild 5-13). Auf die Problematik der Planung der Softwareverteilung wird in Abschnitt 5.4 noch näher eingegangen.

Für ein detaillierteres Verständnis der distributiven Sicht wurde in Bild 5-15 auf ein Beispiel eines Werkstücktransfers durch eine einfache Fertigungsmaschine entsprechend Bild 5-14 zurück gegriffen. Wird in diesem Beispiel in Position 1 ein Werkstück eingelegt, so wird dieses durch eine Schubstange jeweils eine Position weiter gefördert, bis es in Position 4 wieder entnommen werden kann. Werkstücke vom Typ A beziehungsweise B werden abhängig von ihrem Typ in der Bearbeitungsstation A beziehungsweise B bearbeitet. Durch die jeweils andere Bearbeitungsstation wird das Werkstück lediglich durchgetaktet. Für die Abbildung dieser Steuerungsaufgabe sind dabei drei Steuerungsmodule vorgesehen.

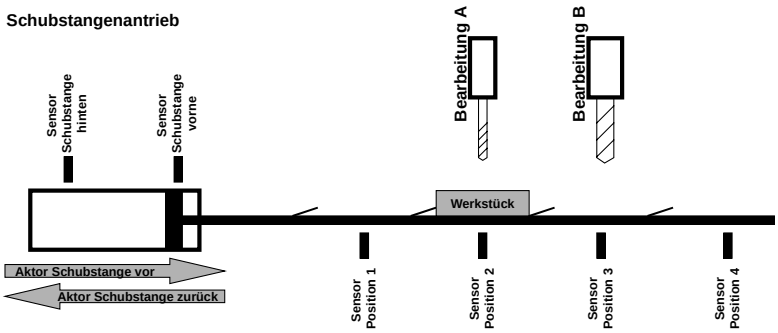

Bild 5-14: Modellierung von Reaktionszeitanforderungen unterschiedlicher Priorität entlang von Pfaden in der funktionalen Sicht.

Das entsprechende Steuerungsnetz ist in Bild 5-15 bereits in der distributiven Sicht dargestellt. Die Werkstücktypen A und B sind dabei durch entsprechende, colorierte Marken vom Typ A oder B abgebildet, die durch die Kantenanschriften entsprechend durch das Netz geleitet werden. Dabei wird die Schubstangensteuerung durch den Transfer von Marken jeweils zu einem Vorwärtstakt angestoßen. Dabei ist für den Markenpfad von der Stelle „Vor“ zur Stelle „Vorne“ („Vo.“) eine Reaktionszeitanforderung der Priorität A von 1ms angegeben. Dadurch soll zum Ausdruck gebracht werden, dass beim Ansprechen des Sensors „vorne“ innerhalb einer Millisekunde das Aktorsignal für die Vorwärtsbewegung zurückgesetzt werden muss, da sonst die für die Positionierung des Werkstücks zulässige Toleranz überschritten würde. Damit steht bei der Verwendung heute üblicher Bussysteme fest, dass alle an dieser Schaltaktion beteiligten Elemente in einem Steuerungsmodul lokalisiert werden müssen, da diese Reaktionszeitanforderung nur ohne eine Kommunikation über ein Bussystem erreichbar ist.

Für die Arbeit mit der distributiven Sicht wird in der Regel die Information über Kantenanschriften und Stellenkapazitäten nicht benötigt. Dementsprechend empfiehlt es sich, in einem entsprechenden Entwicklungswerkzeug diese Informationen nur bei Bedarf einzublenden. Im obigen Beispiel sind diese Informationen lediglich angegeben, um die Funktionalität des Netzes nachvollziehbar zu machen. Für die Verteilung dagegen werden sie hier nicht benötigt.

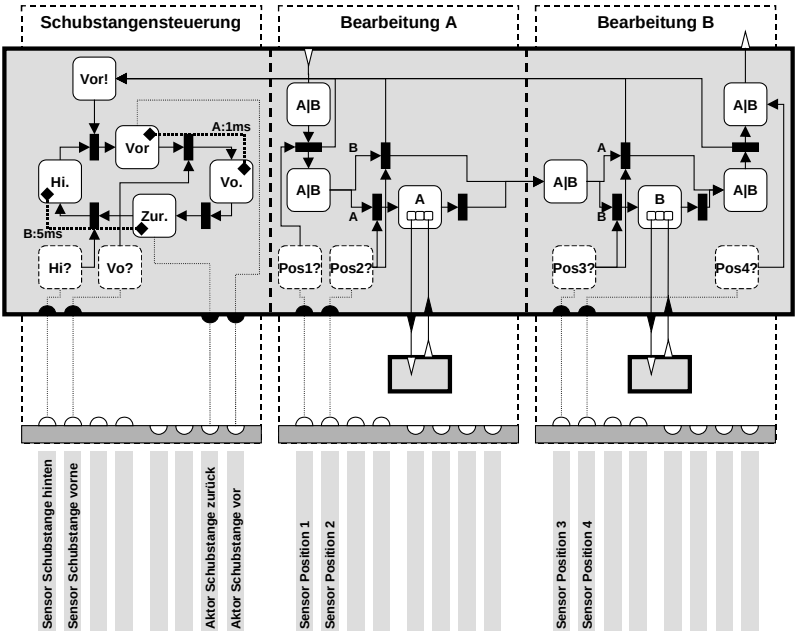

Bild 5-15: Modellierung von Reaktionszeitanforderungen unterschiedlicher Priorität entlang von Pfaden in der funktionalen Sicht.

5.4 Planung der Softwareverteilung

Im vorliegenden Abschnitt wird die Planung der Verteilung der Softwareelemente (Stellen und Transitionen) aus der funktionalen Sicht auf die Module der topologischen Sicht behandelt. Sie findet in der distributiven Sicht statt. Dazu wird zunächst die Konzeption des Verteilungsverfahrens erläutert und zusätzlich zur Möglichkeit der interaktiven Verteilung durch den Softwareentwickler ein automatisches, regelbasiertes Verteilungsverfahren zu entwickelt. Darüber hinaus wird Schätzfunktion zur Bewertung von Verteilungen vorgestellt.

5.4.1 Konzeption des Verteilungsverfahrens

Für den Vorgehensschritt der Verteilung der Softwareelemente auf die Steuerungshardware bestehen eine Reihe von grundsätzlichen Möglichkeiten, die im folgenden kurz aufgelistet werden:

- **Interaktive Verteilung durch den Softwareentwickler:** Nach diesem Verfahren werden die Softwareelemente durch den Softwareentwickler mittels eines grafischen Verfahrens (bekannt unter dem Stichwort „Drag & Drop“) auf die Steuerungsmodule verteilt. Dabei kann davon ausgegangen werden, dass in der überwiegenden Zahl der Fälle der Steuerungsentwickler hierzu ohnehin konkrete Vorstellungen hat (*VDW 2000*). Diese Vorstellungen basieren auf seiner Erfahrung und auf logischen Zusammenhängen. So wird ein Entwickler basierend auf seiner Erfahrung zeitkritische Softwareabschnitte auf dem Steuerungsmodul zusammenfassen, das die angeschlossene Aktorik und Sensorik bedient.

- **Regelbasierte, automatische Verteilung:** Grundsätzlich lassen sich die Zusammenhänge, aufgrund derer ein Softwareentwickler bei der interaktiven Verteilung vorgehen würde, auch in einem regelbasierten Verteilungswerkzeug abbilden. Dazu müssen die in Abschnitt 5.3.1 aufgeführten Informationen über die Reaktionszeitanforderungen der funktionalen Sicht einerseits und die Leistungskennzahlen der topologischen Sicht andererseits in das regelbasierte Verfahren mit einbezogen werden. Sie dienen in diesem Fall über den Zweck der Dokumentation hinaus auch zur Festlegung einer möglichst anforderungsgerechten Verteilung.

- **Automatische Verteilung durch ein Optimierungsverfahren:** Da es sich bei der Ermittlung einer optimalen Verteilung prinzipiell um ein Optimierungsproblem handelt, ist auch der Einsatz eines mathematischen Optimierungsverfahrens denkbar. Hierzu müssen zusätzlich die Kriterien, nach denen eine solche Verteilung optimiert werden soll, in geeigneter Weise formuliert werden. Darüber hinaus muss das Verhalten der gesteuerten Maschine mit modelliert werden, um unter Berücksichtigung der Häufigkeit und der zeitlichen Abfolge der Ereignisse die Wirkung verschiedener Verteilungen auf die Optimierungskriterien ermitteln zu können.

Die oben aufgeführten Verfahren zur Festlegung einer Verteilung können darüber hinaus auch kombiniert werden. Zum Beispiel könnte durch eine regelbasierte Verteilung eine Ausgangsverteilung ermittelt werden, die anschließend durch ein Optimierungsverfahren noch optimiert wird.

Da bei der Nutzung verteilter, kooperativer Steuerungssysteme die Forderung nach einer effizienten Entwicklung besteht (siehe Abschnitt 3.2.1), erscheint insbesondere die Modellierung des Verhaltens der gesteuerten Maschine nur zum Zwecke der Ermittlung einer optimierten Verteilung nicht vertretbar. Eine automatische Verteilung durch ein Optimierungsverfahren kommt deshalb im Rahmen dieser Arbeit nicht in Betracht. Sollte zukünftig durch die zunehmende Ver-

breitung der Entwicklung mit Hilfe virtueller Prototypen (siehe z. B. *Osmers 1998, Tomaszunas 1998 oder Bender 1999*) eine solche Verhaltensmodellierung vorhanden oder leicht ableitbar sein, kann eine Verteilung durch Optimierungsverfahren jedoch sehr wohl in Betracht kommen.

Im weiteren Verlauf dieser Arbeit sollen daher die Verfahren der interaktiven Verteilung durch den Softwareentwickler sowie die regelbasierte, automatische Verteilung weiter verfolgt werden. Dabei sollen beide Verfahren kombiniert werden können, z. B. indem der Softwareentwickler zunächst die Verteilung bestimmter, ihm wichtig erscheinender Funktionen oder Funktionsabschnitte festlegt und anschließend durch den regelbasierten Verteilungsalgorithmus die Verteilungsinformation vervollständigt wird. Umgekehrt soll der Softwareentwickler im Anschluss an die regelbasierte Verteilung noch eingreifen und die Verteilung modifizieren können.

Die Anwendung solcher Verfahren alleine garantiert aber noch nicht die anforderungsgerechte Auslegung eines verteilten, kooperativen Steuerungssystems, da dadurch lediglich die relative Auslastung der Ressourcen zueinander eingestellt wird. Ob die durch die topologische Sicht definierten Komponenten des Steuerungssystems richtig dimensioniert sind, muss abschließend noch überprüft werden. Eventuell müssen dann noch Steuerungsmodule oder Bussysteme entsprechend skaliert oder umkonfiguriert werden. Dazu muss durch eine einfache Schätzfunktion festgestellt werden, ob die geforderten Reaktionszeiten erreichbar sind. Die Schätzfunktion kann zwar keinen endgültigen Aufschluss über die korrekte Erfüllung der Anforderungen geben, sie ermöglicht aber, mit einer fundierten Auslegung des Steuerungssystems in die Inbetriebnahme zu gehen. Kritische Anforderungen sind folglich in der Inbetriebnahme durch entsprechende Diagnosefunktionen (siehe Abschnitt 5.5.4) trotzdem noch zu überprüfen. In Bild 5-16 ist das sich ergebende Verfahren durch ein Ablaufdiagramm schematisch dargestellt.

Im Rahmen dieser Arbeit ist daher ein entsprechendes, mit einfach zu modellierenden Informationen auskommendes, regelbasiertes Verteilungsverfahren zu erarbeiten. Dazu wird in Abschnitt 5.3 die Modellierungstechnik um solche Informationen ergänzt und darauf aufbauend in Abschnitt 5.4 ein entsprechender Verteilungsalgorithmus vorgestellt. Darüber hinaus sind geeignete Schätzfunktionen zur Ermittlung von zu erwartenden Reaktionszeiten und zur Ressourcenauslastung der Steuerungsmodule zu entwickeln (Abschnitt 5.4).

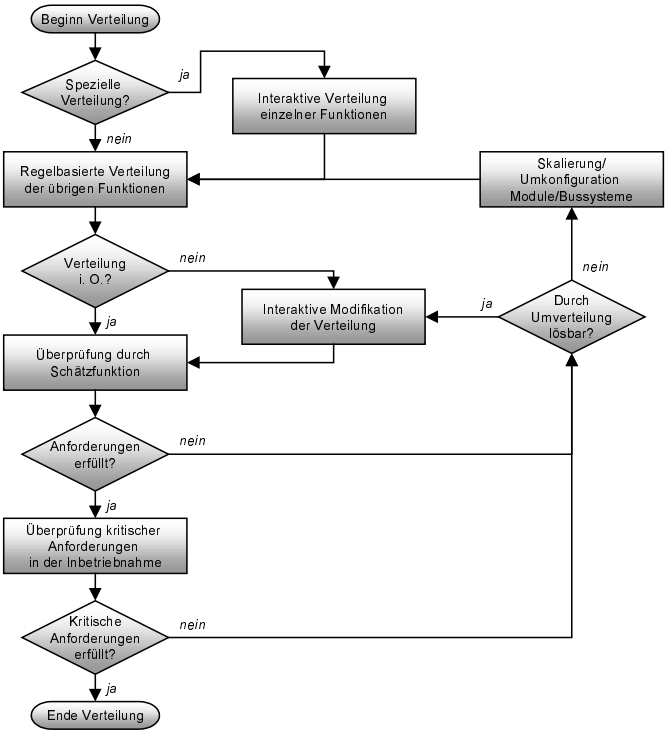

Bild 5-16: Ablaufdiagramm für die Verteilung der Steuerungssoftware auf die Steuerungsmodule

Für die interaktive Verteilung beschränkt sich die zu behandelnde Fragestellung auf die ergonomische Gestaltung einer entsprechenden Benutzerschnittstelle, was aber durch die vorliegende Arbeit nicht näher behandelt werden soll.

5.4.2 Regelbasiertes Verteilungsverfahren

Ziel im Rahmen dieser Arbeit ist es, ein regelbasiertes Verteilungsverfahren zu erarbeiten, welches mit wenigen, einfachen Leistungskennzahlen, wie sie in Abschnitt 5.3.3 vorgeschlagen wurden, auskommt. An dieser Stelle sei ausdrücklich darauf hingewiesen, dass ein komplexes Optimierungsverfahren hier nicht in Betracht kommt, da die hierfür benötigten Eingangsdaten, also z. B. das Zeitverhalten der Steuerung wie auch der gesteuerten Maschine, nicht mit vertretbarem Aufwand in ausreichendem Detaillierungsgrad modelliert werden können.

Es bietet sich hier ein iteratives Verfahren an, welches sich an der Vorgehensweise eines Steuerungsentwicklers orientiert und deshalb auch durch diesen nachvollzogen werden kann. Letzteres stellt eine unabdingbare Voraussetzung dar, um eine nachträgliche, interaktive Modifikation der Verteilung sinnvoll zu ermöglichen.

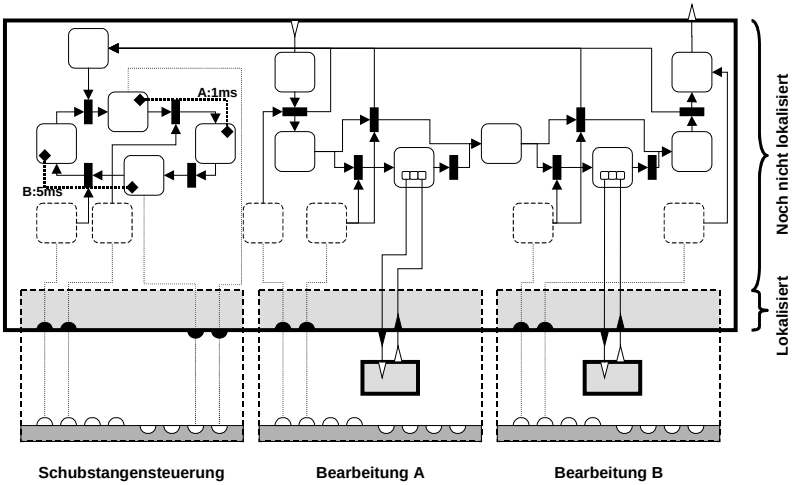

Bild 5-17: Das Beispielnetz nach der Anbindung der clientseitigen Signal- und Markenports durch den Steuerungsentwickler.

Dieses iterative Verfahren wird für jedes funktionale Modul durchlaufen, wobei das Modul vollständig verteilt wird. Dabei kommt immer ein Modul zum Zuge, dessen unterlagerte Module bereits verteilt sind. Dementsprechend wird mit einem Modul begonnen, welches auf keine anderen funktionalen Module als Client zugreift. Kommen nach Anwendung dieser Regel noch mehrere Module für den nächsten Verteilungsschritt in Betracht, so wird das Modul mit der kritischsten Reaktionszeitanforderung ausgewählt (s. hierzu Gleichung 1). Die Elemente der funktionalen Module werden dabei ebenfalls nacheinander den einzelnen Steuerungsmodulen zugewiesen.

Damit wird die Vorgehensweise eines Steuerungsentwicklers nachempfunden, was für die Durchschaubarkeit des Verfahrens durch den Steuerungsentwickler erhebliche Vorteile bringt. Dieses Verfahren zur Verteilung der Elemente eines funktionalen Moduls wird im Folgenden vorgestellt. Dabei zeigt Bild 5-17 die Ausgangssituation nach der Anbindung der clientseitigen Signal- und Marken-

ports an die entsprechenden, serverseitigen Elemente durch den Steuerungsentwickler.

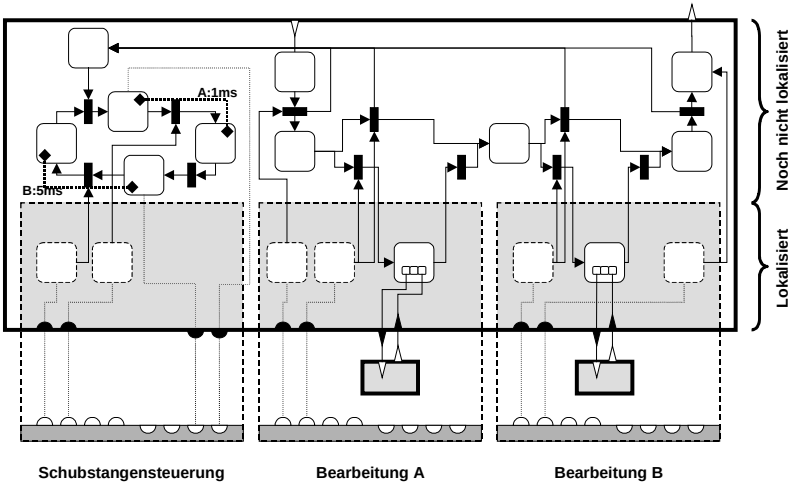

Bild 5-18: Das Beispielnetz nach der Lokalisierung der TEP, der Stellen mit Kontakt zu Signalports und der Clientstellen.

In der ersten Verteilungsphase findet die Verteilung der Transitionsbedingungsstellen und der Stellen statt, die in ihrem Entry-, Do- oder Exit-Bereich Lesebeziehungsweise Schreiboperationen auf Signalports aufweisen. Die TEP wird dabei so lokalisiert, dass eine möglichst geringe Zahl von Kommunikationsverbindungen auf andere Steuerungsmodule notwendig wird. Analog wird mit Clientstellen vorgegangen. Bild 5-18 zeigt das Beispielnetz nach diesem ersten Verteilungsschritt.

Daran schließt sich die zweite Phase der Verteilung an, in der alle noch nicht lokalisierten Netzknoten nacheinander lokalisiert werden. Dabei muss zunächst jeweils ein Netzknoten für die Lokalisierung ausgewählt werden. Um ein möglichst gutes Reaktionszeitverhalten zu erzielen, sollen immer zuerst die Netzknoten verteilt werden, die auf den Markenpfaden mit den kritischsten Reaktionszeitanforderungen liegen. Dies kann durch die folgenden Regeln sichergestellt werden:

- Es wird derjenige Netzknoten lokalisiert, von dem die Kante mit der Reaktionszeitanforderung mit der höchsten Prioritätsstufe mit einem bereits verteilten Netzknoten verbunden ist.
- Liegen mehrere Kanten zu unterschiedlichen Netzknoten, aber mit gleichen Reaktionszeitprioritäten vor, so bekommt der Netzknoten den Vorzug, dessen Kante eine geringere Reaktionszeit abzubilden hat. Dabei wird der für eine Kante zur Verfügung stehende Reaktionszeitanteil entsprechend Gleichung 1 berechnet.
- Genügen die obigen Kriterien nicht zur Auswahl eines noch nicht lokalisierten Netzknotens, so wird derjenige Netzknoten lokalisiert, der die geringste Zahl an Kanten bis zur nächsten Stelle mit Kontakt zu einem Signalport aufweist.
- Führt dies immer noch nicht zu einer eindeutigen Auswahl eines Netzknotens, so kann keine funktional begründbare Auswahl stattfinden. In diesem Fall wird beispielsweise über die Identitätsnummer des Netzknotens ausgewählt.

Gleichung 1:

$$t_R = \frac{t_P - n_{TP} \cdot t_{Tr}}{n_{kK}}$$

Darin sind:

t_R	Zulässiger Reaktionszeitanteil
t_P	Reaktionszeitanforderung entlang des Markenpfades
t_{Tr}	Transitionszeit der langsamsten Transition
n_{TP}	Anzahl der Transitionen entlang des Markenpfades
n_{kK}	Anzahl zu berücksichtigender, kommunikationsbehafteter Kanten

Gleichung 1 dient zur Bestimmung des für eine Kante zur Verfügung stehenden Reaktionszeitanteils hinsichtlich eines konkreten, durch eine Reaktionszeitanforderung belegten Markenpfades. Dabei werden von der angegebenen Reaktionszeitanforderung zunächst die zu erwartenden Transitionszeiten abgezogen. Falls die Transitionszeiten der beteiligten Module differieren, so ist hier die langsamste Reaktionszeit einzusetzen. Die sich damit ergebende Ungenauigkeit kann als tolerierbar angesehen werden, da die Transitionszeiten erheblich geringer sind, als die für die Kommunikation benötigten Zeiten. Selbstverständlich könnte dieses Verfahren auch so ausgebaut werden, dass der Einfluss unterschiedlicher Transitionszeiten berücksichtigt wird, was allerdings einen erheblichen Zusatzaufwand nach sich ziehen würde. Der verbleibende Reaktionszeitanteil wird

anschließend durch die Anzahl der entlang des Markenpfades notwendigen, kommunikationsbehafteten Kanten geteilt. Dabei müssen alle bereits lokalisierten Stellen, die unmittelbar auf dem Markenpfad liegen oder über eine Kante an den Markenpfad angeschlossen sind, berücksichtigt werden. Letztere müssen einbezogen werden, da ihre Markierung beim Schalten der Transitionen des Markenpfades ebenfalls manipuliert wird. Bild 5-19 zeigt hierzu ein Beispiel.

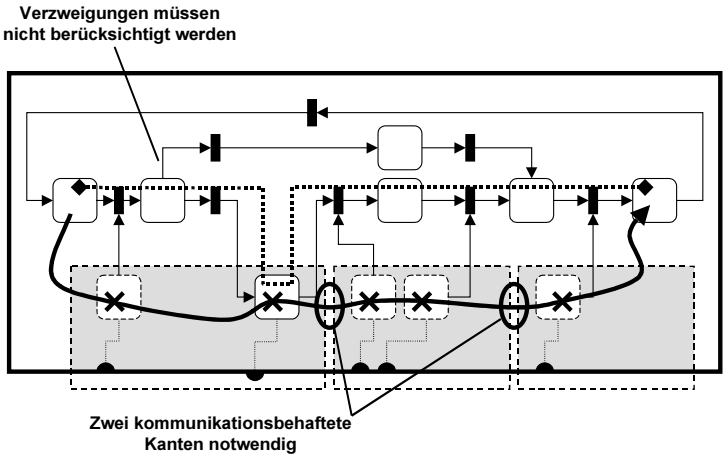

Bild 5-19: Bestimmung der Anzahl notwendiger, kommunikationsbehafteter Kanten an einem Beispiel.

Zur Ermittlung, auf welchem Steuerungsmodul ein zur Lokalisierung ausgewählter Netzknoten lokalisiert werden soll, wird für die daran angeschlossenen und mit einem bereits lokalisierten Netzknoten verbundenen Kanten eine spezifische Kantenwichtung zugewiesen. Dies geschieht für jeden mit einer Reaktionszeitforderung belegten Markenpfad. Die Kantenwichtung (nicht zu verwechseln mit dem Kantengewicht der funktionalen Sicht) wird entsprechend Gleichung 2 aus dem jeweils zur Verfügung stehenden Reaktionszeitanteil und der Speicherauslastung des Steuerungsmoduls, auf dem die Kante endet, berechnet. Eine hohe Kantenwichtung besagt dabei, dass eine Kante möglichst lokal auf einem Steuerungsmodul abzubilden ist. Der Netzknoten wird folglich auf dem Steuerungsmodul lokalisiert, zu dem die Kanten mit der höchsten aufsummierten Kantenwichtung führen.

Gleichung 2:

$$W_K = g_R \cdot \frac{t_K}{t_R} + g_S \cdot \left(1 - \frac{1}{K_S} \sum_{i=0}^{n_O} b_{SO,i}\right)$$

Darin sind:

W_K	Kantenwichtung
g_R	Gewichtungsfaktor Reaktionszeit
t_K	Kommunikationszeit (Zeit, die gerade noch eine Verteilung zulässt)
t_R	Reaktionszeit
g_S	Gewichtungsfaktor Speichernutzung
$b_{SO,i}$	Speicherbelegung des i-ten Objekts des Moduls
n_O	Anzahl der Objekte des Moduls
K_S	Speicherkapazität de Moduls

Für die Bestimmung der Kantenwichtung wird Gleichung 2 vorgeschlagen. Der Quotient des ersten Terms liefert Werte zwischen 0 und 1 für den Fall, dass bezogen auf diese Kante eine kommunikationsbehaftete Lokalisierung überhaupt die Reaktionszeitanforderungen erfüllbar ist. Ist dieser Term größer eins, dann muss der Netzknoten auf dem Steuerungsmodul lokalisiert werden, auf dem die Kante endet. Der Quotient des zweiten Terms liefert ebenfalls Werte zwischen 0 und 1, wobei der Wert eins signalisiert, dass das betreffende Steuerungsmodul bereits vollständig ausgelastet ist und keine weiteren Netzknoten mehr abbilden kann. Sind diese beiden Zwangsbedingungen abgeprüft, kann die Kantenwichtung berechnet werden. Dabei geht die Speicherbelegung negativ in die Kantenwichtung ein, hohe Speicherauslastungen wirken also verdrängend auf weitere Netzknoten, während knappe Reaktionszeitanforderungen positiv eingehen und so anziehend auf Netzknoten wirken. Um den Einfluss der Reaktionszeitanforderungen und der Speicherauslastungen für ein konkretes Steuerungssystem justieren zu können, wurden noch Gewichtungsfaktoren, deren Wert auf Basis von Erfahrungswerten oder mittels experimenteller Untersuchungen zwischen 0 und 1 festzulegen ist, mit vorgesehen.

Ergibt sich aus den oben genannten Zwangsbedingungen ein Konflikt, beispielsweise falls ein Netzknoten Kanten mit kritischen Reaktionszeitanforderungen zu unterschiedlichen Steuerungsmodulen aufweist, kann dies in der distributiven Sicht gekennzeichnet werden. Der Verteilungsalgorithmus kann trotzdem fortgesetzt werden. Sollten solche Konfliktsituationen entstehen, muss der Steuerungsentwickler im Anschluss an die automatische Verteilung hier noch korrigierend eingreifen. Dies kann beispielsweise durch Zusammenlegen in zeitkritischem Zusammenhang stehender Ein-/ und Ausgänge geschehen.

Damit ist die Verteilung der Netzknoten des einzelnen funktionalen Moduls abgeschlossen. Bild 5-15 aus Abschnitt 5.3.4 zeigt dies für das Beispielnetz. Das Verteilungsverfahren wird dann mit dem nächsten funktionalen Modul fortgesetzt. Auf diese Weise wird von den hardwarenahen funktionalen Modulen zu funktionalen Modulen mit zunehmend koordinierenden Eigenschaften übergegangen, bis die Module der funktionalen Sicht vollständig verteilt sind.

5.4.3 Schätzfunktion zur Beurteilung von Verteilungen

Bei der Schätzung von Reaktionszeiten zur Beurteilung von Verteilungen werden nur Reaktionszeitanforderungen der Prioritätsebene A berücksichtigt, also diejenigen, die in der Modellierung der funktionalen Sicht als unbedingt einzuhalten markiert worden sind.

Gleichung 3:

$$T_{P,\max} = \sum_{i=0}^{n_T}\sum_{j=0}^{n_M} T_{Trans,i,j} + 2\cdot\sum_{i=0}^{n_S}\sum_{j=0}^{n_M} T_{Stack,i,j} + \sum_{i=0}^{n_{EA}}\sum_{j=0}^{n_M} T_{EA,i,j}$$

$$+ \sum_{i=0}^{n_C}\sum_{j=0}^{n_B} T_{Comm,i,j} + \sum_{i=0}^{n_B} T_{Wait,i} + 2\cdot\sum_{i=0}^{n_{RM}}\sum_{j=0}^{n_{RS}} T_{Stack,i,j}$$

Darin sind:

$T_{Trans,i,j}$	Transitionszeit der i-ten Transition des j-ten Steuerungsmoduls
$T_{Stack,i,j}$	Stackzeit der k-ten Kommunikation des j-ten Steuerungsmoduls
$T_{EA,i,j}$	l-te Zugriffszeit des j-ten Steuerungsmoduls
$T_{Comm,i,j}$	Zeit der m-ten Kommunikation des n-ten Bussystems
$T_{Wait,i}$	Wartezeit des n-ten Bussystems
$T_{Stack,i,j}$	Stackzeit des m-ten Routingmoduls
n_T	Anzahl betroffener Transitionen eines Moduls
n_M	Anzahl betroffener Module des zu prüfenden Pfades
n_S	Anzahl benötigter Kommunikationschritte eines Module
n_{EA}	Anzahl relevanter Zugriffszeiten eines Moduls
n_C	Anzahl benötigter Kommunikationschritte pro Bussystem
n_B	Anzahl betroffener Bussysteme
n_{RS}	Anzahl benötigter Kommunikationschritte pro Routingmodul
n_{RM}	Anzahl betroffener Routingmodule des Pfades

Durch den Steuerungsentwickler ist bei der Überprüfung anzugeben, welche Markenpfade gleichzeitig aktiv sein können. Für diese Cluster von nebenläufi-

gen, zeitkritischen Markenpfaden kann dann eine Worst-Case-Abschätzung entsprechend Gleichung 3 vorgenommen werden.

In der obigen Gleichung wurde von Worst-Case-Annahmen ausgegangen: für jede Reaktionszeitanforderung eines Clusters müssen alle Worst-Case-Zeiten für Steuerungsmodule und Bussysteme, die auf dem Markenpfad liegen, berücksichtigt werden. Dabei wird angenommen, dass Schaltoperationen auf anderen Markenpfaden mit Priorität A die untersuchten Zeiten maximal negativ beeinflussen.

5.5 Systemsoftware für verteilte, kooperative Steuerungen

Entsprechend der in Abschnitt 5.1 vorgestellten Grundkonzeption müssen seitens der Steuerungsmodule eine Reihe von Funktionalitäten zur Verfügung gestellt werden, die im folgenden unter dem Begriff der Systemsoftware für verteilte, kooperative Steuerungen zusammengefasst werden sollen. Diese Systemsoftware stellt gewissermaßen ein erweitertes, auf die hier vorgeschlagene, verteilte, kooperative Steuerungstechnik zugeschnittenes Betriebssystem dar. Sie besteht aus den Funktionalität zur Abarbeitung der Steuerungssoftware und zur transparenten Kommunikation sowie den Softwaremanagement- und Diagnosefunktionen.

5.5.1 Abarbeitung der Steuerungssoftware

Unter dem Begriff Abarbeitung der Steuerungssoftware wird in dieser Arbeit die Abbildung der grafisch erstellten Steuerungsabläufe und deren Ausführung durch die Steuerungsmodule entsprechend der Semantik der funktionalen Modellierungssicht verstanden. Sie umfasst daher das Lesen und Interpretieren von Eingängen, Variablen, Zuständen und Zeitgliedern, die Verknüpfung dieser Informationen miteinander, das von diesen Verknüpfungen abhängige Weiterschalten von Abläufen und Manipulieren interner Zustände und Variablen sowie das sich daraus ergebende Setzen von Ausgängen, Starten von Zeitgliedern oder Absetzen von Meldungen an das Bedienfeld der Maschine.

Durch die Verteilung muss bei verteilten, kooperativen Steuerungen die Komplexität einer mit parallelen Prozessen arbeitenden Abarbeitungsfunktionalität beherrscht werden. Insofern unterscheidet sich eine verteilte, kooperative Steuerungstechnik deutlich von der herkömmlichen SPS-Technik, da eine streng zyklische Arbeitsweise des gesamten Steuerungsprogramms nicht mehr sinnvoll realisiert werden kann und daher auch der damit verbundene Beherrschbarkeitsvorteil wegfällt. Folglich besteht bei verteilten, kooperativen Steuerungen kein Anlass mehr, den für eine zyklische Bearbeitung typischen, hohen Ressourcenbedarf bei gleichzeitig mäßigen Reaktionszeiten in Kauf zu nehmen. Statt dessen soll auch

für die Abarbeitung der Steuerungssoftware durch die Steuerungsmodule konsequent ein durchgängig ereignisorientiertes Konzept verfolgt werden.

Allerdings darf die ereignisorientierte Abarbeitungsfunktionalität nicht zu einer Komplexitätssteigerung in der Entwicklung der Steuerungssoftware führen. Vielmehr muss die dadurch entstehende Komplexität durch die Steuerungsmodule und das Entwicklungswerkzeug gekapselt werden (vgl. Abschnitt 3.2.1 – Kapselung der Komplexität). Die Zuordnung von Funktionen bzw. Abschnitten von Funktionen zu Tasks (Rechenprozessen) muss daher implizit und eindeutig erfolgen, ohne dass hierzu ein Zutun des Steuerungsentwicklers nötig wird. Dementsprechend soll die Abarbeitungsfunktionalität ein geeignetes Taskzuordnungskonzept implementieren. Eine entsprechende Detaillierung findet in Abschnitt 5.6 statt.

5.5.2 Transparente Kommunikation

In Abschnitt 3.2.1 wurde in der Analyse der Anforderungen die Notwendigkeit einer Funktionalität zur transparenten Kommunikation herausgearbeitet. Dabei handelt es sich um eine Funktionalität der Steuerungsmodule, die durch das Entwicklungswerkzeug aufgrund der verteilungsbedingten Kommunikations- und Synchronisationserfordernisse parametrisiert wird. Diese Funktionen treten für den Softwareentwickler nicht in Erscheinung, weshalb sie als transparent bezeichnet werden.

Bei der Verteilung der Steuerungssoftware ergibt sich in der distributiven Sicht (siehe Abschnitt 5.3.1), dass auf bestimmte Stellen von verschiedenen Steuerungsmodulen aus zugegriffen werden muss. Dies kann durch das Entwicklungswerkzeug anhand der die Steuerungsmodulgrenzen überschreitenden Kanten festgestellt werden. Entsprechend kann ein dafür ausgelegter, durch die Systemsoftware der Steuerungsmodule bereitgestellter Kommunikations- und Synchronisationsmechanismus eingesetzt werden.

Dabei werden diejenigen Stellen, auf die von mehreren Steuerungsmodulen aus zugegriffen wird, in mehrere Lokalstellen aufgeteilt, die jeweils als lokale Kopie auf den betroffenen Steuerungsmodulen abgebildet werden (siehe Bild 5-20). Dadurch kann die Abarbeitungsfunktionalität auf diese Stellen zugreifen, ohne dass deren Verteilung berücksichtigt werden muss. Um die Markierung zusammengehöriger Lokalstellen auf allen Steuerungsmodulen konsistent zu halten, überträgt eine zusätzliche Funktionalität bei jeder lokalen Änderung diese auf sämtliche anderen Lokalstellen. Hierzu wird eine Zentralstelle auf einem der beteiligten Steuerungsmodule eingeführt, der diese Funktionalität zugeordnet wird.

Eine weitere Detaillierung dieser Funktionalität wird in Abschnitt 6.3.2 vorgenommen.

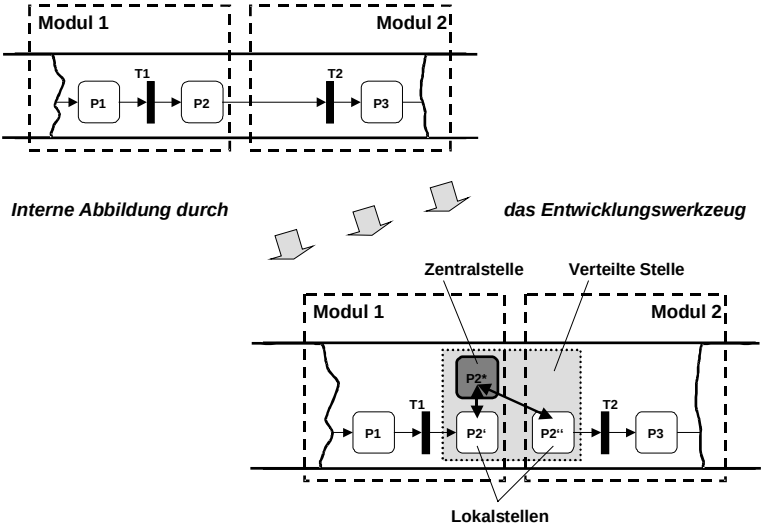

Bild 5-20: Realisierung der Funktionalität der transparenten Kommunikation durch das Konzept der verteilten Stelle.

5.5.3 Softwaremanagementfunktionen

Um die Anforderung nach einer guten Beherrschbarkeit verteilter, kooperativer Steuerungen insbesondere in der Inbetriebnahme zu erfüllen, werden Funktionen zum Down- und Upload der Steuerungssoftware sowie zum Starten und Stoppen der Steuerungsfunktionen benötigt. Diese Funktionen sollen unter dem Begriff Softwaremanagementfunktionen zusammengefasst werden.

Die Softwaremanagementfunktionen werden im Zusammenspiel des Softwaremanagementwerkzeugs einer Entwicklungsumgebung und einer unterstützenden Funktionalität der Steuerungsmodule realisiert. Diese Funktionalität gehört dementsprechend ebenfalls zum Umfang der Systemsoftware der Steuerungsmodule. Der dabei benötigte Funktionsumfang soll im folgenden kurz beschrieben werden.

Grundsätzlich muss in der Inbetriebnahme zunächst die Steuerungssoftware durch eine Downloadfunktion entsprechend der in der distibutiven Sicht festge-

legten Verteilung über das Kommunikationssystem an die Steuerungsmodule übertragen werden. Dazu sollen die Steuerungsmodule spezielle Kontrollobjekte zur Verwaltung jeweils einer Teilfunktion enthalten, an die die Datenpakete, welche die jeweilige Teilfunktion beschreiben, gerichtet werden. Diese Kontrollobjekte setzen daraus die Teilfunktion zusammen, bereiten ihre Abarbeitung vor und parametrisieren die benötigten Funktionen der transparenten Kommunikation. Anschließend kann durch eine weitere Funktion die Abarbeitung gestartet werden.

Um zu vermeiden, dass bei jedem einzelnen Start des Steuerungssystems dieser Downloadvorgang wiederholt werden muss, wird die einmal geladene Software auf dem Steuerungsmodul permanent gespeichert. Findet ein solches Softwaremodul bei einem erneuten Start bereits Steuerungssoftware im Permanentspeicher vor, so wird diese Software sofort zur Abarbeitung gebracht. Damit auch zu einem späteren Zeitpunkt noch die Steuerungssoftware – wie im Beispiel in Bild 5-21 - modifiziert werden kann, muss die Abarbeitung einzelner Teilfunktionen gezielt angehalten, die Kommunikationsverbindungen getrennt und die Teilfunktion aus dem Permanentspeicher entfernt werden können. Entsprechend muss auch der erneute Download, der Aufbau der Kommunikationsverbindungen und der Start der Abarbeitung gezielt für einzelne Teilfunktionen durchgeführt werden können. Dabei sollen die Auswirkungen auf andere Funktionen möglichst gering sein.

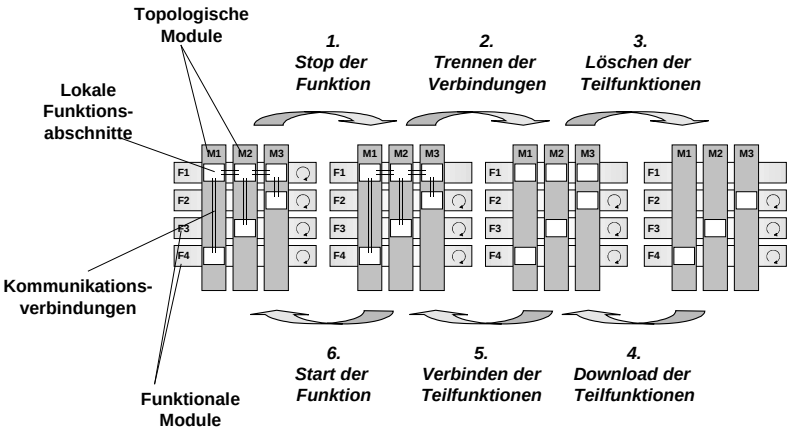

Bild 5-21: Aktualisierung einer Steuerungsfunktion durch die Funktionen des Softwaremanagements.

Die Funktionalität des Upload stellt die Umkehrung des Download dar, wobei durch das Softwaremanagementwerkzeug die Steuerungssoftware aus den Informationen der Steuerungsmodule rekonstruiert wird. Sie wird insbesondere für zwei Zwecke benötigt: Einerseits dient sie dazu, mit einem Visualisierungswerkzeug die verteilte Software, beispielsweise zu Diagnosezwecken, anzeigen zu können, selbst wenn die ursprünglichen Entwicklungsprojektdaten nicht verfügbar sind. Andererseits kann durch einen Upload die Software aller Steuerungsmodule an zentraler Stelle gesichert werden, um einen späteren Modultausch zu vereinfachen.

5.5.4 Diagnosefunktionen

Zur Erfüllung der Anforderung nach einer guten Beherrschbarkeit verteilter, kooperativer Steuerungen in der Inbetriebnahme wie auch im Betrieb werden neben den Softwaremanagementfunktionen auch Funktionen zur Diagnose des verteilten, kooperativen Steuerungssystems benötigt. Unter Diagnose ist in diesem Zusammenhang die Sicherstellung der korrekten Funktion der modellierten Steuerungssoftware sowie die Fehlersuche im Rahmen der Inbetriebnahme oder der Instandsetzung zu verstehen.

Die meldungsbasierte Form der Diagnose, die auf eigens in der Steuerungssoftware programmierten Fehlererkennungsfunktionen aufsetzt, ist davon nicht betroffen (siehe hierzu z. B. *Reinhart u. a. 1999a*). Im Gegensatz zu den hier behandelten Diagnosefunktionen werden dabei nicht Fehler im Steuerungssystem oder der Steuerungssoftware selbst, sondern Fehler in der gesteuerten Maschine diagnostiziert. Daher müssen sie auch weiterhin als maschinenspezifischer Bestandteil der Steuerungssoftware modelliert werden.

Analog zu den Softwaremanagementfunktionen werden die Diagnosefunktionen im Zusammenspiel zwischen dem Diagnosewerkzeug und der Systemsoftware der Steuerungsmodule realisiert. Dabei werden durch den Benutzer des Diagnosewerkzeuges spezifische Diagnoseaufträge in entsprechenden Bediendialogen formuliert. Darauf ermittelt das Diagnosewerkzeug über die im gemeinsamen Datenmodell der Steuerungssoftware (siehe Bild 5-3) hinterlegten Verteilungsinformation das jeweils betroffene Steuerungsmodul und setzt den Diagnoseauftrag dorthin ab. Die Systemsoftware des Steuerungsmoduls nimmt diesen Diagnoseauftrag entgegen, ermittelt die gewünschten Informationen und sendet das Ergebnis an das Diagnosewerkzeug.

Folglich muss die Systemsoftware für alle diagnoserelevanten Informationen entsprechende Diagnosefunktionen enthalten. Als diagnoserelevant werden dabei der Zustand von Ein- und Ausgängen, der Wert von Variablen und der Stand von

Zeitgliedern sowie der Abarbeitungszustand von Funktionen erachtet (*VDW 2000*). Dabei wird neben den aktuellen Werten häufig auch der zeitliche Verlauf dieser Werte benötigt. Zusätzlich ist die korrekte Funktion der Steuerungshardware von Interesse.

Die Diagnose der Steuerungshardware umfasst die Überprüfung der Busverbindungen und der Steuerungsmodule. Dazu wird in der Systemsoftware der Steuerungsmodule eine Funktion vorgesehen, die in einem einstellbaren Zeitraster die Erreichbarkeit aller Kommunikationspartner überprüft. Die Ergebnisse dieser Erreichbarkeitstests können jederzeit durch das Diagnosewerkzeug abgerufen und in der topologischen Modellierungssicht visualisiert werden.

Zur Diagnose der Steuerungssoftware sind Funktionen zum Anzeigen und Testen von Eingängen und Ausgängen sowie zum Beobachten und Manipulieren von Variablen vorgesehen. Zur Analyse des zeitlichen Verlaufs von Variablen kann lokal auf dem Steuerungsmodul ein Protokollmechanismus gestartet werden, der Veränderungen der zu beobachtenden Softwareelemente zeitbezogen zwischenspeichert. Diese Daten können dann später durch das Diagnosewerkzeug abgefragt werden. Um dabei die Werteverläufe von Softwareelementen verschiedener Steuerungsmodule in eine zeitliche Beziehung setzen zu können, müssen die lokalen Systemuhren der Steuerungsmodule ausreichend genau synchronisiert werden (siehe Abschnitt 6.1.5). Damit können dann Rückschlüsse auf das Maschinen- wie auch auf das Reaktionszeitverhalten der verteilten Steuerung gezogen werden.

Für die Diagnose von Abläufen der Steuerungssoftware sind über die Visualisierung des jeweiligen Ablaufzustandes hinaus Funktionen zum Setzen von Haltepunkten und für einen Einzelschrittbetrieb der Ablaufsoftware vorgesehen.

5.6 Zusammenfassung

Im vorliegenden Abschnitt wurde zur Erarbeitung eines Konzepts für die verteilte, kooperative Steuerungstechnik zunächst eine Grundkonzeption vorgestellt, die die Entwicklungswerkzeuge einerseits und die Systemsoftware der Steuerungsmodule andererseits in Beziehung zueinander setzt. Daraufhin wurde die durch die Entwicklungswerkzeuge abzubildende Entwicklungsmethode konzeptioniert. Sie basiert auf den drei wesentlichen Schritten der funktionalen Modellierung, der Festlegung der Steuerungstopologie sowie der Festlegung der Verteilung. Für diese drei Schritte wurde jeweils ein Konzept für eine am Vorgehen ausgerichtete, aufgabenorientierte Modellierungssicht vorgestellt.

Die **funktionale Sicht** dient der von der Steuerungstopologie sowie der späteren Verteilung unabhängigen Modellierung der Steuerungsfunktionen. Dabei kommen als Basismodellierungstechnik colorierte Petrinetze zur Anwendung, auf die die im Werkzeugmaschinenbau etablierten Modellierungstechniken abgebildet werden können, was durch entsprechend ausgestaltete Editoren geschehen kann. Damit kann das verteilte, kooperative Steuerungssystem in der jeweils gewünschten Technik modelliert werden, ohne dass deshalb steuerungsintern mit verschiedenen Techniken gearbeitet werden müsste. Des weiteren werden in die funktionale Sicht noch Sprachelemente aufgenommen, die die Modellierung der sich aus der Steuerungsfunktionalität ergebenden Reaktionszeitanforderungen erlauben.

Die **topologische Sicht** dient der Modellierung der Steuerungstopologie, die die Steuerungsmodule, deren Verbindung durch Bussysteme sowie die Zuordnung der Sensoren und Aktoren zu den Steuerungsmodulen umfasst. Sie umfasst die zur Ermittlung einer günstigen Verteilung benötigten Kenndaten zur Leistungsbeschreibung der Steuerungsmodule und Bussysteme.

Die **distributive Sicht** dient zur Festlegung der Verteilung der Steuerungssoftware auf die Steuerungshardware. In Abschnitt 5.4 wurde ein Verteilungsverfahren vorgeschlagen, das auf einer Kombination von interaktiver Verteilung durch den Steuerungsentwickler und einem automatischen, regelbasierten Verteilungsalgorithmus beruht. Zusätzlich wurde eine Schätzfunktion zur Bewertung von Verteilungen erarbeitet.

Durch dieses Konzept wurde die angestrebte Entkopplung der Entwicklung der Funktionalität von der Festlegung der Steuerungstopologie erreicht. Damit konnten die Anforderungsbereiche der effizienten Entwicklung (Abschnitt 3.2.1) und der geeigneten Modellierungstechnik (Abschnitt 3.2.2) erfüllt werden. Darüber hinaus wurde für die Systemsoftware ein Konzept vorgestellt. Es umfasst die Funktionsbereiche zur Abarbeitung der Steuerungssoftware, zur transparenten Kommunikation, zum Softwaremanagement und zur Diagnose. Insbesondere die letzteren beiden Funktionsbereiche tragen entscheidend zur Erfüllung des Anforderungsbereichs der Beherrschbarkeit in der Inbetriebnahme und im Betrieb bei. Mit einer verteilten, kooperativen Steuerungstechnik nach dem hier erarbeiteten Konzept können also deren Potenziale genutzt werden.

6 Designkonzept der Systemsoftware

Im vorliegenden Kapitel wird ein Designkonzept für die Systemsoftware verteilter, kooperativer Steuerungen erarbeitet, welches einerseits für die prototypische Realisierung der Steuerungsmodule im Rahmen dieser Arbeit verwendet wird und andererseits als Leitlinie für kommerzielle Entwicklungen dienen kann. Dazu werden in Abschnitt 6.1 die zur Anwendung kommenden, grundlegenden Designprinzipien behandelt, in Abschnitt 6.2 wird das Designkonzept der Systemsoftware vorgestellt und in Abschnitt 6.3 wird auf die Abbildung der Systemsoftwarefunktionen auf das Designkonzept eingegangen.

6.1 Grundlegende Designprinzipien

6.1.1 Softwaretechnische Abbildung colorierter Petrinetze

Um die in der funktionalen Sicht in Form von colorierten Petrinetzen spezifizierten Steuerungsfunktionen zur Abarbeitung zu bringen, wird ein Verfahren zu deren softwaretechnischer Abbildung benötigt. Hierzu können prinzipiell die folgenden Verfahren eingesetzt werden:

- Beim Verfahren der **Programminterpretation** wird die bei der Modellierung erzeugte, in Form einer Beschreibungsdatei oder einer Datenbank vorliegende Information zur Laufzeit der Steuerungsapplikation durch die Steuerungsmodule interpretiert und entsprechend die Steuerungsfunktionalität ausgeführt. Der entscheidende Nachteil dieses Verfahrens ist der hohe Interpretationsaufwand zur Laufzeit, der zwangsläufig zu einer verminderten Reaktionsgeschwindigkeit des Steuerungssystems führt.

- Beim Verfahren der **Programmcompilation** wird aus den Modellierungsdaten in einem oder mehreren Compilationsschritten ein ablauffähiges Programm erzeugt. Dazu muss entweder ein eigener Hochsprachencompiler gebaut werden, was sehr aufwendig ist, oder es erfolgt ein Zwischenschritt, der die Modellierungsdaten in eine gebräuchliche Hochsprache übersetzt, von der aus eine weitere Compilation ohne Probleme möglich ist.

- Ein weiteres Verfahren, das hier **Programminstanzierung** genannt werden soll, setzt auf einer spezifischen Systemsoftware auf, die in der Lage ist, aus den Modellierungsdaten gewonnene Parametersätze in ein lauffähiges Programm zu überführen. Dies geschieht, indem die Parametersätze benutzt werden, um Petrinetz-Objekte zu instanzieren und zu verknüpfen. Das Zusammenwirken dieser Objekte nach den Regeln der colorierten Petrinetze erzeugt

das spezifizierte Steuerungsverhalten. Die Petrinetz-Klassen sind dabei Teil der Systemsoftware.

Die beiden letztgenannten Verfahren weisen gegenüber der Programminterpretation nicht den Nachteil einer rechenzeitaufwendigen Interpretation zur Laufzeit auf. Vergleicht man die Programmcompilation mit der Programminstanzierung, so sind hier keine wesentlichen Leistungsunterschiede zu erwarten. Die Programminstanzierung weist aber den Vorteil auf, dass die modellierte Steuerungsfunktionalität direkt und mit geringem Aufwand in entsprechende Objekte in den Steuerungsmodulen umgewandelt werden kann. Dies bietet auch hinsichtlich der Beherrschbarkeit der Steuerung erhebliche Vorteile, da der Zustand der Steuerungsnetze ohne wesentlichen Zusatzaufwand in der funktionalen Sicht beispielsweise zu Diagnosezwecken visualisiert werden kann. Aus diesen Gründen ist das Verfahren der Programminstanzierung für die softwaretechnische Abbildung colorierter Petrinetze am besten geeignet.

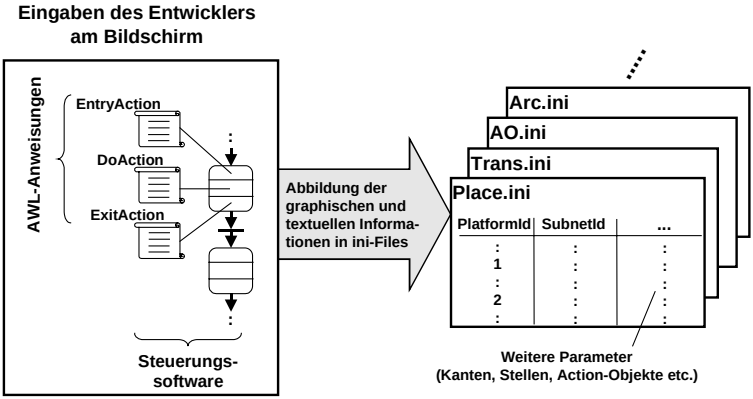

Bild 6-1: Abbildung der modellierten Steuerungsfunktionen in durch die Systemsoftware leicht interpretierbare Parametersätze.

Für das Konzept der Programminstanzierung werden zunächst Klassen für die Objekte der funktionalen Sicht benötigt, die als Teil der Systemsoftware zu implementieren sind. Bevorzugt zu nennen sind dabei die Stelle, die Transition, Kanten und Anweisungen (Action-Objekte).

Die für die Instanzierung der Objekte benötigten Parametersätze werden dabei entsprechend Bild 6-1 erzeugt. Um eine unkomplizierte Instanzierung zu ermöglichen, werden eine Reihe von Parametern – z. B. Informationen über die Ver-

bindungen der Objekte untereinander - in den Parameterfiles bewusst redundant abgelegt.

Je nach Komplexität der durch die grafische Spezifikation festgelegten Objekte können vereinfachte Klassen eingesetzt werden. Daraus folgt, dass die Beziehungen zwischen den Petrinetzklassen zwischen abstrakten Basisklassen geknüpft werden. Real wird dann jeweils ein komplexes Objekt – z. B. für die Abbildung einer Stelle mit verschiedenen Kapazitäten für Marken unterschiedlicher Farbgruppenzugehörigkeit - oder ein entsprechend einfaches Objekt – z. B. für die Abbildung einer einfachen Synchronisationsstelle – eingesetzt.

6.1.2 Ereignisorientierung

Zur Ausnutzung der Leistungspotenziale durch lokale Reaktionen (siehe Abschnitt 3.1.3) muss der Kommunikationsaufwand minimiert werden. Dies ist erreichbar, wenn nur bei Änderung einer für die Steuerungsfunktionen relevanten Information eine Kommunikation erfolgt. Daraus folgt, dass das Kommunikationssystem konsequent ereignisorientiert aufgebaut werden muss. Dies erfordert, dass alle Steuerungsmodule aus eigener Initiative heraus Ereignisse zu jedem anderen Steuerungsmodul übertragen können, was zugleich die Voraussetzung für die in den Potenzialen (siehe Abschnitt 3.1.3) beschriebene, direkte Kooperation darstellt. Für die einzusetzenden Kommunikationssysteme stellt dies gleichzeitig die Anforderung der Multimasterfähigkeit dar.

Zur Erreichung eines optimalen Reaktionsverhaltens wird daher konsequent ein ereignisorientiertes Konzept auch in der Systemsoftware eingesetzt. So kann z. B. Polling bzw. eine zyklische Abarbeitung vermieden werden. Eingetretene Ereignisse, wie z. B. die Änderung eines Eingangssignals, werden dabei von der Systemsoftware erkannt und abhängig davon die entsprechende Steuerungsfunktion abgearbeitet. Die Vergabe von Rechenzeit nur an von Veränderungen betroffene Softwaresegmente ist durch eine Aufteilung des Steuerungsprogramms in Tasks möglich. Die Verarbeitung von gleichzeitig eintreffenden Ereignissen bzw. die vorrangige Verarbeitung von wichtigen Ereignissen, wie z. B. das Erreichen einer Endlage, setzt den Einsatz eines preemptiven, prioritätsgesteuerten Multitaskingsystems voraus. Daher ergibt sich als weitere Fragestellung, wie ein solches Multitaskingsystem eingesetzt werden soll. Diese Frage wird im folgenden Abschnitt näher erörtert.

6.1.3 Multitaskingkonzept

Für den Einsatz eines Mutlitaskingsystems muss zunächst erörtert werden, welche Elemente der individuellen Steuerungssoftware mit eigener Rechenleistung ausgestattet werden müssen.

Eine Transition kann nur dann schaltfähig werden, wenn sich die Markierungssituation in ihrer Umgebung verändert. Durch die Parallelität in Petrinetzen können auch mehrere Transitionen quasigleichzeitig schaltfähig werden. Somit ist es sinnvoll, jeder Transition eine eigene Task zuzuordnen. Beim Schaltvorgang werden zuerst die Exit-Anweisungen des Vorbereichs und dann die Entry-Anweisungen des Nachbereichs durchgeführt. Falls Stellen Do-Anweisungen enthalten, müssen hierfür ebenso eigene Tasks vorgesehen werden.

Eine weitere Fragestellung beim Einsatz von Multitaskingsystemen ist generell die Gewährleistung der Datenkonsistenz, d. h. der Schutz von Daten vor unerlaubtem Mehrfachzugriff. Dies wird mit den bei Multitasking-Systemen (*Witzack 2000*) üblichen Konzepten der Semaphore (zur Synchronisation von Prozessen und zum Zugriffsschutz bei gemeinsam genutzten Daten) und der Mailbox (zum zeitentkoppelten Nachrichtenaustausch zwischen Prozessen) gelöst.

Für einen korrekten Markenfluss werden deshalb Stellen mittels Semaphoren vor konkurrierenden Transitionszugriffen geschützt (siehe Bild 6-2), indem sie während einer Markierungsabfrage seitens einer Transition zunächst gesperrt werden. Eine Transition gibt die von ihr gesperrten Stellen erst wieder frei, wenn die Schaltfähigkeitsüberprüfung negativ abbricht oder wenn der Schaltvorgang komplett beendet ist. Tasks, die auf eine bereits gesperrte Stelle zugreifen wollen, blockieren solange, bis dieser wieder freigegeben wurde.

Diese Abfrage der Schaltfähigkeit kann auch für das Auswerten der Transitionsbedingung eingesetzt werden. Neben dem in Abschnitt 5.3.1 erläuterten Vorteil einer günstigen, grafischen Visualisierung des Netzzustandes müssen Transitionen dann nur die Markierungssituation in ihrem Umfeld beobachten. Eine durch die Transition realisierte Transitionsbedingung ist damit nicht mehr nötig. Die Funktionalität der TEP ist dagegen bereits nahezu vollständig in der Basisklasse der Stelle vorhanden, da zur Ergebnisermittlung des booleschen Ausdrucks beispielsweise der Entry-Bereich der Stelle genutzt werden kann.

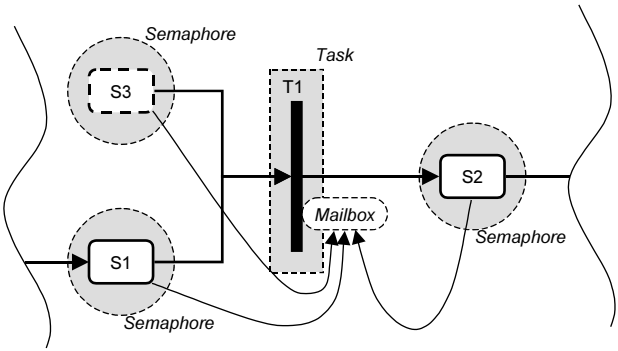

Bild 6-2: Überblick über die Taskstruktur sowie die Verwendung von Semaphoren und Mailboxen zur Abbildung des Petrinetzes.

Aus Gründen der Rechenzeitoptimierung ist es sinnvoll, Transitionen immer dann den Anstoß zur Überprüfung ihrer Schaltfähigkeit zu geben, wenn sich die Markierungssituation im Vor- oder Nachbereich geändert hat. Der Überprüfungsvorgang wird sofort abgebrochen, sobald eine Markierung festgestellt wird, die einem Schalten entgegensteht. Die Transition wartet dann an einer Mailbox auf die Meldung einer relevanten Markierungsänderung. Die Überprüfung der Schaltfähigkeit wird nur dann wieder aufgenommen, wenn die Änderungsmeldung von der Stelle stammt, bei der zuletzt der Überprüfungsvorgang abgebrochen wurde, da zum Erreichen der Schaltfähigkeit sich mindestens diese Markierung ändern muss. Dadurch kann die Häufigkeit der Überprüfung der Schaltfähigkeit von Transitionen sehr weit reduziert werden, was zu einer effizienteren Prozessorzeitnutzung führt.

6.1.4 Kommunikationsbündelung und -priorisierung

Jedes Ereignis, welches über die Modulgrenze hinaus von Interesse ist, würde zunächst einmal einen kompletten Kommunikationszyklus auslösen, wobei zusätzliche Daten - der vom eingesetzten Kommunikationssystem abhängige Overhead – übertragen werden müssen. Insbesondere, wenn der Informationsgehalt nur aus einem oder wenigen Bit besteht, ergibt sich dabei eine extrem ungünstige Nutzdatenrate. Dadurch wird der Vorteil der rein ereignisorientierten Kommunikation gegenüber der zyklischen Kommunikation teilweise aufgezehrt.

Weiterhin ist zu beachten, dass nicht jedes Ereignis die gleichen Anforderungen an die Reaktionszeit hat. Beispielsweise muss das Schließen des Greifers beim Werkzeugwechsel sofort übertragen werden, da dies direkt in die so weit wie

möglich zu reduzierende Span-zu-Span-Zeit eingeht. Im Vergleich dazu ist beispielsweise das Überschreiten einer Kühlschmierstofftemperatur nicht als zeitkritisches Ereignis anzusehen und muss deshalb nicht sofort übertragen werden.

Ausgehend von diesen Überlegungen wird ein Konzept vorgeschlagen, das eine Bündelung von Ereignismeldungen pro Kommunikationsverbindung zwischen zwei Modulen vorsieht. Dabei wird je nach Anforderungen an die Reaktionszeit der einzelnen Ereignisse eine bestimmte Zeit gewartet, in der weitere Ereignismeldungen zum gleichen Zielmodul in ein gemeinsames Telegramm abgelegt werden. Hat das Telegramm seine Maximalgröße erreicht, ist die maximal zulässige Wartezeit eines bereits enthaltenen Ereignisses erreicht oder kommt eine Meldung sehr hoher Priorität hinzu, wird das Telegramm abgesendet. Der Telegrammoverhead ist folglich nur einmal vorhanden, die Nutzdatenrate kann also deutlich verbessert werden.

Dementsprechend kann durch eine zielgerichtete Vergabe von maximalen Wartezeiten bei Ereignismeldungen eine Priorisierung der zeitkritischen Kommunikation erfolgen. Dazu kann aus den modellierten Reaktionszeitanforderungen jeweils eine Maximalwartezeit ermittelt werden. Für nicht durch Reaktionszeitanforderungen belegte Abschnitte der Petrinetze kann dabei eine entsprechend längere Standardmaximalwartezeit verwendet werden. Für die Kommunikation, die nicht in direktem Zusammenhang mit der Steuerungsfunktionalität an sich steht – also beispielsweise für die Kommunikation zur Visualisierung des Markenflusses durch ein Visualisierungswerkzeug – können diese Zeiten nochmals größer bemessen werden.

6.1.5 Synchronisation der Systemuhren

In einem verteilten System mit mehreren Steuerungsmodulen besteht grundsätzlich das Problem, eine einheitliche Systemzeit zur Verfügung zu haben. Dazu müssen die lokalen Systemuhren der einzelnen Steuerungsmodule miteinander synchronisiert werden. Hierzu gibt es prinzipiell zwei Möglichkeiten: Einerseits durch Synchronisation auf Basis einer Hardwareunterstützung durch das Kommunikationssystem, andererseits durch Synchronisation durch den Austausch von Zeitstempeln auf einer höheren Ebene.

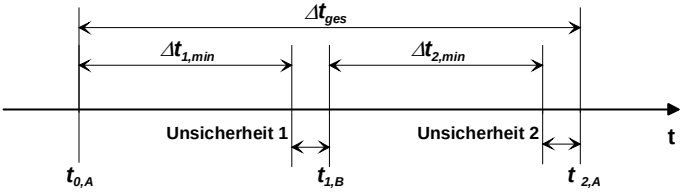

Bild 6-3: Theoretische Betrachtung der mit der Übertragung von Zeitstempeln erreichbaren Synchronisationsgenauigkeit

Gleichung 4:

$$t_{2,A} = t_{1,B} + \Delta t_{2,\min} + \frac{1}{2} \cdot \left(\Delta t_{ges} - \Delta t_{1,\min} - \Delta t_{2,\min} \right)$$

Gleichung 5:

$$U = \pm \frac{1}{2} \cdot \left(\Delta t_{ges} - \Delta t_{1,\min} - \Delta t_{2,\min} \right)$$

Darin sind:

$t_{0,A}$	mit der Systemuhr des Moduls A gemessener Zeitpunkt 0
$t_{1,B}$	mit der Systemuhr des Moduls B gemessener Zeitpunkt 1
$t_{2,A}$	errechnete Zeit für die Systemuhr des Moduls A
$\Delta t_{1,min}$	Minimale Telegrammübertragungszeit von Modul A nach Modul B
$\Delta t_{2,min}$	Minimale Telegrammübertragungszeit von Modul B nach Modul A
Δt_{ges}	Gemessene Gesamtzeit für den Synchronisationszyklus
U	Unschärfe

Weil die Mehrzahl der in Frage kommenden Bussysteme keine Hardwareunterstützung zur Synchronisation anbieten (z. B. Ethernet, Profibus, CAN), diese Bussysteme aber unbedingt im Werkzeugmaschinenbau benötigt werden, muss die Synchronisation durch den Austausch von Zeitstempeln stattfinden. Bei Bussystemen, die eine Hardwareunterstützung bieten, kann selbstverständlich auf diese zurückgegriffen werden. Ebenso kann in Zukunft, falls es bei den verwendeten Bussystemen zu einer entsprechenden Protokollerweiterung kommt, auch auf ein hardwareunterstütztes Verfahren zurückgegriffen werden.

Zur Synchronisation mittels Austausch von Zeitstempeln wird innerhalb des Steuerungssystems ein Steuerungsmodul als Zeitserver definiert. Alle anderen Module fordern in bestimmten Abständen Zeitstempel von diesem Modul an. Dabei messen diese Module die Zeit zwischen dem Absenden der Anforderung

und dem Eintreffen der Antwort. Zieht man noch bekannte Mindestverzögerungszeiten des Kommunikationssystems entsprechend Bild 6-3 ab, so kann man den Zeitfehler der jeweils lokalen Systemuhr mit einer bekannten Unschärfe (Gleichung 5) ermitteln und, falls der Fehler größer der Unschärfe ist, entsprechend korrigieren (Gleichung 4). Die damit erreichbaren Genauigkeiten sind vom jeweils eingesetzten Bussystem wie auch vom eingesetzten Betriebssystem abhängig und müssen daher jeweils experimentell ermittelt werden.

6.1.6 Portierbarkeit

Die zu entwickelnde Systemsoftware soll für die im Rahmen dieser Arbeit durchzuführende, prototypische Realisierung auf einer möglichst breiten Palette von Steuerungsmodulen umsetzbar sein. Da die jeweils für die Steuerungsmodule verfügbaren Betriebssysteme sich unterscheiden, wird eine Isolationsschicht zwischen dem Betriebssystem und der Basisfunktionalität vorgesehen. Sie muss die benötigten Multitaskingfunktionen für die Systemsoftware einheitlich darstellen. Diese Funktionen werden in Form von Klassen angeboten. Im einzelnen sind das die Klassen Task, Semaphore, Mailbox und System-Clock (siehe Abschnitt 6.2.8).

Weiterhin werden für eine gute Portierbarkeit abstrakte Basisklassen zur Spezifikation der Kommunikations- und E/A-Anbindung benötigt, auf deren Basis dann spezifische Treiber für die jeweiligen Kommunikationssysteme bzw. E/A-Karten durch Ableitung und Überladung virtueller Funktionen erstellt werden können (s. Abschnitt 6.2.5 bzw. 6.2.6).

6.2 Designkonzept der Systemsoftware

Das Designkonzept für die Systemsoftware der dezentralisierten Steuerung wurde mit der Unified Modeling Language (UML, s. Abschnitt 4.2.2) nach objektorientierten Prinzipien unter Verwendung eines CASE-Tools (Computer Aided Software Engineering) modelliert. Dieses Konzept ist entsprechend der einzelnen Aufgaben in verschiedene Module gegliedert, wie dies in der Softwaretechnik allgemein üblich ist. In der Kommunikationstechnik kommen dabei in der Regel Schichtenmodelle entsprechend dem Basisreferenzmodell nach *ISO 7498 (1984)* zur Anwendung. Aufgrund der zahlreichen Kommunikationsaufgaben bei automatisierungstechnischen Systemen ist auch dort die Anwendung dieses bzw. darauf aufbauender Schichtenmodelle verbreitet (vgl. z. B. OSACA – *Pritschow u. a. 1996*).

Daher wurde für die Systemsoftware für verteilte, kooperative Steuerungen ein Schichtenkonzept entsprechend Abschnitt 6.2.1 entwickelt, welches in den nachfolgenden Abschnitten 6.2.2 bis 6.2.8 in den wesentlichen Aspekten erläutert wird. Eine detaillierte Behandlung des Designkonzepts ist aus Umfangsgründen im Rahmen dieser Arbeit jedoch nicht möglich.

6.2.1 Schichten des Designkonzepts der Systemsoftware

Das Designkonzept für die Systemsoftware ist entsprechend Bild 6-4 aus mehreren Schichten aufgebaut. Je nach Aufgabenumfang der einzelnen Module müssen unterschiedliche Schichten auf den Modulen vorhanden sein (siehe Bild 6-4). Im Folgenden sollen die Funktion der einzelnen Schichten des Softwaredesignkonzepts nach überblicksartig erläutert werden:

- Der **AL** (Application Layer) dient der Abarbeitung der individuellen Steuerungssoftware.
- Der **ACL** (Application Cooperation Layer) übernimmt die Funktionalität zur Kooperation verteilter Steuerungssoftware und dient daneben als Kooperationspartner für den TAL (siehe unten).
- Der **MML** (Message Management Layer) verteilt eingehende und ausgehende Kommunikationsaufrufe entsprechend an den ACL, den TAL (siehe unten) oder den CDL (siehe unten). Dabei werden die Kommunikationsaufrufe zu Messages gebündelt, um eine bessere Performance des Bussystems zu erreichen.
- Der **CDL** (Communication Driver Layer) kapselt die Spezifika der jeweiligen Kommunikationssysteme, so dass der Betrieb unterschiedlicher Bussysteme - auch parallel - einfach möglich ist. Dazu können mehrere CDLs existieren.
- Der **IODL** (Input-/Output Driver Layer) kapselt die Spezifika der jeweiligen Eingangs- und Ausgangstreiber.
- Der **TAL** (Tool Access Layer) bietet die benötigte Client-Funktionalität für den Zugriff auf die Download-/ Upload- und die Diagnosefunktionen an, mit der auf die ACLs anderer Steuerungsmodule zugegriffen wird.

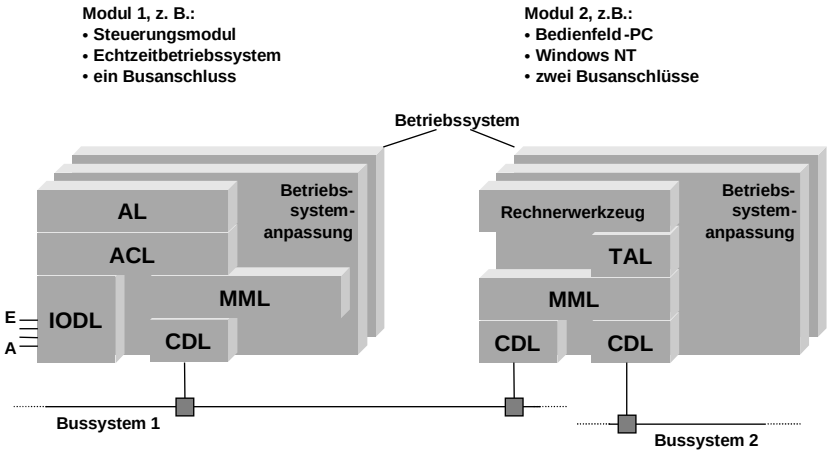

Bild 6-4: Übersicht über das Designkonzept der Systemsoftware.

Zusätzlich zu diesen Schichten ist eine Anpassungsschicht (Real Time Operating System Adaptor, **RTA**) an das Betriebssystem vorgesehen, so dass die zu entwickelnde Software einfach auf andere Betriebssysteme portiert werden kann. Insgesamt kann damit die Software schnell und aufwandsarm an verschiedene Hardware- und Kommunikationslösungen angepasst werden. Somit stellt auch der Betrieb eines Steuerungsverbundes beispielsweise mit einer Spezialhardware für die Steuerungsmodule und einem Bedienfeld-PC kein wesentliches Problem dar.

6.2.2 Application Layer

Wie in Abschnitt 6.1.1 bereits erläutert, erfolgt die Umsetzung der Netzstruktur in eine durch die Steuerungsmodule ausführbare Form durch Instanzierung und Verknüpfung von Objekten, die den Netzelementen Stelle, Transition, Kante und Marke entsprechen. Die Anweisungen aus den Entry-, Do- und Exit-Bereichen der Stellen werden ebenfalls auf Objekte abgebildet:

- **Stellenobjekte** nehmen das Marken einer oder mehrerer Farbgruppen vorübergehend auf. Die definierten Farbgruppen sowie die farbgruppenbezogene Kapazität sind Attribute der Klasse Stelle. Außerdem verfügt jede Stelle über eine Liste von Marken, in der sich ihre augenblickliche Markierungssituation widerspiegelt. Die Veränderung der Markierungssituation erfolgt über ent-

sprechende Methoden. Daneben sind für Stellen Methoden definiert, die es ermöglichen, die Markierungssituation abzufragen.

- **Transitionsobjekte** sind für das Auslösen von Schaltvorgängen verantwortlich. Dementsprechend besitzen sie Methoden zum Erkennen der Schaltfähigkeit zur Initiierung von Markenwanderungen im Netz.
- Als Verbindungselemente zwischen Transitionen und Stellen fungieren **Kantenobjekte**. Sie legen über die ihnen zugewiesene Farbgruppe und das zugehörige Kantengewicht Farbe und Anzahl der von einem Schaltvorgang betroffenen Marken fest.
- **Farbgruppenobjekte** kapseln Daten und Methoden, die es ermöglichen, Marken oder andere Farbgruppen als Elemente beziehungsweise als Teilmengen zu identifizieren.
- **Markenobjekte** tragen im Wesentlichen die Farbinformation. Sie haben aber auch eine Identität, die beispielsweise für Diagnosezwecke benutzt werden kann.
- Die Ausführung der Anweisungen, die in den Schritten spezifiziert sind, geschieht mittels **Anweisungsobjekten**. Je nach Aktivierung der Schrittobjekte werden diese Objekte abgearbeitet. Einen entsprechenden Realisierungsaufwand vorausgesetzt, könnten diese Action-Objekte auch Methoden zur Abbildung von NC-Funktionalität oder von Regelungsfunktionen enthalten.

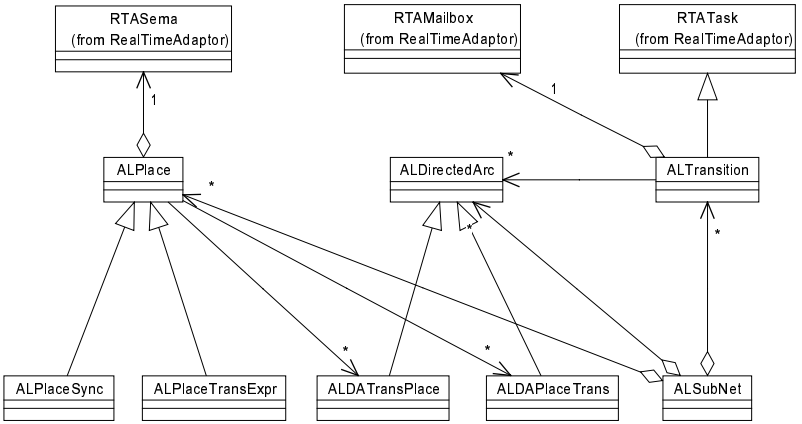

Bild 6-5: Übersicht über die wichtigsten Klassen im Application Layer und deren Anbindung an die Betriebssystemklassen.

Zusätzlich werden im AL noch **Kontrollobjekte** benötigt, die als Server für Down- und Upload-Vorgänge von Teilnetzen fungieren und die die Instanzierung und Verknüpfung der oben aufgeführten Objekte übernehmen. Auf die Zusammenhänge zwischen diesen Objekten soll im Folgenden näher eingegangen werden.

In Bild 6-5 sind die Elemente der funktionalen Sicht in einer Klassenstruktur abgebildet. Dabei werden Stellen auf `ALPlace`, Transitionen auf `ALTransition` und deren Kanten auf `ALDirectedArc` abgebildet. Die zwei Ableitungen von `ALPlace` entsprechen Synchronisationsstellen (`ALPlaceSync`) und TEP (`ALPlaceTransExpr`). Da die Methoden zur Abfrage der Aktivierungssituation von `ALDirectedArc` bereitgestellt werden, wird zwischen den Verbindungen vom Schritt zur Transition (`ALDAPlaceTrans`) und den Verbindungen von der Transition zum Schritt (`ALDATransPlace`) unterschieden. Das Teilnetz wird von `ALSubNet` verwaltet, d. h. die durch den Download bereitgestellten Parametersätze werden von diesem Objekt eingelesen und daraus `ALPlace`-, `ALTransition`-Objekte etc. bzw. deren jeweilige Ableitung instanziert. Jeder `ALPlace` aggregiert ein Semaphorenobjekt (`RTASema`), um Zugriffskonflikte durch mehrere konkurrierende Transitionen zu verhindern (siehe Abschnitt 6.1.3). Da die Rechenleistung den Transitionen zugeteilt wird, müssen diese von `RTATask` abgeleitet sein. Jede Transition aggregiert eine `RTAMailbox`, über die ihr relevante Änderungen in den angrenzenden Schritten mitgeteilt werden.

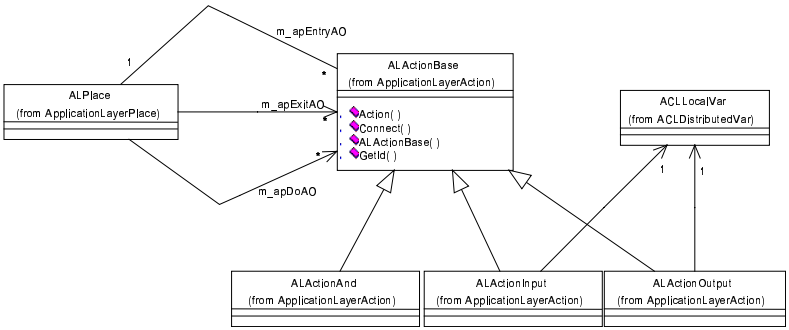

Bild 6-6: Die Anbindung der Anweisungsobjekte im Klassendiagramm.

Bild 6-6 zeigt einige von der Basisklasse der Anweisungsobjekte (`ALActionBase`) abgeleitete Anweisungsobjekte. Die Ableitungen (`ALActionAnd`, `ALActionInput`, `ALActionOutput`) führen die Schreib-, Lese- und Ver-

knüpfungsfunktionalität in den jeweiligen Überladungen der abstrakten Funktion `Action` aus. Eine Erweiterung der Funktionalität kann durch weitere Ableitungen von der Basisklasse `ALActionBase` erreicht werden. Jeder `ALPlace` besitzt drei Zeigerfelder (`m_apEntryAO`, `m_apExitAO`, `m_apDoAO`), durch die die spezifizierten Entry-, Exit- und Do-Anweisungen verknüpft werden. Die Anweisungsobjekte greifen dabei auf Lokalvariablen (`ACLLocalVar`) zu, die die Verbindung zu den - eventuell auf anderen Steuerungsmodulen befindlichen - physikalischen Ein- und Ausgängen oder zu anderen Zentralvariablen herstellen (näheres siehe Abschnitt 6.2.3).

6.2.3 Application Cooperation Layer

Der Application Cooperation Layer übernimmt die Funktionalität zur Kooperation der auf die Application Layers mehrerer Steuerungsmodule verteilter Steuerungssoftware. Zusätzlich dient er für den Tool Access Layer als Vermittler für Zugriffe auf die Objekte des AL.

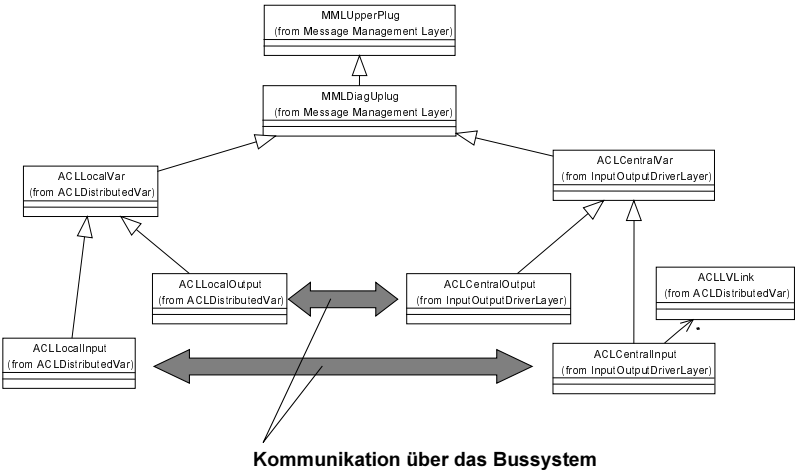

Bild 6-7: Verteilte Variablen im Klassendiagramm.

In Bild 6-7 ist das Klassendiagramm der verteilten Variablen dargestellt. Lokalvariablen werden dabei durch die Ableitungen `ACLLocalInput` und `ACLLocalOuput` von der Basisklasse `ACLLocalVar` abgebildet. Sie werden vom AL mittels `ALActionInput`- und `ALActionOutput`-Objekten gelesen bzw.

beschrieben. Diese Klassen stellen die Verbindung zu den entsprechenden Zentralvariablen (`ACLCentralInput`, `ACLCentralOutput`) her, welche von `ACLCentralVar` abgeleitet sind.

`ACLCentralVar`-Objekte sind entweder Variablen im Sinne eines Merkers, oder sie sind den physikalischen Ein- und Ausgängen zugeordnet. In letzterem Fall befinden sie sich auf denjenigen Modulen, an denen die jeweiligen Ein- und Ausgänge angeschlossen sind. Dabei wird über die Kommunikation eine Beziehung entweder zwischen einem `ACLLocalOutput` und einem `ACLCentral-Output` oder zwischen einem `ACLCentralInput` und einem oder mehreren `ACLLocalInput` hergestellt. Die Kommunikation ist dabei ereignisorientiert konzipiert, daher wird nur bei Änderung eines `ACLCentralInput`-Objekts bzw. eines `ACLLocalOutput`-Objekts ein Kommunikationsaufruf notwendig.

Merkervariablen, die im Gegensatz zu Ein- und Ausgängen gelesen und geschrieben werden können, werden durch eine Kombination von Ein- und Ausgangsobjekten realisiert. Ebenso wird dies bei Timern gehandhabt, bei denen der Eingang zum Setzen der Zeiteinstellung dient und der Ausgang den Ablauf der Zeit anzeigt.

Die Anbindung dieser Klassen für den Nachrichtenempfang geschieht dabei durch die Ableitung von der Klasse `MMLUpperPlug` über die Klasse `MMLDiagUPlug`, die eigens zu diesem Zweck vom MML (siehe Abschnitt 6.2.4) bereitgestellt werden. `MMLDiagUpperPlug` stellt dabei die Funktionalität der Diagnose durch entsprechende Methoden zur Verfügung.

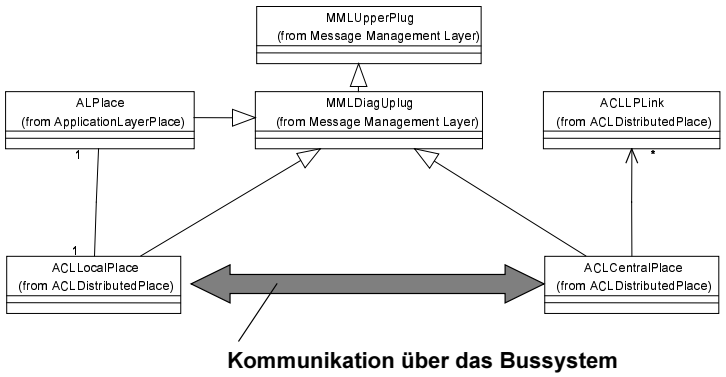

Bild 6-8: Verteilte Stelle im Klassendiagramm.

Bild 6-8 zeigt das Klassendiagramm einer verteilten Stelle. Darin werden Lokalstellen auf die Klasse `ACLLocalPlace` und Zentralstellen auf die Klasse `ACLCentralPlace` abgebildet. Für jedes an einer verteilten Stelle beteiligtes Teilnetz wird dabei ein `ACLLocalPlace`–Objekt instanziert, welches sich dann mit dem zugehörigen `ACLCentralPlace`-Objekt verbindet. Der `ACLCentralPlace` koordiniert den Zustand aller angeschlossenen `ACLLocalPlace`-Objekte. Dabei sind in den Klassen `ACLCentralPlace` und `ACLLocalPlace` entsprechende Zugriffsschutzmechanismen implementiert, so dass Fehler im Kontrollfluss zuverlässig vermieden werden.

6.2.4 Message Management Layer

Der Message Management Layer dient zur Verteilung eingehender und ausgehender Kommunikationsaufrufe an den ACL, den TAL bzw. den CDL. Ebenfalls wird im MML die Funktionalität der Kommunikationsbündelung und –priorisierung (s. Abschnitt 6.1.4) abgebildet.

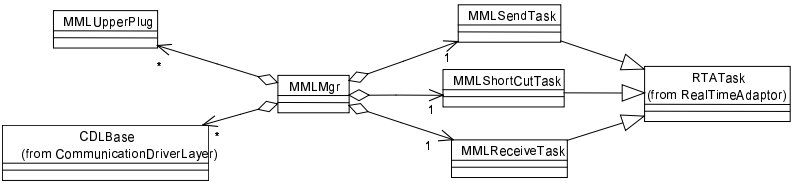

Bild 6-9: Klassendiagramm des MML.

In Bild 6-9 ist das grundlegende Klassendiagramm des MML abgebildet. `MMLMgr` ist die zentrale Klasse des MML. Sie verwaltet die Nachrichtenempfänger im ACL bzw. im TAL. Dazu stellt sie eine Basisklasse `MMLUpperPlug` zur Verfügung, von denen die verschiedenen Nachrichtenempfänger die Eigenschaften für den Nachrichtenempfang erben. Dies geschieht durch Überladung der Funktion `ReceiveMessage`, die vom MML aufgerufen wird, sobald eine Message für den jeweiligen `MMLUpperPlug` eingeht. Dadurch können die verschiedenen Arten von Nachrichtenempfängern in jeweils eigenen `ReceiveMessage`-Funktionen die eingehenden Nachrichten entsprechend auswerten.

Im MML sind die drei Tasks `MMLSendTask`, `MMLReceiveTask` und `MMLShortCutTask` enthalten. `MMLSendTask` versendet Nachrichten an andere Module und übernimmt dabei die Nachrichtenbündelung und die Zeitsteuerung. Die ausgehenden Telegramme werden dabei an das - je nach Bussystem –

zuständige, von der Basisklasse `CDLBase` abgeleitete CDL-Objekt (siehe Abschnitt 6.2.5) zum Versand weitergeleitet. `MMLReceiveTask` nimmt Nachrichten vom CDL entgegen und teilt die darin enthaltenen Nachrichtenpakete auf die Empfänger auf. `MMLShortCutTask` leitet Nachrichten, die zu einem anderen Empfänger desselben Moduls gesendet werden an diesen durch, ohne dass dazu tiefere Kommunikationsschichten belastet werden müssen

6.2.5 Communication Driver Layer

Der CDL dient der Kapselung der Spezifika der jeweils verwendeten Kommunikationssysteme und stellt damit eine einheitliche Schnittstelle für den MML dar.

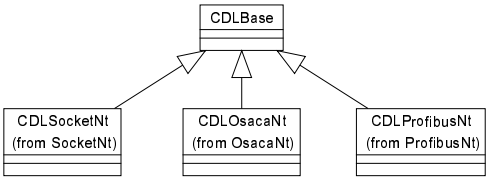

Bild 6-10: Klassendiagramm des CDL

In Bild 6-10 ist der Aufbau des CDL dargestellt. `CDLBase` ist die zentrale Klasse des CDL, mittels der sämtliche Spezialisierungen auf bestimmte Kommunikations- und Betriebssystemkombinationen durch Ableitung integriert werden können. Im Rahmen der Arbeit wurden von `CDLBase` die Klassen `CDLSocketNt`, `CDLOsacaNt` und `CDLProfibusNt` abgeleitet. Je nachdem welches Kommunikations- und Betriebssystem zum Einsatz kommt, wird die entsprechende Klasse eingesetzt. `CDLSocketNt` enthält die Spezifika der Socket-Schnittstelle. Dies ist zur Zeit die gebräuchlichste API (Application Programming Interface) für TCP/IP (siehe z. B. *Furrer 1998*). `CDLOsacaNt` ermöglicht die Kommunikation nach dem OSACA-Standard, `CDLProfibusNt` den Einsatz von Profibus.

In `CDLBase` sind alle Funktionen des CDL deklariert. Der Zugriff des MML auf den CDL erfolgt nur über diese Funktionen, die von den abgeleiteten Klassen überladen werden, um die spezifische Funktionalität des entsprechenden Kommunikationssystems zu implementieren. Damit bleibt die spezifische Funktionalität des jeweiligen Kommunikationssystems gegenüber dem MML verborgen. Zur Einbindung neuer Kommunikationssysteme muss daher nur eine entsprechend abgeleitete Klasse entwickelt werden.

6.2.6 Input / Output Driver Layer

Der Input / Output Driver Layer dient analog zum CDL der Kapselung von Spezifika der jeweils verwendeten E/A-Hard-und Software und stellt somit eine einheitliche Schnittstelle für den ACL zur Verfügung.

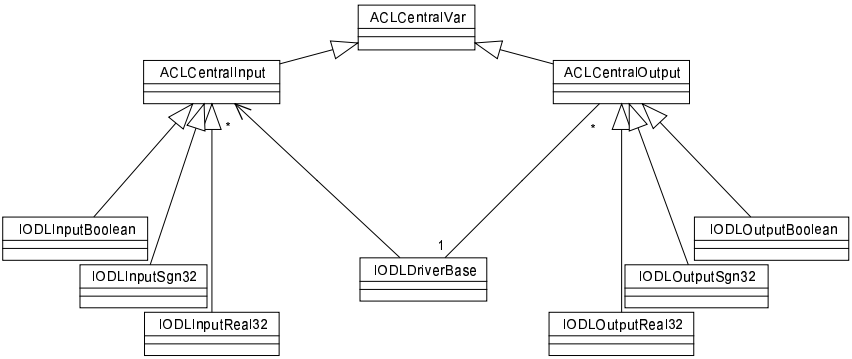

Bild 6-11: Klassendiagramm des IODL

In Bild 6-11 ist das Klassendiagramm des IODL dargestellt. `ACLCentralVar` fungiert dabei als zentrale Basisklasse des IODL und gleichzeitig als Schnittstelle zum ACL. Die Ein- bzw. Ausgänge eines Steuerungsmoduls werden jeweils einem `ACLCentralInput`- bzw. `ACLCentralOutput`-Objekt zugeordnet. `ACLCentralVar` stellt diese Ein- und Ausgänge auch anderen Steuerungsmodulen zur Verfügung (siehe hierzu Abschnitt 6.2.3). Je nachdem, welchem Variablentyp der Ein- bzw. Ausgang zugeordnet ist, wird dafür eine der weiteren Ableitungen dieser Klassen benutzt.

Die E/A-Karten der Steuerungsmodule haben in der Regel kartenspezifische Treiber. Die Klasse `IODLDriverBase` kapselt diese Treiberfunktionalität und bildet die Eingänge auf von der Klasse `ACLCentralInput` abgeleitete Objekte ab bzw. schreibt die Werte von `ACLCentralOutput` abgeleiteter Objekte auf die Ausgänge der Karte. Analog zu `CDLBase` (siehe Abschnitt 6.2.5) werden auch hier die Funktionen von `IODLDriverBase` durch die spezifische Funktionalität zur Einbindung der jeweiligen E/A-Karte überladen.

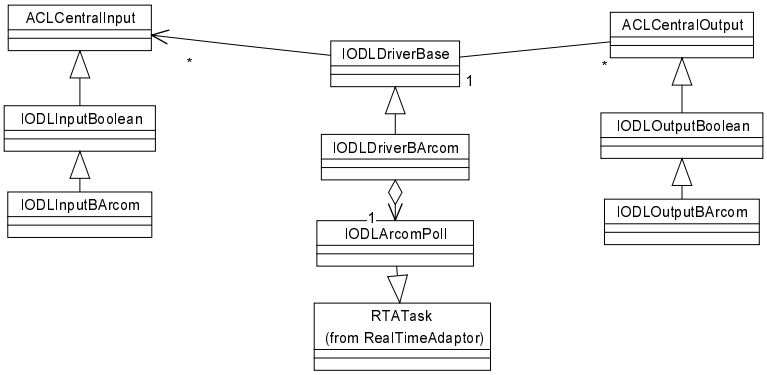

Bild 6-12: Klassendiagramm für eine konkrete Ein-/Ausgabehardware

Bild 6-12 zeigt die im Projekt realisierte Klassenstruktur der E/A-Anbindung an einem Beispiel. Als E/A-Karte wurde eine spezifische Digital-IO-Karte benutzt. Die hierfür entwickelte Treiberfunktionalität ist in `IODLDriverBArcom`, einer Ableitung der Basisklasse `IODLDriverBase`, enthalten. Dieser Treiber instanziert eine Task (`IODLArcomPollTask`), da die Karte keine Interrupts unterstützt und daher pollendes Einlesen notwendig ist. Da die Karte nur boolesche Ein- und Ausgänge liefert, genügen `IODLInputBArcom` und `IODLOutput-BArcom` als entsprechend spezialisierte Ein- und Ausgangsklassen.

6.2.7 Tool Access Layer

Der Tool Access Layer stellt die Client-Funktionalität für den Zugriff auf die Softwaremanagement- und Diagnosefunktionen zur Verfügung. Bild 6-13 zeigt das Klassendiagramm des Software- und des Diagnosemanagers. Für den Zugriff auf die Softwaremanagementfunktionalität greift das Rechnerwerkzeug auf die Funktionen der Klasse `TALSubNetLink` zu. Jedem Teilnetz und damit jedem `ALSubNet`-Objekt der gesamten Steuerung kann ein solches `TALSubNet-Link`-Objekt zur Herstellung einer Softwaremanagementverbindung zugeordnet werden. Damit ist zum einen ein zentraler Down- / Upload möglich, zum anderen muss wegen der separaten Handhabung der Teilnetze der Down- / Upload nicht für ein gesamtes Steuerungsmodul durchgeführt werden. Die Anbindung für den Nachrichtenempfang geschieht dabei durch die Ableitung von der Klasse `MMLUpperPlug` (siehe Abschnitt 6.2.4).

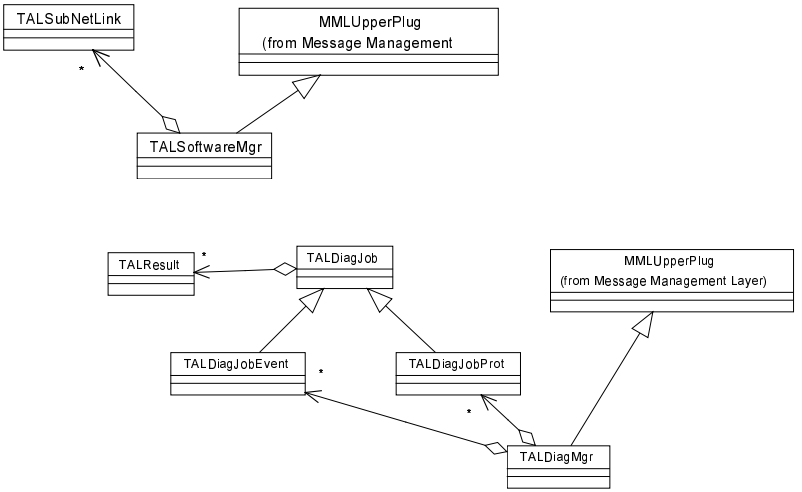

Bild 6-13: Das Klassendiagramm des TAL.

Der Diagnosemanager ist ähnlich aufgebaut. Das Rechnerwerkzeug greift auf Objekte der Klasse `TALDiagJob` zu, die die benötigte Funktionalität für den Zugriff auf Variablen und Stellen enthält. Die Diagnoseergebnisse können entweder auf dem Steuerungsmodul lokal protokolliert und dann gemeinsam zum Bedienfeldrechner übertragen werden oder jede Änderung eines Diagnoseelements wird sofort zum Bedienfeldrechner übertragen. Zur Realisierung dieser unterschiedlichen Verhaltensweisen sind die Klassen `TALDiagJobProt` für die Protokollierungsfunktion und `TALDiagJobEvent` für einfache Visualisierungsfunktionen wie eine Oszilloskopfunktion von der Basisklasse `TALDiagJob` abgeleitet. In Objekten der Klasse `TALResult` werden die Ergebnisse zur Abholung durch das Diagnose- oder Visualisierungswerkzeug zwischengespeichert. Die Klasse `TALDiagMgr` erfüllt einen zur Klasse `TALSoftwareMgr` vergleichbaren Zweck.

6.2.8 Real Time Operating System Adaptor

Der RTA dient zur Ermöglichung einer einfachen Portierbarkeit der Systemsoftware auf andere Multitaskingsysteme. Innerhalb der Implementierung der Systemsoftware wird daher niemals direkt auf Funktionen des Multitaskingsystems zugegriffen. Statt dessen bietet der RTA Klassen für alle in der Systemsoftware benötigten Multitaskingfunktionalitäten an. Dies sind die Klassen für die Task (`RTATask`), die Semaphore (`RTASema`), die Mailbox (`RTAMailbox`) und die

Systemuhr (RTASystemClock). Bei einem Übergang auf ein anderes Multitaskingsystem ist es daher ausreichend, die Implementierung dieser Klassen entsprechend anzupassen. Eine besondere Rolle nimmt dabei die Klasse RTATask ein, die als Basisklasse für alle mit eigener Rechenleistung ausgestatteten Klassen der Systemsoftware dient. Die Funktionalität der jeweiligen Task wird dabei durch Überladung der abstrakten Funktion TaskMethod implementiert.

6.3 Abbildung der Systemsoftwarefunktionen

6.3.1 Abarbeitung der Steuerungssoftware

Da die Systemsoftware vollständig ereignisorientiert aufgebaut ist, sind alle Transitionstasks blockiert, solange sich keine Änderungen der Eingangssignale, Timerüberläufe oder als Folge von letzterem Änderungen der Markierungssituation oder von Variablenwerten ergeben. Die Aktivität der Systemsoftware geht also vom IODL aus, wobei dies je nach verwendeter Hardware durch einen Interrupt oder eine zyklische Lesetask für die Eingänge realisiert wird. Nach jeder derartigen Eingangssignalveränderung bleibt die Systemsoftware solange aktiv, wie durch das Weiterschalten von Marken oder die Veränderung von Variablen Transitionen schaltfähig werden.

Verändert ein Eingang seinen Wert, so wird dies von einem von IODLDriverBase hardwarespezifisch abgeleiteten Objekt per Funktionsaufruf dem entsprechenden ACCentralInput-Objekt angezeigt. Durch die Kommunikationsmechanismen des ACL wird diese Veränderung dann den zugehörigen ACLLocalInput-Objekten angezeigt.

Daraufhin ruft ACLLocalInput die Funktion Trigger bei den betroffenen ALPlaceTransExpr-Objekten auf, die dann die Listen der Anweisungsobjekte (ALActionBase) ihrer Entry-Sektion abarbeiten. Dabei können beliebige Eingänge und Variablen eingelesen und miteinander verknüpft werden. Das Verknüpfungsergebnis wird anschließend als boolescher Wert interpretiert und entsprechend zur Markierung des jeweiligen ALPlaceTransExpr-Objekts verwendet. Hat sich die Markierung des ALPlaceTransExpr dabei verändert, so wird eine Nachricht in der Mailbox des angeschlossenen ALTransition-Objekts abgelegt.

Das ALTransition-Objekt wartet in seiner Task-Funktion (TaskMethod) an dieser Mailbox auf Nachrichten, wird also durch die eintreffende Nachricht aktiv. Sollte diese Nachricht von demjenigen ALPlace-Objekt stammen, dessen

Markierung zuletzt ein Schalten verhindert hat, wird eine erneute Schaltfähigkeitsüberprüfung durchgeführt, andernfalls wird die Nachricht ignoriert.

Die Schaltfähigkeitsüberprüfung geht beginnend mit dem `ALPlace`-Objekt, von dem die Nachricht empfangen wurde, alle verbundenen `ALPlace`-Objekte durch, wobei die Schaltfähigkeitsüberprüfung bei der ersten fehlenden Bedingung sofort wieder abgebrochen wird und die Task-Funktion der `ALTransition` wieder an der Mailbox auf eine relevante Markierungsänderung wartet.

Kann die Schaltfähigkeitsüberprüfung dagegen erfolgreich abgeschlossen werden, so wird zunächst bei allen vorhergehenden `ALPlace`-Objekte die Markierung entfernt und dabei deren Exit-Sektion abgearbeitet. Anschließend werden durch die Transition die nachfolgenden Schritte aktiviert und dabei deren Entry-Sektion durchlaufen. Die Transition beginnt anschließend sofort eine neue Schaltfähigkeitsüberprüfung, die im Normalfall ein negatives Ergebnis erbringt. Daraufhin blockiert die Transitionstask wieder an ihrer Mailbox und gibt dadurch den Prozessor wieder frei.

6.3.2 Transparente Kommunikation

Wird nun die individuelle Steuerungssoftware auf mehrere Steuerungsmodule verteilt, so wird, wie schon in Abschnitt 6.2.3 erläutert, das Petrinetz aufgetrennt und die Stellen, auf die von anderen Modulen aus zugegriffen werden soll, durch jeweils ein `ACLCentralPlace`- und je Modul ein damit verbundenes `ACLLocalPlace`-Objekt ersetzt. Dabei fungieren die `ACLLocalPlace`-Objekte als Spiegelbilder des zugeordneten `ACLCentralPlace`-Objekts, welches die Koordination der `ACLLocalPlace`-Objekte übernimmt. Das Verfahren, durch das der somit aufgebaute, verteilte Schritt unabhängig von seinem Verteilungsgrad koordiniert wird, soll im Folgenden anhand Bild 6-14 erläutert werden.

Die Markierungsänderung einer verteilten Stelle beginnt zunächst wie in Abschnitt 6.3.1 mit der Schaltfähigkeitsüberprüfung einer angrenzenden Transition. Durch die lokale Abbildung der Markierung der verteilten Stelle muss dazu keine Kommunikation erfolgen. Erst wenn die Schaltfähigkeit festgestellt ist, wird eine Kommunikation notwendig, um den Schaltvorgang ordnungsgemäß durchführen zu können. Durch dieses Verfahren kann das Kommunikationssystem erheblich entlastet werden.

Als nächste Aktion erfolgt das Einholen der Freigabe für das Schalten des verteilten Schritts. Falls noch kein anderer lokaler Schritt die Freigabe angefragt hat, wird die Freigabe bestätigt und der Schaltvorgang eingeleitet, andernfalls wird die Freigabe entweder zurückgestellt oder verweigert. Sobald die Freigabe durch

den ACLLocalPlace erfolgt ist, kann der Schaltvorgang. Nachdem das lokale Schalten beendet ist, wird die Aktualisierung des verteilten Schritts durch ACL-CentralPlace veranlasst.

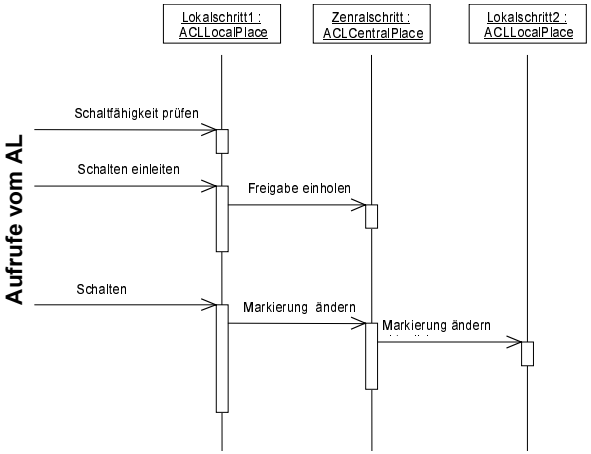

Bild 6-14: Ablauf eines Schaltvorgangs mit einer verteilten Stelle.

Durch dieses Verfahren wird sichergestellt, dass auch bei einer beliebig komplex verteilten Stelle keine Inkonsistenz durch konkurrierende Zugriffe von Transitionen entstehen kann. Deshalb kann mit dem Konzept der verteilten Stelle ohne Einschränkungen jede beliebige Verteilung realisiert werden. Der Steuerungsentwickler braucht bei der Verteilung also keinerlei Restriktionen beachten.

Ein ähnliches, aber deutlich einfacheres Verfahren wurde auch zur Abbildung verteilter Variablen mittels der von ACLCentralVar und ACLLocalVar abgeleiteten Klassen implementiert.

6.3.3 Softwaremanagementfunktionen

Beim Einschalten des Steuerungsmoduls wird zunächst für jedes ALSubNet-Objekt festgestellt, ob Initialisierungsdaten für die entsprechende Teilfunktion bereits vorhanden sind. Ist dies nicht der Fall, so geht das ALSubNet-Objekt in den Zustand LAZY über. Andernfalls werden automatisch die Zustandsübergänge Aggregate, Initialize, Connect und Start durchgeführt und damit der Zustand RUNNING eingenommen, in dem die Teilfunktion ausgeführt wird.

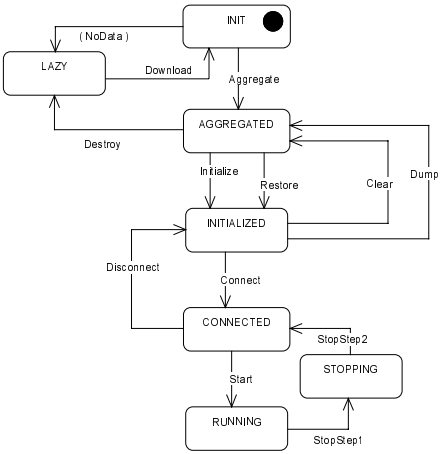

Bild 6-15 Zustandsgraf der Klasse `ALSubNet`.

In Bild 6-15 ist der Zustandsgraf der Klasse `ALSubNet` dargestellt. Im Folgenden sollen die oben angesprochenen Funktionen kurz erläutert werden:

- `Aggregate` legt die Objekte der Klassen `ALPlace`, `ALTransition`, `ALDirectedArc`, `ALAction`, `ACLLocalVar` und `ACLLocalPlace` entsprechend der vorliegenden Parametersätze an.
- `Initialize` stellt die Initialaktivierung der `ALPlace`-Objekte her.
- `Connect` verknüpft die genannten Steuerungsobjekte entsprechend der spezifizierten Funktionalität miteinander und verbindet die `ACLLocalVar`- und `ACLLocalPlace`-Objekte mit den zugehörigen `ACLCentralVar`- und `ACLCentralPlace`-Objekten.
- `Start` startet die Abarbeitung des Teilnetzes.

Ein Download kann nur durchgeführt werden, wenn sich das `ALSubNet`-Objekt im Zustand `LAZY` befindet, d. h. wenn entweder für dieses Objekt lokal keine Initialisierungsdaten vorhanden sind oder wenn das Teilnetz durch die Funktionen `StopStep1`, `StopStep2`, `Disconnect` und `Destroy` gelöscht worden ist.

- `StopStep1` bewirkt, dass diejenigen Transitionen der Teilfunktion, die in einen Bewegungsschritt – also zu einer Bewegung der Maschine – führen blockiert werden. Dadurch werden die durch die Teilfunktion angesteuerten Maschinenkomponenten angehalten.

- `StopStep2` wird durch das Steuerungsmodul automatisch ausgeführt, sobald alle Bewegungsschritte verlassen worden sind. Optional kann diese Funktion – beispielsweise im Falle einer Störung – auch durch das Rechnerwerkzeug ausgelöst werden.
- `Disconnect` trennt die Verbindungen zwischen den Objekten des Teilnetzes und zu den zugehörigen `ACLCentralVar`- und `ACLCentralPlace`-Objekten.
- `Destroy` entfernt die Objekte des Teilnetzes wieder aus dem Arbeitsspeicher. Zusätzlich werden die Parameterfiles des Teilnetzes gelöscht.

Sobald der Download abgeschlossen ist, geht das `ALSubNet`-Objekt in den Zustand `INIT` über. Anschließend kann der eingangs beschriebene Ablauf durchgeführt werden, bis sich `ALSubNet` im Zustand `RUNNING` befindet. Dieser Ablauf wird in diesem Fall jedoch nicht automatisch durch das Steuerungsmodul durchgeführt, sondern wird über Befehle des Rechnerwerkzeugs explizit gesteuert.

Die Funktionen `Dump` und `Restore` der Klasse `ALSubNet` sind vorgesehen, um eine vorliegende Markierung des Netzes zu speichern (`Dump`), um sie beispielsweise nach einem Modultausch durch `Restore` wieder herstellen zu können. Ein Upload der vorliegenden Initialisierungsdaten kann anders als die bisher erläuterten Funktionen immer durchgeführt werden, sofern sich `ALSubNet` nicht im Zustand `LAZY` befindet.

6.3.4 Diagnosefunktionen

Die Diagnosefunktionalität der Systemsoftware umfasst einerseits Funktionen zur Diagnose der modellierten Steuerungsfunktionen und andererseits Funktionen zur Diagnose von Fehlfunktionen der Steuerungshardware. Für die Diagnose von Ein- und Ausgängen sowie von Variablen besteht zum einen die Möglichkeit einer Online-Visualisierung (Oszilloskopfunktion) und zum anderen die einer Offline-Protokollierung (Protokollfunktion).

Bei der Online-Visualisierung wird in der funktionalen Sicht der gewünschte Ein- oder Ausgang bzw. die gewünschte Variable ausgewählt und deren Modul- und E/A-Adresse festgestellt. Dann wird mit Hilfe eines `TALDiagJobEvent`-Objekt über den `TALDiagMgr` eine Verbindung zu dem entsprechenden `ACLCentralVar`-Objekt aufgebaut. Mit der Funktion `EventOn` des `TALDiagJobEvent`-Objekts wird über das Kommunikationssystem beim `ACLCentralVar`-Objekt die Event-übertragung gestartet. Damit wird jede Änderung sofort zum `TALDiagJobEvent`-Objekt übertragen und wird dort zeitbezogen

in einem `TALResult`-Objekt zwischengespeichert um zur Anzeige zur Verfügung zu stehen.

Die Offline-Protokollierung funktioniert ähnlich wie die Online-Visualisierung mit dem Unterschied, dass die Änderungen auf dem Steuerungsmodul lokal zwischengespeichert und zu einem späteren Zeitpunkt durch das Rechnerwerkzeug abgerufen werden. Da während der Protokollierung keine zusätzliche Kommunikation ausgelöst wird, wird das Reaktionszeitverhalten der Steuerung kaum beeinflusst. Außerdem kann für die Zeitmessung die lokale, durch den Mechanismus aus Abschnitt 6.1.5 synchronisierte Systemuhr benutzt werden. Dadurch können präzisere Aussagen über das Reaktionszeitverhalten der Steuerung gemacht werden.

Neben den Ein-/Ausgängen und Variablen besteht auch für die Markierung von Stellen die Möglichkeit einer Online-Visualisierung oder Offline-Protokollierung. Dadurch kann der Ablauf des Netzes visualisiert und ausgetestet sowie das Reaktionszeitverhalten der Steuerung detaillierter analysiert werden. Für eine weitere Unterstützung des Funktionstests der Steuerung können darüber hinaus die `ALTransition`-Objekte einzeln gesperrt werden, was mit der Funktionalität von Breakpoints in der konventionellen Softwareentwicklung (siehe Debugging in C++ Entwicklungsumgebungen) vergleichbar ist. In einem Einzelschrittbetrieb kann jede einzelne Transition auch für jeweils genau einen Schaltvorgang freigegeben werden.

Die Diagnose des Kommunikationssystems geschieht über verbindungsspezifische Telegrammzähler für gesendete und empfangene Telegramme. Damit kann ermittelt werden, ob das Kommunikationssystem fehlerfrei arbeitet. Diese Telegrammzähler werden von der Klasse `CDLBase` zur Verfügung gestellt. Etwaige Diskrepanzen können dann durch das Diagnosewerkzeug in der topologischen Sicht dargestellt werden.

Darüber hinaus können Auswirkungen eines eventuellen Kommunikationsfehlers durch die Analyse der Werte zusammengehöriger Zentral- (`ACLCentralVar`) und Lokalvariablen (`ACLLocalVar`) bzw. der Aktivierung zusammengehöriger Zentral- (`ACLCentralPlace`) und Lokalschritte (`ACLLocalPlace`) festgestellt und durch einen manuellen Eingriff mit dem Diagnosewerkzeug korrigiert werden.

6.4 Zusammenfassung

Im vorliegenden Kapitel wurde ausgehend von grundsätzlichen Überlegungen zum Aufbau der Systemsoftware (Abschnitt 6.1) ein auf Schichten basierendes

Designkonzept für die Systemsoftware (Abschnitt 6.2) vorgestellt. Das Zusammenwirken dieser Schichten ist in Abschnitt 6.3 hinsichtlich der Abarbeitungsfunktionalität, der transparenten Kommunikation, der Down - / Upload- und der Diagnosefunktionalität beschrieben.

Mit diesem Designkonzept können einerseits die Leistungspotenziale verteilter, kooperativer Steuerungen ausgeschöpft werden. Dies ist gleichzeitig eine wesentliche Voraussetzung zur Nutzung der Kostenpotenziale. Andererseits unterstützt eine nach diesem Softwaredesignkonzept aufgebautes Steuerungssystem in nahtlos eine Werkzeugunterstützung im Sinne der in Kapitel 5 vorgeschlagenen Konzepte und kann so entscheidend zur Beherrschbarkeit verteilter, kooperativer Steuerungssysteme beitragen.

7 Anwendungsbeispiel und Bewertung

In Abschnitt 7.1 wird zunächst die prototypische Realisierung einer steuerungstechnischen Plattform auf der Basis des im vorangegangenen Kapitel erarbeiteten Softwaredesignkonzepts vorgestellt. Darauf aufbauend wird in Abschnitt 7.2 in einem Anwendungsbeispiel an einer simulierten Werkzeugmaschine entsprechend des in Kapitel 5 vorgeschlagenen Vorgehens und der dort beschriebenen Modellierungstechnik eine Steuerungsaufgabe exemplarisch gelöst, die dann auf der Steuerungsplattform umgesetzt wird. In diesem Beispiel kann des weiteren die erreichte Leistungssteigerung durch lokale Reaktionen und direkte Kooperation abgeschätzt werden. Anhand des Anwendungsbeispiels und der damit ermittelten Eckwerte für den Steuerungshardwareaufwand wird schließlich in Abschnitt 7.3 die entwickelte verteilte, kooperative Steuerungstechnik für maschinennahe Abläufe bewertet. Dabei wird auch auf die monetär schwer bewertbaren Vorteile der vorgestellten Konzepte kurz eingegangen.

7.1 Aufbau der steuerungstechnischen Plattform

Bild 7-2 zeigt den Aufbau des prototypisch realisierten, verteilten, kooperativen Steuerungssystems. Es besteht aus drei Steuerungsmodulen, wobei bewusst ein heterogenes Kommunikationssystem zur Verbindung gewählt wurde. Zwei dieser Module kommunizieren dabei untereinander und mit dem Entwicklungs- und Visualisierungsrechner über Ethernet/TCP/IP. Das dritte Modul (in der Abbildung links) ist über Profibus an eines der beiden anderen Steuerungsmodule angeschlossen. Die Topologie dieses Aufbaus orientiert sich an der Topologie aus Abschnitt 5.3.3, die hier nochmals zum Vergleich abgebildet ist (in der Abbildung links oben).

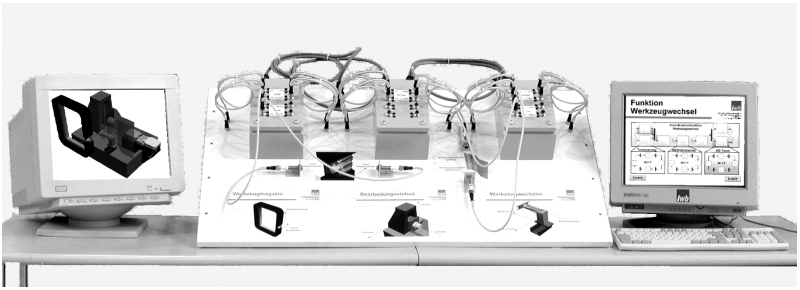

Bild 7-1: Ansicht des am iwb aufgebauten Versuchs- und Demonstrationsstands für verteilte, kooperative Steuerungen.

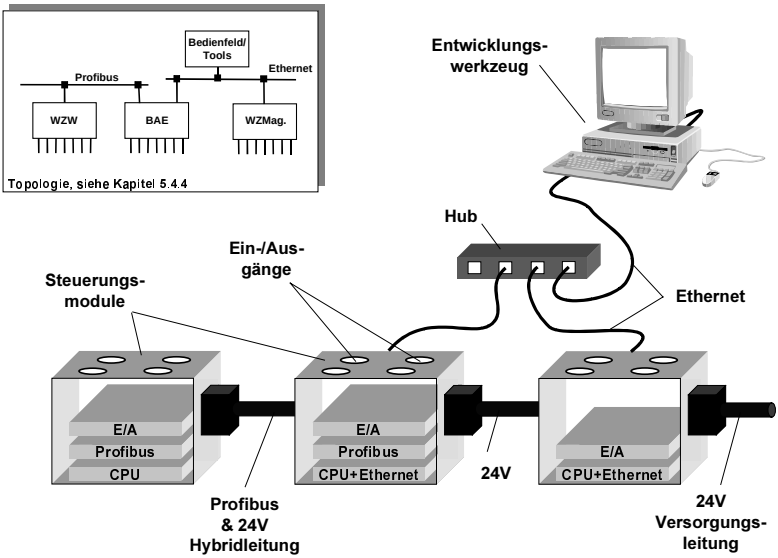

Bild 7-2: Aufbau der steuerungstechnischen Versuchsplattform.

Der am iwb entsprechend Bild 7-2 aufgebaute Versuchs- und Demonstrationsstand ist im folgenden Bild 7-1 abgebildet. In der Mitte sind die drei Steuerungsmodule zu sehen, rechts befindet sich der Rechner, mit dem die Funktionalität des Entwicklungswerkzeugs abgebildet wird. Links im Bild ist ein Rechner mit der simulierten Beispielmaschine zu erkennen (siehe auch Abschnitt 7.2.1). Bild 7-3 zeigt eine Detailansicht eines der realisierten Steuerungsmodule dieses Versuchsstands.

Als Betriebssystem für die Steuerungsmodule wird Windows 95 eingesetzt. Da im Softwaredesignkonzept auf die Portierbarkeit der Software (siehe Abschnitt 6.1.6) ein hoher Wert gelegt wurde, kann leicht auf ein aus Echtzeitgesichtspunkten besser geeignetes Betriebssystem - z. B. QNX oder VxWorks - umgestellt werden.

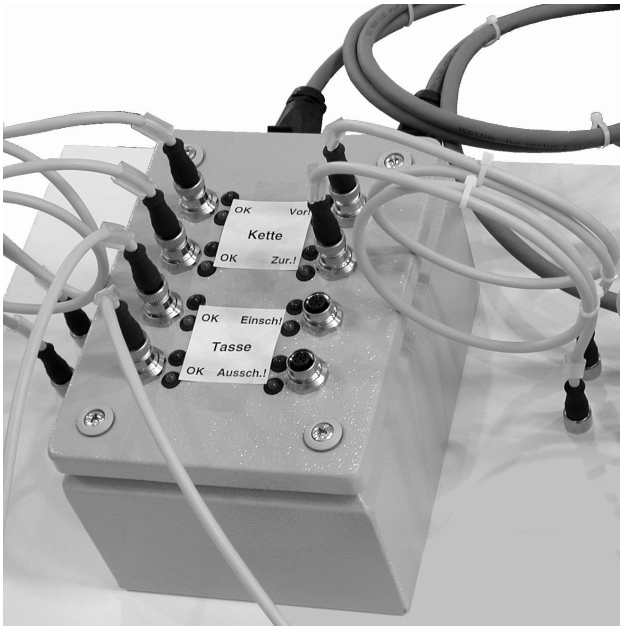

Bild 7-3: Detailansicht eines Steuerungsmoduls des Versuchs- und Demonstrationsstands.

Die Komponenten der realisierten Steuerungsmodule sind in Bild 7-4 dargestellt. Die Auswahl dieser Komponenten erfolgte in Hinblick auf genügende Prozessorleistung und ausreichenden Arbeits- (RAM) und Permanentspeicher (Flash-Disk), um eine effiziente Entwicklung der Systemsoftware und ein komfortables Debugging zu ermöglichen. Testläufe der Software haben jedoch ergeben, dass zur Realisierung solcher Module – entsprechende Optimierungen der Software vorausgesetzt – beispielsweise ein Prozessor 386/40, 2 MB RAM und 2 MB Flash-Speicher ausreichend sind. Durch Spezialarchitekturen können die Kosten solcher Module noch weiter gesenkt werden. Ein Beispiel für ein aus hardwaretechnischer Sicht geeignetes Modul ist die für ca. 700 DM verfügbare, in einem feldfähigen Modulgehäuse untergebrachte Steuerung ET200X mit integrierter SPS.

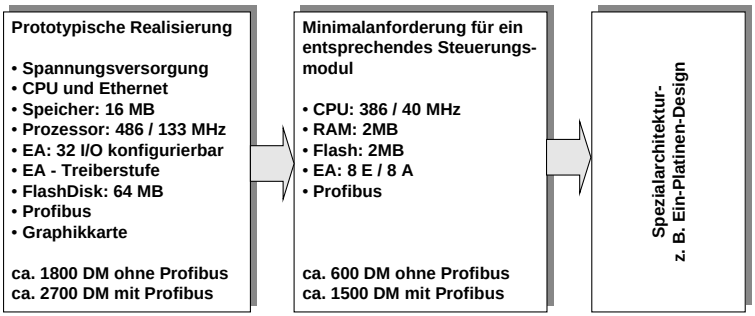

Bild 7-4: Aufbau und Kosten eines Prototypmoduls im Vergleich zu den ermittelten Minimalanforderungen. Weitere Optimierungen kann eine Spezialarchitektur bringen.

7.2 Anwendungsbeispiel

7.2.1 Einsatzumgebung

Als Einsatzbeispiel der entwickelten Konzepte sowie der implementierten Systemsoftware wurde die Werkzeugwechselfunktion des am *iwb* für Versuchszwecke zur Verfügung stehenden Horizontalfräsbearbeitungszentrums Heckert CWK 400 (Bild 7-5) gewählt.

Zu Versuchs- und Demonstrationszwecken wurde diese Maschine mit einem 3D-Simulationssystem nachgebildet. Durch den Einsatz einer Simulationsumgebung können einerseits Testszenarien wesentlich flexibler gestaltet werden und andererseits besteht bei einer eventuellen Fehlfunktion nicht die Gefahr eines Maschinenschadens. Bild 7-6 zeigt die in der Simulation angesteuerten Maschinenmodule der Beispielmaschine aus Bild 7-5.

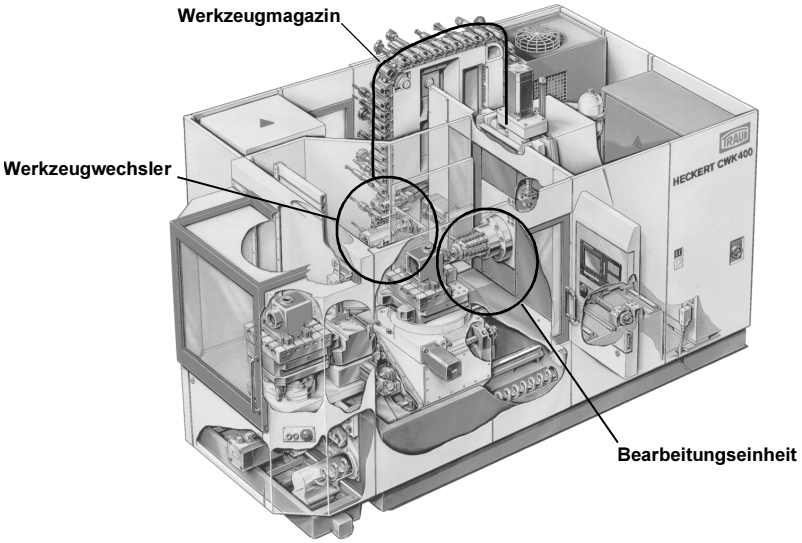

Bild 7-5: Beispielmaschine Heckert CWK 400.

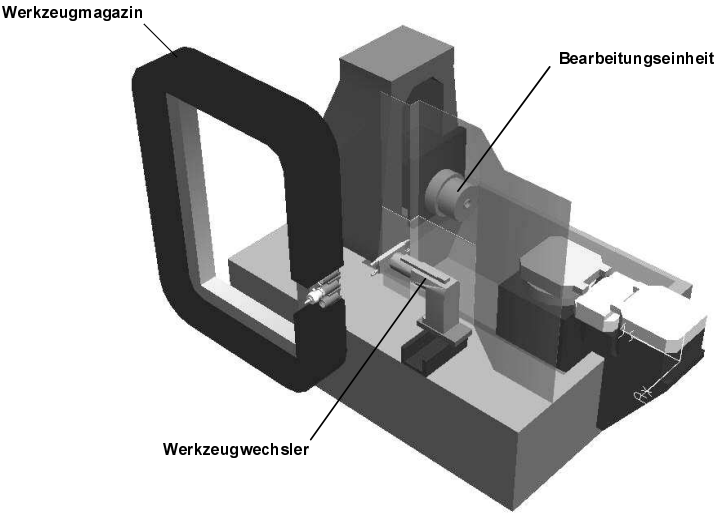

Bild 7-6: Die Beispielmaschine im Simulationssystem.

In der Simulation werden die Bewegungen der mechanischen Komponenten von den Ausgangssignalen der Steuerung angestoßen und laufen dann zeitgesteuert ab. Beim Erreichen der jeweiligen Endlage wird dann ein entsprechendes Eingangssignal in die verteilte Steuerung eingespeist. Zusätzlich kann durch einen weiteren, an die Ein- und Ausgänge der Steuerungsmodule angeschlossenen Rechner mit einer einfachen Signalerfassungs- und –simulationssoftware das Verhalten des verteilten, kooperativen Steuerungssystems präzise analysiert werden.

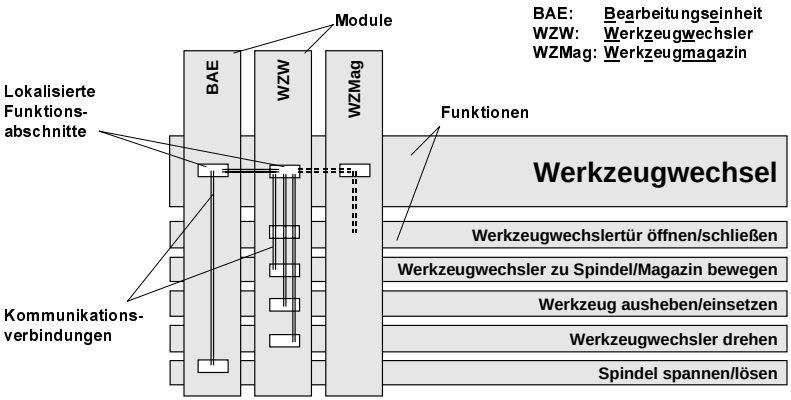

Bild 7-7: Zuordnung der Teilfunktionen zu den Modulen.

An der Werkzeugwechselfunktion sind der Werkzeugwechsler (WZW), die NC-Bearbeitungseinheit (BAE) und das Werkzeugmagazin (WZMag) beteiligt. Ausgehend von einer idealisierten Modularisierung einer Werkzeugmaschine (s. auch Abschnitt 7.3.2), enthält jede Funktionseinheit eine eigene Steuerung. In Bild 7-7 ist eine mögliche Zuordnung einiger dabei benötigter Teilfunktionen dargestellt.

In Bild 7-8 ist ein Auszug dieser Koordinationsfunktion dargestellt. Die dargestellten Client-Stellen fungieren dabei als Funktionsaufrufe für die entsprechenden, zu koordinierenden Einzelfunktionen, die beim Werkzeugwechsel benötigt werden.

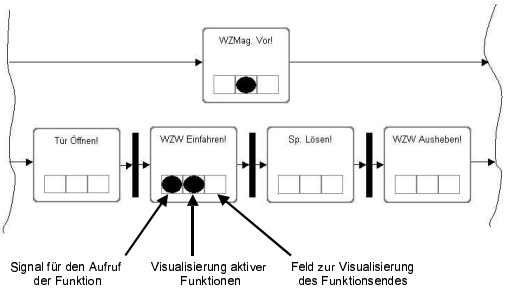

Bild 7-8: Ausschnitt aus der Koordinationsfunktion für den Werkzeugwechsel.

Bild 7-9 zeigt die der Koordinationsfunktion untergeordnete Einzelfunktion „Türsteuerung“. Es werden vier Zustände der Tür abhängig von Anweisungs- und Sensorsignalen durchlaufen. Dabei wird die Aktion zum Öffnen der Tür gestartet, wenn zum einen die Einzelfunktion sich im Zustand „Tür ist zu“ befindet und zum anderen „Öffnen Anweisung“ markiert wird.

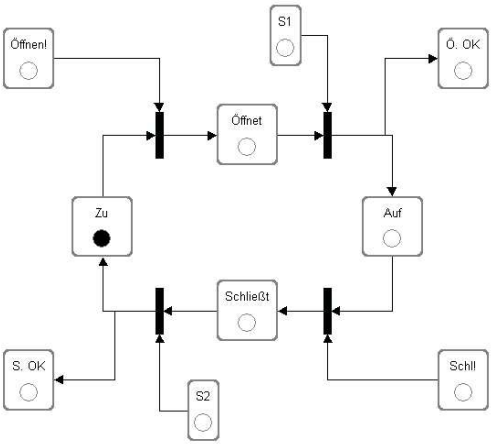

Bild 7-9: Einzelfunktion Türsteuerung.

7.2.2 Schritte des Vorgehens

Bei der Umsetzung der Werkzeugwechselfunktion an der simulierten Maschine wurde entsprechend des in Abschnitt 5.2 vorgeschlagenen Vorgehens die Funktionalität vollständig unabhängig von der Topologie des Steuerungssystems entwickelt. Dabei wurde die in Bild 7-8 als Ausschnitt dargestellte Koordinationsfunktion und eine Reihe von Einzelfunktionen, die meist ähnlich der in Bild 7-9 dargestellten Funktion sind, entworfen. Parallel dazu wurde die aus Bild 7-2 bekannte Topologie festgelegt. Die Verteilung der entstandenen funktionalen Module auf die Steuerungsmodule der topologischen Sicht wurde im Anschluss daran durchgeführt.

Da im Rahmen dieser Arbeit kein Entwicklungswerkzeug realisiert werden sollte (Abschnitt 2.6), mussten diese Schritte zunächst mit Bleistift und Papier durchgeführt werden. Auch die Planung der Verteilung geschah dabei von Hand, wobei aber nach den Regeln des vorgeschlagenen, regelbasierten Verteilungsverfahrens vorgegangen wurde. Es ergab sich dabei stets eine sinnvolle Verteilung. Anschließend mussten die dabei entstandenen, grafischen Entwürfe von Hand in die entsprechenden, von der Systemsoftware lesbaren Parametersätze transformiert werden, die dann durch die Softwaremanagementfunktionen auf die Steuerungsmodule geladen und gestartet werden konnten.

Bei diesen Arbeiten konnten das Vorgehen, die vorgeschlagenen Modellierungssichten sowie das Verfahren zur Planung der Softwareverteilung verifiziert werden. Durch eine Entwicklungsumgebung, die die Modellierung in der funktionalen, der topologischen sowie der distributiven Sicht unterstützt, das regelbasierte Verteilungsverfahren sowie entsprechende Schätzfunktionen anbietet und die Softwaremanagement- und Diagnosefunktionen integriert, kann auf Basis der entwickelten Konzepte ein effizienter Umgang mit verteilten, kooperativen Steuerungen erreicht werden.

7.2.3 Einsatz an der simulierten Werkzeugmaschine

Der Einsatz eines verteilten, kooperativen Steuerungssystems an einer Werkzeugmaschine soll im Folgenden anhand eines Beispielszenarios dargestellt werden. Dazu wird von dem in Abschnitt 7.2.1 vorgestellten Steuerungssystem ausgegangen. Soweit für das Einsatzbeispiel benötigt wird die Funktionalität des Entwicklungswerkzeugs durch einfache, zu diesem Zweck entwickelte ActiveX-Kontrollelemente realisiert, die in eine Power-Point-Anwendung eingebunden sind. Die Steuerungsfunktionen entsprechen der in Abschnitt 7.2.1 eingeführten Koordinationsfunktion sowie Einzelfunktionen entsprechend der Türsteuerungen.

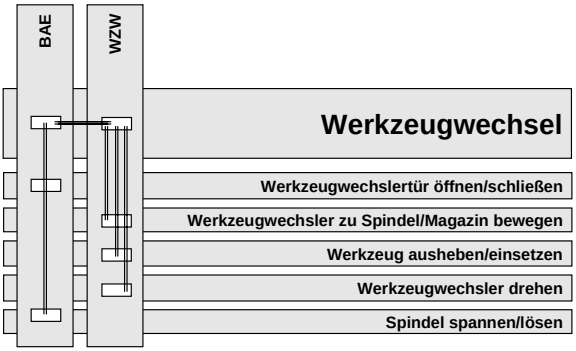

Bild 7-10: Erste Konfiguration des Beispielszenarios.

In einem ersten Schritt des Beispielszenarios (Bild 7-10) soll davon ausgegangen werden, dass das Bearbeitungszentrum beim Hersteller noch ohne Werkzeugmagazin in Betrieb genommen werden soll. Dazu wird die Steuerungssoftware der Bearbeitungseinheit und des Werkzeugwechslers in die entsprechenden Modulsteuerungen geladen. Die Abschnitte der Koordinationsfunktion, die später durch das Werkzeugmagazin auszuführen sind, werden dabei nicht berücksichtigt. Dazu werden die Kommunikationsübergänge zur Modulsteuerung durch eine entsprechende Parametrierung kurz geschlossen.

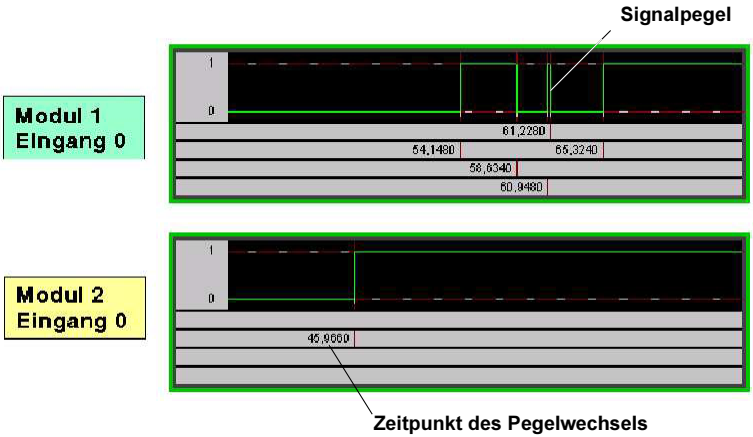

Bild 7-11: Visualisierung der Diagnose von zwei auf unterschiedlichen Modulen liegenden Eingängen.

Anhand dieser Steuerungsfunktionen kann die implementierte Diagnosefunktionalität demonstriert werden. Dazu werden durch entsprechende ActiveX-Kontrollelemente auf der Power-Point-Präsentationsoberfläche beispielsweise Ein- und Ausgänge visualisiert und Signalverläufe von Ein- und Ausgängen unterschiedlicher Steuerungsmodule in zeitlichem Bezug zueinander durch eine Oszilloskopfunktion dargestellt (Bild 7-11). Diese Diagnosefunktionen können auch auf die Markierung der Stellen angewandt werden.

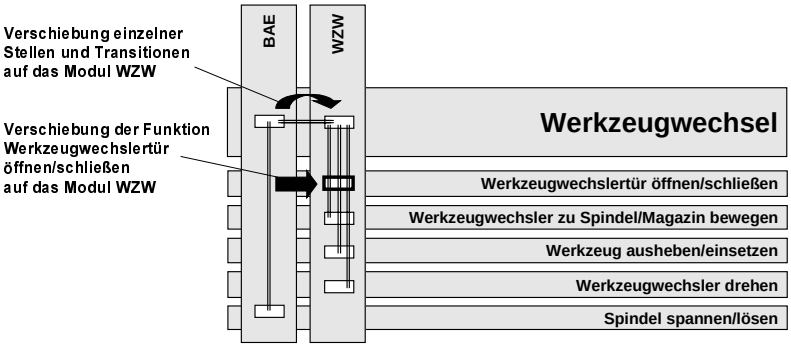

Bild 7-12: Zweite Konfiguration des Beispielszenarios.

In einem zweiten Schritt des Beispielszenarios wird eine Änderung der Softwareverteilung durchgeführt (Bild 7-12). Dazu soll angenommen werden, dass zur Erzielung einer einfacheren und besser zugänglichen Installation der Sensoren und Aktoren der Werkzeugwechslertür deren Verkabelung nicht mehr zur Steuerung der Bearbeitungseinheit, sondern zur sich in unmittelbarer Nähe befindenden Steuerung des Werkzeugwechslers führen soll. Durch diese Änderung in der Steuerungstopologie wird die Verlegung der Einzelfunktion „Türsteuerung“ auf die Modulsteuerung des Werkzeugwechslers notwendig. Zur Vermeidung eines erhöhten Kommunikationsaufkommens ist eine Anpassung der Verteilung der Koordinationsfunktion sinnvoll. Entsprechend werden die betroffenen, bereits gestarteten Teilfunktionen angehalten und durch in ihrer Verteilung angepasste Teilfunktionen ersetzt sowie die Verkabelung der Sensoren und Aktoren der Werkzeugwechslertür verändert. Die Steuerungsmodule können anschließend die Abarbeitung der Werkzeugwechslerfunktion wieder aufnehmen.

In einem dritten Schritt wird die Integration des bereits vorgetesteten Werkzeugmagazins in das Steuerungssystem gezeigt (Bild 7-13). Dies könnte bereits beim Kunden im Rahmen der dortigen Inbetriebnahme geschehen, was insbesondere

dann sinnvoll erscheint, wenn ein solches Werkzeugmagazin von einem anderen Standort als die Bearbeitungseinheit bezogen wird. Das Werkzeugmagazin enthält dabei auf seiner Modulsteuerung bereits die Einzelfunktionen zur Ansteuerung seiner Komponenten. Zur Integration muss nur noch der entsprechende Anteil der Koordinationsfunktion geladen und entsprechend verknüpft werden. Damit kann ein solches mechatronisches, bereits vorgetestetes Maschinenmodul schnell und einfach in eine Maschine integriert werden.

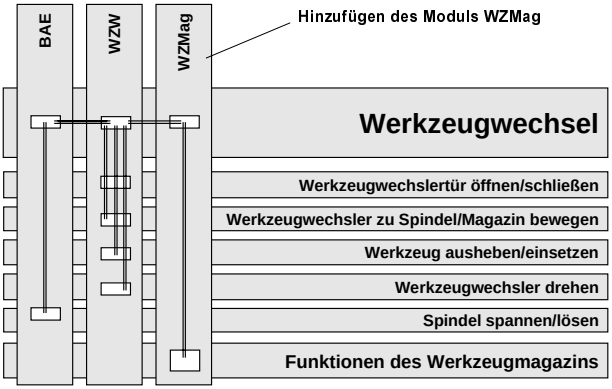

Bild 7-13: Dritte Konfiguration des Beispielszenarios.

7.3 Bewertung

7.3.1 Erreichte Verbesserung der Leistungsfähigkeit der realen Werkzeugmaschine

Um die Leistungsfähigkeit einer verteilten, kooperativen Steuerung nach den entwickelten Konzepten beurteilen zu können, wurden mit Hilfe eines weiteren Rechners und einer einfachen Signalerfassungs- und –simulationssoftware Reaktionszeitmessungen vorgenommen. Zunächst wurden an dem aus Abschnitt 7.2.1 bekannten und mit einer heute üblichen, zentralen Steuerung mit dezentraler Ein- und Ausgabe ausgestatteten Bearbeitungszentrum Referenzwerte ermittelt. Dabei konnte zwischen einem Sensorsignal und dem unmittelbar dadurch ausgelösten Aktorsignal ein Zeitverzug von 40 ms festgestellt werden. Dieser Zeitverzug

schwankte dabei zwischen 25 ms und 55 ms. Zwischen den einzelnen Sensor/Aktor-Kombinationen wurden nur geringfügige Unterschiede festgestellt.

Eine Analyse der Werkzeugwechselfunktion ergab, dass innerhalb derjenigen Zeit, die additiv in die Span-zu-Span-Zeit und damit direkt in die Bearbeitungszeit eingeht, 10 derartige Sensor-/Aktor-Signalketten enthalten sind. Dementsprechend ergibt sich eine steuerungsbedingte Verzögerungszeit von 400 ms.

An dem aus Abschnitt 7.1 bekannten Steuerungssystem wurde dagegen für eine lokale Reaktion 5 ms gemessen, für eine Reaktion, die mit einer Markenwanderung über die Steuerungsmodulgrenzen hinweg verbunden ist, beträgt diese Zeit 40 ms. Durch die gewählte Verteilung waren 4 Markenwanderungen notwendig. Damit ergibt sich eine steuerungsbedingte Verzögerungszeit von 150 ms. Anhand dieser Gegenüberstellung wird deutlich, welche Leistungspotenziale durch verteilte, kooperative Steuerungen nutzbar gemacht werden können.

Der Umstand, dass hier nicht der gesamte Steuerungsumfang modelliert wurde, hat auf die Aussagekraft dieser Ergebnisse kaum Einfluss. Einerseits bewirkt die grundsätzliche Eigenschaft ereignisorientierter Konzepte, dass Funktionen, die keine Ereignisse zu verarbeiten haben, weder das Kommunikationssystem, noch die Prozessoren belasten. Bei der Beispielmaschine sind während des Werkzeugwechsels keine anderen ereignisintensiven Funktionen aktiv, weshalb der Verzicht auf deren Abbildung sich nicht nennenswert auswirkt. Andererseits kämen bei einer vollständigen Abbildung des Steuerungsumfangs auch weitere Steuerungsmodule und damit Prozessorleistung hinzu. Somit bliebe für die am Werkzeugwechsel beteiligten Steuerungsmodule der Funktionsumfang weitgehend konstant

7.3.2 Kostenabschätzung

Zur Abschätzung der Kosten, die sich beim Einsatz dezentralisierter Steuerungen ergeben, wurde für die Beispielmaschine eine Aufteilung in mechanische Module vorgenommen. Dabei wurde bewusst von einer starken Modularisierung ausgegangen. Damit ergab sich die in Bild 7-14 dargestellte mechanische Modularisierung. Wird jedem dieser mechanischen Module eine eigene Steuerung zugeordnet, dann kommen zur ehemaligen Zentralsteuerung (NC mit SPS) und der weiterhin kontaktbehafteten Sicherheitstechnik mit Lastspannungsaufschaltung (bei späterer Einbeziehung der Sicherheitstechnik als Modul "Kapselung") weitere sechs Modulsteuerungen hinzu.

Würden hierfür beispielsweise Steuerungen vom Typ Siemens ET200X BM147/CPU (siehe Abschnitt 7.1) verwendet, dann verteuert sich das Steue-

rungssystem insgesamt maximal um die Preisdifferenz zwischen den Geräten ET200X mit und ohne integrierter SPS (ca. 280 DM), da zum Vergleich eine entsprechende, dezentrale Installation herangezogen wurde. Damit beziffern sich die zusätzlichen Kosten zunächst auf 6 x 280 DM = 1680 DM. Aufgrund des Kostendrucks im Werkzeugmaschinenbau sprechen diese zusätzlichen Kosten vorerst gegen eine Dezentralisierung der Steuerungstechnik. Die genannte Preisdifferenz von 1680 DM bezieht sich allerdings auf aktuelle Preise und auf die Voraussetzung, dass die Hardware der bisherigen Zentralsteuerung unverändert beibehalten wird. Bei einer zunehmenden Marktdurchdringung entsprechender Module und den damit verbundenen Mengeneffekten und einem weiter anhaltenden Preisverfall in der Mikroelektronik werden sich diese Zusatzkosten weiter verringern.

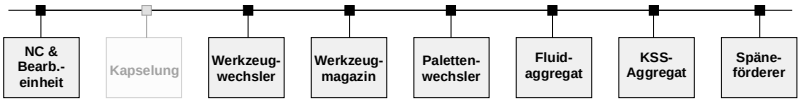

Bild 7-14: Idealisierte Modularisierung der Heckert CWK 400.

Wird dagegen konsequent auf verteilte, kooperative Steuerungen gesetzt und auf die Hardware einer Zentralsteuerung gänzlich verzichtet, so können sich bereits mit heutigen Hardwarepreisen Kosteneinsparungen ergeben. Weitere Kosteneinsparungen wären durch Steuerungs- und Kommunikationssysteme zu erwarten, die es erlauben, beispielsweise die NC-Funktionalität der Maschine auf integrierte Motor/Antriebsreglermodule zu verteilen. In diesem Fall könnte man den Schaltschrank und damit verbundene Materialkosten sowie Projektierungs- und Dokumentationsaufwendungen teilweise einsparen.

Darüber hinaus muss auch der Mehrwert der Maschine - beispielsweise durch verbesserte Reaktionszeiten und damit einen schnelleren Werkzeugwechsel bei Bearbeitungszentren - mit berücksichtigt werden. Bezogen auf die Beispielmaschine aus Abschnitt 7.2 ergibt sich unter der Annahme einer Taktzeit von 10 min und 10 Werkzeugwechselvorgängen pro Werkstück eine Einsparung von 10 * 0,25 s = 2,5 s, die direkt in die Span-zu-Span-Zeit und damit in die Taktzeit eingeht (siehe Abschnitt 7.3.1). Dies entspricht einer Erhöhung der Maschinenleistung von ca. 0,4%, rechtfertigt also bei einer Investitionshöhe von 500.000 DM Mehrkosten in Höhe von 2.000 DM.

Neben diesen, direkt oder indirekt mit der Funktion und der Ausrüstung der Maschine zusammenhängenden Kosteneffekten dezentralisierter Steuerungstechnik soll hier aber auch auf die Vorteile sowohl im Entwicklungsprozess (Parallelisie-

rung, Wiederverwendbarkeit etc.) als auch in der Herstellung (Outsourcing, Vortestbarkeit etc.) hingewiesen werden. Diese Vorteile sind zwar stark einzelfallabhängig und zudem monetär schwer bewertbar, können aber einen entscheidenden Einfluss auf die Gesamtkostensituation haben.

7.4 Zusammenfassung

Mit der prototypischen Realisierung eines verteilten, kooperativen Steuerungssystems konnte die Realisierbarkeit der konzipierten Systemsoftware nachgewiesen und die vorgeschlagene Entwicklungsmethode verifiziert werden. Anhand eines Beispielszenarios konnte dann die Anwendung der entwickelten Konzepte hinsichtlich des Softwaremanagements und der Diagnosefunktionalität demonstriert werden. Dabei konnten auf experimentellem Wege auch Daten über das Reaktionszeitverhalten des verteilten, kooperativen Steuerungssystems gewonnen werden. Im Vergleich mit der bestehenden Maschinensteuerung ergab sich dabei eine deutliche Verbesserung der Reaktionszeiten und damit der Leistungsfähigkeit der Werkzeugmaschine durch die Reduktion der steuerungsbedingten Verzögerungen beim Werkzeugwechsel.

Die Abschätzung der wirtschaftlichen Auswirkungen einer verteilten, kooperativen Steuerungstechnik ergab bei aktuellen Preisen einen moderaten Kostenzuwachs, wobei noch keine Kosteneinsparungen durch den Wegfall zentraler Hardwarekomponenten berücksichtigt wurden. Wird auf solche Komponenten verzichtet, so ist sogar eine Hardwarekosteneinsparung durch den Übergang auf eine verteilte, kooperative Steuerungstechnik erreichbar. Weitere Potenziale können sich durch die Ausdehnung des verteilten, kooperativen Konzepts auf die NC- und die Sicherheitstechnik ergeben. Auch ist für die Zukunft aufgrund von Mengeneffekten in der Herstellung geeigneter Steuerungsmodule und durch den voranschreitenden Preisverfall in der Mikroelektronik mit weiteren Kosteneinsparungen zu rechnen. Darüber hinaus ergeben sich durch die mechatronische Modularisierung und den optimierten Entwicklungsprozess noch weitere, allerdings nur schwer monetär bewertbare Vorteile.

Damit konnte beispielhaft gezeigt werden, dass durch die entwickelten Konzepte die analysierten Potenziale der verteilten, kooperativen Steuerungstechnik unter gleichzeitiger Berücksichtigung der gestellten Anforderungen nutzbar gemacht werden können.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Als Ziel dieser Arbeit wurde in Abschnitt 1.2 definiert, die Potenziale verteilter, kooperierender Steuerungskonzepte durch eine auf die Anforderungen in der Entwicklung maschinennaher Steuerungssoftware angepasste Entwicklungsmethodik und einer präzise darauf abgestimmten Steuerungstechnik für den Werkzeugmaschinenbau nutzbar zu machen.

Dazu wurde zunächst in Kapitel 2 der Stand der Technik hinsichtlich des Einsatzes verteilter, kooperativer Steuerungstechnik im Werkzeugmaschinenbau eingehend und im Vergleich zu anderen Einsatzgebieten diskutiert und daraus der dieser Arbeit zugrundeliegende Forschungsbedarf präzisiert. Der besonders hervorzuhebender Kernpunkt ist es dabei, Konzepte zu entwickeln, mit denen verteilte, kooperative Steuerungen ohne eine aufwendige, fehlerträchtige und unflexible, manuelle Programmierung der Kommunikation beziehungsweise Festlegung von Kommunikationsschnittstellen eingesetzt werden können. Es soll möglich sein, die eigentliche Steuerungssoftware ohne Veränderungen auf unterschiedliche Steuerungstopologien zu verteilen. Dabei soll sich das verteilte, kooperative Steuerungssystem dem Steuerungsentwickler aus funktionaler Sicht gewissermaßen wie bisher als monolithische Steuerung präsentieren.

In Kapitel 3 wurden dann die Potenziale einer verteilten, kooperativen Steuerungstechnik sowie die durch den Werkzeugmaschinenbau gestellten Anforderungen detaillierter analysiert. Die Potenziale wurden einerseits durch Betrachtung von verteilten, kooperativen Steuerungskonzepten in anderen Bereichen abgeleitet. Andererseits wurden durch gezielte Befragungen im Rahmen eines vom Verein Deutscher Werkzeugmaschinenfabriken geförderten Forschungsprojekts bei Vertretern der Werkzeugmaschinenindustrie die Anforderungen herausgearbeitet.

Zur Erarbeitung eines Konzepts für die verteilte, kooperative Steuerungstechnik wurden dann in Kapitel 4 gezielt weitere, modellierungstechnische Grundlagen erarbeitet. Dabei wurden die im Maschinenbau gebräuchlichen Modellierungstechniken vorgestellt und ihre jeweiligen Einsatzmöglichkeiten kurz diskutiert. Zudem wurden die aktuellsten Modellierungstechniken aus dem Bereich der objektorientierten Modellierung vorgestellt. Abschließend wurde dabei der Einsatz colorierter Petrinetze als Basismodellierungstechnik zur Modellierung von Abläufen festgelegt.

Auf dieser Grundlage konnte dann ein Konzept für die Entwicklungsmethode und die Systemsoftware (Kapitel 5) einer verteilten, kooperativen Steuerungstechnik entwickelt werden. Für die Entwicklungsmethode wird ein Vorgehen in drei Schritten zur Modellierung der Funktionalität, der Steuerungstopologie und der Verteilung vorgeschlagen. Dies soll durch eine aus der funktionalen, der topologischen und der distributiven Sicht bestehenden Modellierungstechnik sowie durch ein regelbasiertes Verfahren zur Planung der Softwareverteilung unterstützt werden. Darüber hinaus wurden Funktionalitäten der Abarbeitung der Steuerungssoftware, der transparenten Kommunikation, des Softwaremanagements und der Diagnose grob konzipiert. Daran anschließend wurde auf Grundlage einiger softwaretechnischer Basiskonzepte ein Designkonzept der Systemsoftware vorgestellt (Kapitel 6). Ebenfalls wird dort die Abbildung der geforderten Funktionalitäten auf die Schichten des Designkonzepts detailliert besprochen.

In einem Anwendungsbeispiel (Kapitel 7) auf Basis einer prototypischen Realisierung eines verteilten, kooperativen Steuerungssystems konnte die Realisierbarkeit der konzipierten Systemsoftware nachgewiesen und die vorgeschlagene Entwicklungsmethode anhand eines Beispiels verifiziert werden. Es konnten durch Vergleichsmessungen von Reaktionszeiten an der Beispielmaschine und an einem verteilten, kooperativen Steuerungssystem die analysierten Leistungspotenziale durch eine verkürzte Span-zu-Span-Zeit beim Werkzeugwechsel beispielhaft nachgewiesen werden. Dabei wiegt im Beispiel der aus der Leistungssteigerung herrührende Mehrwert der Maschine die heute noch bestehenden Mehrkosten einer verteilten, kooperativen Steuerungslösung bereits auf. Durch eine entsprechende Weiterentwicklung verteilter, kooperativer Konzepte und deren Ausdehnung auf die NC- und die Sicherheitstechnik können zukünftig weitere Einspareffekte realisierbar werden.

Damit ist die Zielsetzung der Arbeit, die Potenziale verteilter, kooperativer Steuerungstechnik für den Werkzeugmaschinenbau nutzbar zu machen, erreicht.

8.2 Ausblick

Um die Ergebnisse dieser Arbeit in der Praxis nutzen zu können, ist die kommerzielle Verfügbarkeit einer Entwicklungsumgebung sowie von Steuerungsmodulen mit einer Systemsoftware entsprechend der hier vorgeschlagenen Konzepte erforderlich. Die Entwicklungsumgebung könnte dabei auch an der Entwicklungsoberfläche die gebräuchlichen, grafischen Modellierungssprachen anbieten, die auf die vorgeschlagene, gemeinsame Basismodellierungstechnik aufsetzen.

Darüber hinaus sollte die Vision einer vollständig nach verteilten, kooperativen Konzepten aufgebauten Steuerungstechnik entsprechend Bild 8-1 in der Forschung weiterverfolgt werden, da sich die Potenziale dieser Technik nicht auf die Steuerung von Abläufen beschränken. Zu deren Verwirklichung wären die drei folgenden Themenkomplexe zu bearbeiten:

- Bisherige, zentrale NC-Steuerungen müssten in ähnlicher Weise auf integrierte, an den Motoren angebrachte Antriebsmodule zerfallen. Die Geometrieverarbeitung erfolgt dann ebenfalls durch Kooperation dieser Achsmodule über das Bussystem. Wechselnde Achsverbünde – wie z. B. in Mehrspindel-Drehmaschinen – könnten so ebenfalls einfacher realisiert werden.
- Sicherheitstechnische Funktionen müssten in einem derartigen Steuerungssystem durch einen gezielt redundanten Einsatz von Steuerungsmodulen verwirklicht werden, so dass der Einsatz in sich redundant aufgebauter und entsprechend teuere Sicherheitssteuerungen vermieden werden kann.
- Die oben aufgeführte Entwicklungsumgebung für verteilte, kooperative Steuerungen wäre um integrierte Entwicklungsfunktionen zur Modellierung von NC- und sicherheitstechnischen Steuerungsaufgaben zu erweitern, um den Steuerungsentwickler einen nahtlosen Übergang zwischen diesen Bereichen der Steuerungstechnik zu ermöglichen.

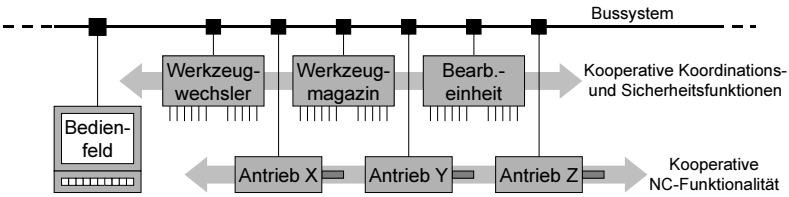

Bild 8-1: Vision einer vollständig dezentralisierten Steuerungstechnik im Werkzeugmaschinenbau am Beispiel eines Fräsbearbeitungszentrums.

9 Formelverzeichnis

10 Abkürzungsverzeichnis

ACL	Application Cooperation Layer
AL	Application Layer
AS	Ablaufsprache
AS-I	Aktuator-/Sensor-Interface
AWL	Anweisungsliste
B/E-Netz	Bedingungs-/Ereignis-Netz
BAE	Bearbeitungseinheit
CAN	Controller Area Network
CASE	Computer Aided Software Engineering
CDL	Communication Driver Layer
CPN	Coloriertes Petrinetz
DeKOS	Dezentrale, kooperierende, offene Systeme
E-/A-Modul	Eingangs-/Ausgangsmodul
FBS	Funktionsbausteinsprache
HÜMNOS	Herstellerübergreifende Module zur Nutzung der offenen Steuerungsachitektur
IODL	Input Output Driver Layer
KOP	Kontaktplan
LON	Local Operating Network
MML	Message Management Layer
MRP	Mehrrechner Pearl
NC	Numerical Control
OSACA	Open Systems Architecture for Controls within Automation Systems
PAP	Programmablaufplan

Profibus	Process Field Bus
ROOM	Real Time Object Oriented Modeling
RTA	Real Time Operating System Adaptor
S/T-Netz	Stellen-/Transitionen-Netz
SPS	Speicherprogrammierbare Steuerung
ST	Strukturierter Text
TAL	Tool Access Layer
TCP/IP	Transmission Control Protocol / Internet Protocol
TEP	Transition Expression Place
UML	Unified Modeling Language
UML-RT	UML-Erweiterung für reaktive Systeme (Real Time)
WZMag	Werkzeugmagazin
WZW	Werkzeugwechsler

11 Literaturverzeichnis

Abel 1990
Abel, D.: Petri-Netze für Ingenieure – Modellbildung und Analyse diskret gesteuerter Systeme. Berlin: Springer 1990.

Anker & Wirth 1994
Anker, A.; Wirth, M.: Produktion im Wandel: Automatisierung tut not. VDI-Z 136 (1994) 4, S. 28 – 29.

Aplin 1997
Aplin, J. D.: Primary Flight Computers for the Boeing 777. Microprocessors and Microsystems 20 (1997) 8, S. 473 – 478.

Awad u. a. 1996
Awad, M.; Kuusela, J.; Ziegler, J.: Object Oriented Technology for Real Time Systems – A Practical Approach Using OMT and Fusion. Upper Saddle River: Prentice Hall 1996.

Baginski & Müller 1998
Baginski, A.; Müller, M.: Interbus. Grundlagen & Praxis. Heidelberg: Hüthig 1998.

Balzert 1996
Balzert, H.: Lehrbuch der Softwaretechnik – Software-Entwicklung. Heidelberg: Spektrum 1996.

Bender 1990
Bender, K. (Hrsg.): PROFIBUS. München, Wien: Hanser Verlag 1990.

Bender 1999
Bender, K. (Hrsg.): Echtzeitsimulation zum Test von Maschinensteuerungen. München: Utz Verlag Wissenschaft 1999.

Bertram u. a. 1997
Bertram, T.; Schroeder, W.; Dominke, P.; Volkart, A.: CARTRONIC -Ein Ordnungskonzept für die Steuerungs- und Regelungssysteme in Kraftfahrzeugen. In: Systemengineering in der KFZ-Entwicklung, VDI-Bericht Nr. 1374. Düsseldorf: VDI-Verlag 1997.

Bitzenberger & Schmidt 1997
Bitzenberger, T.; Schmidt, S.: Antriebsmanagement am Beispiel eines Fahrzeugs mit CVT-Getriebe. In: Steuerungssysteme für den Antriebsstrang, Symposium, Tagungsband, Berlin, 18. – 19. September 1997, S. 1 – 13.

Booch 1994

Booch, G.: Object-Oriented Analysis and Design with Applications. Redwood City: Benjamin/Cummings 1994.

Booch u. a. 1998

Booch, G.; Rumbaugh, J.; Jacobson, I.: Unified Modeling Language User Guide. Reading: Addison Wesley 1998.

Brandl u. a. 1996

Brandl, T.; Lutz, R.; Reichenbächer, J.: ASPECT - a CASE Tool for Control Functions Originating from Mechanical Layout. In: Storr, A.; Jarvis, D. (Hrsg.): Software Engineering for Manufacturing Systems. London: Chapman & Hall, 1996, S. 95 - 106.

Bregulla u. a. 2000

Bregulla, M.; Bender, K.; Birkhofer, R.: Dezentrale Kooperierende Offene Systeme – Projektbericht. Magdeburg: Tagungsband Verteilte Automatisierung 2000, S. 325 - 332.

Broy u. a. 1999

Broy, M.; Huber, F.; Schätz, B.: AutoFocus - Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme. Informatik Forschung und Entwicklung 14 (1999) 3, S. 121 - 134.

Budde & Nieters 1992

Budde, R.; Nieters, H.: Einführung in die Netztheorie (Theorie der Petri-Netze). In: Schnieder, E: Petrinetze in der Automatisierungstechnik. München: Oldenbourg 1992.

Coad & Yourdon 1991a

Coad, P.; Yourdon, E.: Object-Oriented Analysis. Englewood Cliffs: Yourdon Press 1991.

Coad & Yourdon 1991b

Coad, P.; Yourdon, E.: Object-Oriented Design. Englewood Cliffs: Yourdon Press 1991

Colombo 1998

Colombo, A. W.: Development and Implementation of Hierarchical Control Structuresof Flexible Production Systems Using High Level Petri Net. Bamberg: Meisenbach 1998.

Daniel 1996

Daniel, C.: Dynamisches Konfigurieren von Steuerungssoftware für offene Systeme. Berlin: Springer 1996.

Dietrich u. a. 1999

Dietrich, D.; Loy, D.; Schweintzer, H.-J.: LON-Technologie. Verteilte Systeme in der Anwendung. Heidelberg: Decker-Müller 1999

DIN 19226-5 1994
DIN 19226 Teil 5: Leittechnik, Regelungstechnik und Steuerungstechnik. Funktionelle Begriffe. Berlin: Beuth 1994.

DIN 66001 1983
DIN 66001: Sinnbilder und ihre Anwendung. Berlin: Beuth 1983.

DIN 66253 1988
DIN 66253 Teil 2 und 3: Programmiersprache PEARL, Mehrrechner PEARL. Berlin: Beuth 1988.

DIN 66261 1985
DIN 66261: Sinnbilder für Struktogramme nach Nassi-Shneiderman. Berlin: Beuth 1985.

DIN EN 28631 1994
DIN EN 28631: Programmkonstrukte und Regeln für ihre Anwendung. Berlin: Beuth 1994.

DIN EN 50170-2 1996
DIN EN 50170 Teil 2: Universelles Feldbuskommunikationssystem, Teil 2: PROFIBUS. Berlin: Beuth 1996.

DIN EN 50254 1998
DIN EN 50254: Kommunikationssubsystem mit hoher Effizienz für kleine Datenpakete. Berlin: Beuth 1998.

DIN IEC 1131-3 1992
DIN IEC 1131 Teil 3: Speicherprogrammierbare Steuerungen - Programmiersprachen. Berlin: Beuth 1992.

Douglass 1998
Douglass, B. P.: Real-Time UML – Developing Efficient Objects for Embedded Systems. Reading: Addison Wesley 1998.

Eckrich 1997
Eckrich, M.: Semiformale Entwurfsmethoden bei BMW. Informationstechnik und technische Informatik 39 (1997) 3, S. 29 – 33.

Etschberger 1994
Etschberger, K.: CAN - Controller Area Network, Grundlagen, Protokolle, Bausteine, Anwendungen. München; Wien: Hanser 1994.

Färber 1994
Färber, G.: Feldbus-Technik heute und morgen. Automatisierungstechnische Praxis 36 (1994) 11, S. 16 – 36.

Feldmann 1997

Feldmann, C.: Eine Methode für die integrierte rechnergestützte Montageplanung. Berlin: Springer 1997 (iwb Forschungsberichte 104).

Ferscha 1992

Ferscha, A.: A Petri Net Approach for Performance Oriented Parallel Program Design. Journal of Parallel and Distributed Computing 15 (1992), S. 188 - 206.

Fischer 1993

Fischer, M.: Vor-Entwicklung einer Flugsteuerung für ein kleines Verkehrsflugzeug. Zeitschrift für Flugwissenschaften und Weltraumforschung 17 (1993) 1, S. 57 – 64.

Fleckenstein 1987

Fleckenstein, J.: Zustandsgrafen für SPS – Graphikunterstützte Programmierung und steuerungsunabhängige Darstellung. Berlin: Springer 1987.

Furrer 1998

Furrer, F. J.: Ethernet TCP-IP für die Industrieautomation. Grundlagen und Praxis. Heidelberg: Hüthig 1998.

Gausemeier u. a. 1998

Gausemeier, J.; Förste, S.; Gerdes, K. H.: Dezentrale Steuerungsarchitektur für modulare Materialflusssysteme. Nürnberg: Tagungsband SPS/IPC/DRIVES 1998, S. 602 – 611.

Gigou 1998

Gigou, D.: CAN nicht nur fürs Auto. 16-Bit-Mikrocontroller mit CAN 2.0B und OTP. Components (Deutsche Ausgabe) 36 (1998) 2, S. 28 - 29.

Glas 1993

Glas, J.: Standardisierter Aufbau anwendungsspezifischer Zellenrechnersoftware. Berlin: Springer 1993. (iwb Forschungsberichte 61).

Glüer & Schmidt 1988

Glüer, D.; Schmidt, G.: Die Anwendung von Petrinetzen zu Modellbildung, Simulation und Steuerungsentwurf bei Flexiblen Fertigungssystemen. at Automatisierungstechnik 36 (1988) 12, S. 436 - 471.

Groha 1988

Groha, A.: Universelles Zellenrechnerkonzept für flexible Fertigungssysteme. Berlin: Springer 1988. (iwb Forschungsberichte 14).

Grötsch & Seubert 1997
Grötsch, E.; Seubert, L.: SPS II – Speicherprogrammierbare Steuerungen. Band 2: Programmbeispiele und Produkte. München: Oldenbourg 1997.

Groupe Schneider 1998
Groupe Schneider (Hrsg.): Automatisieren mit Premium. Katalog. Ratingen 1998.

Günzel 1995
Günzel, U.: Menschen und ihre Rechner am Beispiel Airbus. In: Erfolg durch zuverlässige Technik, VDI-Bericht Nr. 1239. Düsseldorf: VDI-Verlag 1995.

Harel 1987
Harel, D.: A Visual Formalism for Complex Systems. In: Science of Computer Programming, Elsevier Science Publishers (North Holland) 1987, S. 231 - 274.

Harel 1990
Harel, D.: STATEMATE: A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering, 16 (1990) 4, S. 403 - 414.

Helmey 1995
Helmey, D.: Understanding GRAFCET. Plant Engineering 49 (1995) 9, S. 78 – 80.

Herkommer 1999
Herkommer, G.: Mehr Wunsch denn Wirklichkeit. Computer & Automation 1 (1999) 5, S. 64 – 67.

Herrscher 1981
Herrscher, A.: Beitrag zum Entwurf und zur Realisierung prozessnaher Steuerungsfunktionen. Berlin: Springer 1981.

Honekamp 1998
Honekamp, U.: IPANEMA – Verteilte Echtzeit - Informationsverarbeitung in mechatronischen Systemen. Düsseldorf: VDI-Verlag 1998.

Hopcroft & Ullman 1979
Hopcroft, J. E.; Ullman, J. D.: Introduction to Automata Theory, Languages and Computation. Reading: Addison Wesley 1979.

HÜMNOS 1998
Autorenkollektiv: Trendwende in der Steuerungstechnik - Herstellerübergreifend offene Systeme. Frankfurt: VDW 1998. (VDW Forschungsberichte).

ISO 7498 1984
ISO 7498: Information Processing Systems – Open Systems Interconnection. Basic Reference Model. Genf: ISO 1984.

Ivantysynova 1996
Ivantysynova, M.: Neue Aktuatortechnologien in Verkehrsflugzeugen. In: 12. Aachener Fluidtechnisches Kolloquium, Band 1. Aachen: Verlag Mainz 1996.

Jackson 1975
Jackson, M. A.: Principles of Program Design. London: Academic Press 1975.

Jakobson u. a. 1992
Jakobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: Object Oriented Software Engineering. London: Academic Press 1975.

Jasperneite 1997
Jasperneite, J.: Interbus-UART als anpassungsfähiges Feldbusinterface. Elektronik Informationen (1997) 4.

Jetter 1999
Jetter, M.; Buchwitz, M.: Das Netz ist die Steuerung. Elektronik 47 (1999) 8, S. 38 - 56.

John & Tiegelkamp 1995
John, K.-H.; Tiegelkamp M.: SPS-Programmierung mit IEC 1131-3. Berlin: Springer 1995.

Jörns u.a. 1995
Jörns, C.; Litz, L.; Bergold, S.: Automatische Erzeugung von SPS-Programmen auf der Basis von Petri-Netzen. atp Automatisierungstechnische Praxis 37 (1995) 3, S. 10 - 14.

Kasturia u. a. 1988
Kasturia, E.; DiCesare, F.; Desrochers, A.: A Real Time Control of Multilevel Manufacturing Systems using Colored Petri Nets. In: Proceedings of the IEEE International Conference on Robotics and Automation 1988, Philadelphia. New York: IEEE 1988, S. 1114 - 1119.

Kernighan & Ritchie 1978
Kernighan, B. W.; Ritchie, D. M.: The C Programming Language. Englewood Cliffs: Prentice Hall 1978.

Kieß 1995
Kieß, J. U.: Objektorientierte Modellierung von Automatisierungssystemen. Software Engineering für Embedded Systems. Berlin: Springer 1995.

Koch 1996

Koch, M. R.: Autonome Fertigungszellen – Gestaltung, Steuerung und integrierte Störungsbehandlung. Berlin: Springer 1996. (iwb Forschungsberichte 98).

Kriesel & Madelung 1999

Kriesel, W. R.; Madelung, O. W.:Das Aktuator-Sensor-Interface für die Automation. München: Hanser 1999.

Lange 1993

Lange, N.: Dezentrale universelle Steuerungsarchitektur für flexible Fertigungssysteme: ein Beitrag zur Reduzierung der Softwareentwicklungskosten in der Fertigungstechnik. Aachen: Shaker 1993. (WZL/IPT Berichte aus der Produktionstechnik 15/93).

Leindl 1997

Leindl, R.: Nicht nur fürs Auto. Design und Elektronik (1997) 4, S. 33 - 34.

Lenschow 1991

Lenschow, R.: Rechnerintegrierte Erstellung und Verifikation von Steuerungsprogrammen als Komponente einer durchgängigen Planungsmethodik. Karlsruhe: Institut für Werkzeugmaschinen und Betriebstechnik, Universität Karlsruhe (TH) 1991.

Luttenberger & Cramer 1992

Luttenberger, N.; Cramer, A.: Messung, Modellierung und Bewertung von Echtzeitsystemen: Methodik und Fallstudie. In: Schnieder, E: Petrinetze in der Automatisierungstechnik. München: Oldenbourg 1992

Meier 1996

Meier, H. S.: Verteilte Intelligenz in zukünftigen Zellensteuerungen. In: Reinhart, G.; Milberg, J.: Dezentrale Steuerungen in Produktionsanlagen. München: Herbert Utz Verlag Wissenschaft 1996, S. 110-124. (iwb Seminarberichte 20).

Meier & Englberger 1999

Meier, H.; Englberger, G.: Verteilte, kooperative Steuerungstechnik - Ansätze und Wechselwirkungen zur Sicherheitstechnik. In: Reinhart, G.; Milberg, J.: Sicherheitstechnik an Werkzeugmaschinen. München: Herbert Utz Verlag Wissenschaft 1999, S. 5-1 - 5-15. (iwb Seminarberichte 48)

Meisel 1996

Meisel, J.: Distributed Control for Hydro-Electric Plants. International Water Power and Dam Construction 48 (1996) 1, S. 22 24.

Milberg 1997

Milberg, J.: Produktion – Eine treibende Kraft für unsere Volkswirtschaft. In: Reinhart, G.; Milberg, J.: Mit Schwung zum Aufschwung. Landsberg: Moderne Industrie 1997.

Möhrle 1989

Möhrle, M.: Petrinetze in der Produktionstechnik - Integration von Planung, Simulation und Steuerung von Produktionsanlagen. Bochum: 1989. (Schriftenreihe des Lehrstuhls für Produktionssysteme und Prozeßleittechnik 89.3).

Moßig & Stäble 1995

Moßig, K.; Stäble, M.: Steuerungssynthese mit kontrollierten Free-Choice Petri-Netzen zur Prozeßbeschreibung. at Automatisierungstechnik 43 (1995) 11, S. 506 - 513.

Nassi & Shneiderman 1973

Nassi, I.; Shneiderman, B.: Flowchart Techniques for Structured Programming. In: SIGPLAN, Aug. 1973, S. 12 - 26.

Oestereich 1997

Oestereich, B.: Objektorientierte Softwareentwicklung mit der Unified Modeling Language. München: Oldenbourg 1997.

OMAC 1994

N. N.: Requirement of Open, Modular Architecture Controllers for Applications in the Automotive Industry, Version 1.1. Detroit: Chrysler, Ford, GM 1994.

Orr 1977

Orr, K. T.: Structured Systems Development. Englewood Cliffs: Yourdon Press 1977.

Osmers 1998

Osmers, U.: Projektieren Speicherprogrammierbarer Steuerungen mit Virtual Reality. Karlsruhe: Institut für Werkzeugmaschinen und Betriebstechnik, Universität Karlsruhe (TH) 1998.

Ottawa Report 1986

N. N.: The Ottawa Report on Reference Models for Manufacturing Standards. Version 1.1. ISO TC 184/SC5/WG1 Document N51, 1986.

Peterson 1981

Peterson, J. L.: Petri Net Theory and the Modeling of Systems. Englewood Cliffs: Prentice-Hall 1981.

Petri 1962
Petri, C. A.: Kommunikation von Automaten, Universität Bonn 1962. (Schriften des rhein.-westfäl. Instituts für instrumentelle Mathematik an der Universität Bonn, Nr. 2).

Pritschow & Sperling 1997
Pritschow, G.; Sperling, W.: Modular System Platform for Open Control Systems. Production Engineering 4 (1997) 2, S. 77 - 80.

Pritschow u. a. 1996
Pritschow, G.; Sperling, W; Müller, J.; Daniel, Ch.: OSACA - Auf dem Weg zur herstellerübergreifend offenen Steuerung. In: Pritschow, G.; Spur, G.; Weck, M.: Komponenten und Schnittstellen für offene Steuerungssysteme. München: Hanser 1996.

Rath 1995
Rath, G.: HiGraph – Grafischer Entwurf, Programmierung und Test von SPS-Software. In: Storr, A.: Software Engineering. Stuttgart 1995.

Rational 1999
RationalSoftware (Hrsg.):Rational Rose Real Time. User's Guide. Cupertino 1999.

Reinhart & Cuiper 1999
Reinhart, G.; Cuiper, R.: Assembly Process and Assembly Control Development – A Holistic and Consistent Approach. Annals of the CIRP 48 (1999) 1, S. 25 – 28.

Reinhart & Meier 2000
Reinhart, G.; Meier, H.: Verteilte, kooperative Steuerungstechnik. Werkstatt und Betrieb 133 (2000) 3, S. 32-34.

Reinhart u. a. 1998
Reinhart, G.; Ansorge, D.; Mauderer, M.; Sabbah, A.: Software-Engineering im Bereich der Produktionstechnik. ZwF 93 (1998) 6, S. 282 – 285.

Reinhart u. a. 1999a
Reinhart, G.; Meier, H.; Sabbah, A.: Störungstolerante Steuerungen. ZwF 94 (1999) 9, S. 486 - 489.

Reinhart u. a. 1999b
Reinhart, G.; Bauer, L.; Meier, H.; Wagner, P.; Weißenberger, M.: Vernetzte Entwicklung komplexer mechatronischer Produkte - Am Beispiel Werkzeugmaschine. ZwF 94 (1999) 4, S. 191-194.

Rumbaugh u. a. 1991
Rumbaugh, J.; Blaha, M.; Premerlani, W; Eddy, F; Lorensen, W: Object-Oriented Modeling and Design. Englewood Cliffs: Prentice-Hall 1991

Sabbah 1995
Sabbah, A.: Steuerung und Störungsbehandlung auf Zellenebene. In: Reinhart, G.; Milberg, J.: Autonome Produktionssysteme. München. Herbert Utz Verlag Wissenschaft 1995, S. 44 - 54. (iwb Seminarberichte 12)

Schäfer 1997
Schäfer, W.: Trendwende in der Steuerungstechnik. Einheitliche Schnittstelle für offene Steuerungen. Fertigung 25 (1997) 9, S. 46 – 48.

Schnell 1997
Schnell, R.: Induktive Näherungsschalter mit Überwachungsfunktionalität. In: Installationstechnik an Werkzeugmaschinen (iwb Seminarberichte Nr. 29). München: Herbert Utz Verlag Wissenschaft 1997, S. 64 - 76.

Schnell 1999
Schnell, G.: Bussysteme in der Automatisierungstechnik. Grundlagen und Systeme in der industriellen Kommunikation. Wiesbaden: Vieweg 1999.

Schnieder & Gückel 1989
Schnieder, E.; Gückel, H.: Petrinetze in der Automatisierungstechnik. Automatisierungstechnik 37 (1989) 5, S. 173 – 181.

Schelberg 1994
Schelberg, H. J.: Objektorientierte Projektierung von SPS-Software. Universität Karlsruhe (TH) 1998.

Selic u. a. 1994
Selic, B.; Gullekson, G.; Ward, P. T.: Real Time Object Oriented Modeling. New York: John Wiley & Sons 1994.

Shlaer & Mellor 1992
Shlaer, S.; Mellor, S.: Object Lifecycles: Modeling the World in States. Englewood Cliffs: Yourdon Press 1992.

Siegler 1998
Siegler, R.: Mechatronischer Entwicklungsprozeß am Beispiel Werkzeugmaschinen. In: Reinhart, G.; Milberg, J.: Innovative Entwicklung von Produktionsmaschinen. Mechatronische Maschinen effizient entwickeln. München: Herbert Utz Verlag Wissenschaft 1998, S. 2/1 - 2/16. (iwb Seminarberichte 41).

Siemens 1999a

Siemens (Hrsg): SIMATIC. S7-GRAPH für S7-300/400. Ablaufsteuerungen Programmieren. Handbuch. Ausgabe 01. Erlangen: 1999.

Siemens 1999b

Siemens (Hrsg): SIMATIC. Basismodul 147/CPU. Handbuch. Ausgabe 03. Erlangen: 1999.

Specks 1997

Specks, J. W.: Läuft und läuft ... Halbleiter für die Automobilelektronik der Zukunft. Elektronikpraxis 32 (1997) 14, S. 34-35.

Strauss 1997

Strauss, W.: Veränderungen des Kunden/Lieferanten-Verhältnisses bei der Elektronikentwicklung. In: Systemengineering in der KFZ-Entwicklung, VDI-Bericht 1374. Düsseldorf, VDI-Verlag 1997.

Storr u. a. 1997

Storr, A.; Brandl, T.; Lutz, R.; Reichenbächer, J.: ASPECT - ein CASE-Tool zur systematischen Erstellung von Software im Maschinenbau. In: Schnittstellen im CAD/CAM-Bereich. München: Hanser 1997, S. 155 - 168.

Stroustrup 1997

Stroustrup, B.: The C++ Programming Language. Amsterdam: Addison-Wesley Longman 1997

Thomas 1999

Einfach auf der sicheren Seite. Integrierte Sicherheitsfunktionen bei der ET200S – Fehlersichere Kommunikation mit AS-Interface. Drive, Switch & Control 1 (1999) 3 S. 14 – 15.

Tomaszunas 1998

Tomaszunas, J.: Komponentenbasierte Maschinenmodellierung zur Echtzeit-Simulation für den Steuerungstest. München: Utz Verlag Wissenschaft 1998 (Informationstechnik im Maschinenwesen)

Tsujino u. a. 1984

Tsujino, Y.; Ando, T.; Araki, T.; Tokura, M.: Concurrent C: A Programming Language for Distributed Multiprocessor Systems. Software - Practice & Experience 14 (1984) 11, S. 1061 - 1078

VDW 1997a

N. N.: Installationstechnik an Werkzeugmaschinen. Abschlußbericht zum VDW-Projekt 0816. Frankfurt: Verein Deutscher Werkzeugmaschinenfabriken 1997.

VDW 1997b

N. N.: Werkzeug zur störungstoleranten Steuerung von Abläufen. Abschlußbericht zum VDW-Projekt 0817. Frankfurt: Verein Deutscher Werkzeugmaschinenfabriken 1997.

VDW 2000

N. N.: Dezentralisierte Steuerungstechnik. Abschlußbericht zum VDW-Projekt 0820. Frankfurt: Verein Deutscher Werkzeugmaschinenfabriken 2000.

VDI 3683 1986

VDI-Richtlinie 3683: Beschreibung von Steuerungsaufgaben, Anleitung zur Erstellung eines Pflichtenhefts. Berlin: Beuth 1986.

Wagner & Bürgel 1997

Wagner, P.; Bürgel, R.: DESINA – Ergebnisse des Projekts und des begleitenden Arbeitskreises. In: Reinhart, G.; Milberg, J.: Installationstechnik an Werkzeugmaschinen. München: Herbert Utz Verlag 1997, S. 16 - 63. (iwb Seminarberichte 29)

Wagner 1997

Wagner, M.: Fehlertolerante Steuerung maschinennaher Abläufe. Berlin: Springer 1997 (iwb Forschungsberichte 106).

Wagner 1995

Wagner, P.: Kostenanalysen und resultierende Neuerungsansätze. In: Reinhart, G.; Milberg, J.: Installationstechnik an Werkzeugmaschinen. München. Herbert Utz Verlag Wissenschaft 1995, S. 12 – 36. (iwb Seminarberichte 9)

Waldner 1996

Waldner, E.: Dezentrale Steuerungstechnik in Anlagen der Automobilindustrie. In: Reinhart, G.; Milberg, J.: Dezentrale Steuerungen in Produktionsanlagen. München: Herbert Utz Verlag Wissenschaft 1996, S. 47 - 67. (iwb Seminarberichte 20).

Warnier 1979

Warnier, J.-D.: Logical Construction of Programs. New York: Van Nostrand 1979.

Weck u. a. 1993

Weck, M.; Kohring, A.; Klein, F.: Offene NC-Systeme, Grundlage herstellerunabhängiger Flexibilität. VDI-Z 135 (1993) 5, S. 51 – 55.

Weck 1995

Weck, M.: Werkzeugmaschinen – Fertigungssysteme. Band 3.1: Automatisierung und Steuerungstechnik 1. Düsseldorf: VDI-Verlag 1995.

Weber 1990

Weber, H.: Monolithische Programmierung verteilter Automatisierungssysteme. Universität Karlsruhe: Dissertation 1990.

Witzack 2000

Witzak, M. P.: Echtzeit Betriebssysteme. Eine Einführung in Architektur und Programmierung. Feldkirchen: Franzis 2000

Zeller 1999

Zeller, W.: Sicherheitstechnisches Gesamtkonzept für Werkzeugmaschinen. In: Reinhart, G.; Milberg, J.: Sicherheitstechnik an Werkzeugmaschinen. München: Herbert Utz Verlag Wissenschaft 1999, S. 3/1 – 3/16. (iwb-Seminarberichte 48).

iwb Forschungsberichte Band 1–121

Herausgeber: Prof. Dr.-Ing. J. Milberg und Prof. Dr.-Ing. G. Reinhart, Institut für Werkzeugmaschinen und Betriebswissenschaften der Technischen Universität München

Band 1–121 erschienen im Springer Verlag, Berlin, Heidelberg und sind im Erscheinungsjahr und den folgenden drei Kalenderjahren erhältlich im Buchhandel oder durch Lange & Springer, Otto-Suhr-Allee 26–28, 10585 Berlin

1 *Streifinger, E.*
Beitrag zur Sicherung der Zuverlässigkeit und Verfügbarkeit moderner Fertigungsmittel
1986 · 72 Abb. · 167 Seiten · ISBN 3-540-16391-3

2 *Fuchsberger, A.*
Untersuchung der spanenden Bearbeitung von Knochen
1986 · 90 Abb. · 175 Seiten · ISBN 3-540-16392-1

3 *Maier, C.*
Montageautomatisierung am Beispiel des Schraubens mit Industrierobotern
1986 · 77 Abb. · 144 Seiten · ISBN 3-540-16393-X

4 *Summer, H.*
Modell zur Berechnung verzweigter Antriebsstrukturen
1986 · 74 Abb. · 197 Seiten · ISBN 3-540-16394-8

5 *Simon, W.*
Elektrische Vorschubantriebe an NC-Systemen
1986 · 141 Abb. · 198 Seiten · ISBN 3-540-16693-9

6 *Büchs, S.*
Analytische Untersuchungen zur Technologie der Kugelbearbeitung
1986 · 74 Abb. · 173 Seiten · ISBN 3-540-16694-7

7 *Hunzinger, I.*
Schneiderodierte Oberflächen
1986 · 79 Abb. · 162 Seiten · ISBN 3-540-16695-5

8 *Pilland, U.*
Echtzeit-Kollisionsschutz an NC-Drehmaschinen
1986 · 54 Abb. · 127 Seiten · ISBN 3-540-17274-2

9 *Barthelmeß, P.*
Montagegerechtes Konstruieren durch die Integration von Produkt- und Montageprozeßgestaltung
1987 · 70 Abb. · 144 Seiten · ISBN 3-540-18120-2

10 *Reithofer, N.*
Nutzungssicherung von flexibel automatisierten Produktionsanlagen
1987 · 84 Abb. · 176 Seiten · ISBN 3-540-18440-6

11 *Diess, H.*
Rechnerunterstützte Entwicklung flexibel automatisierter Montageprozesse
1988 · 56 Abb. · 144 Seiten · ISBN 3-540-18799-5

12 *Reinhart, G.*
Flexible Automatisierung der Konstruktion und Fertigung elektrischer Leitungssätze
1988 · 112 Abb. · 197 Seiten · ISBN 3-540-19003-1

13 *Bürstner, H.*
Investitionsentscheidung in der rechnerintegrierten Produktion
1988 · 74 Abb. · 190 Seiten · ISBN 3-540-19099-6

14 *Groha, A.*
Universelles Zellenrechnerkonzept für flexible Fertigungssysteme
1988 · 74 Abb. · 153 Seiten · ISBN 3-540-19182-8

15 *Riese, K.*
Klipsmontage mit Industrierobotern
1988 · 92 Abb. · 150 Seiten · ISBN 3-540-19183-6

16 *Lutz, P.*
Leitsysteme für rechnerintegrierte Auftragsabwicklung
1988 · 44 Abb. · 144 Seiten · ISBN 3-540-19260-3

17 *Klippel, C.*
Mobiler Roboter im Materialfluß eines flexiblen Fertigungssystems
1988 · 86 Abb. · 164 Seiten · ISBN 3-540-50468-0

18 *Rascher, R.*
Experimentelle Untersuchungen zur Technologie der Kugelherstellung
1989 · 110 Abb. · 200 Seiten · ISBN 3-540-51301-9

19 *Heusler, H.-J.*
Rechnerunterstützte Planung flexibler Montagesysteme
1989 · 43 Abb. · 154 Seiten · ISBN 3-540-51723-5

20 *Kirchknopf, P.*
Ermittlung modaler Parameter aus Übertragungsfrequenzgängen
1989 · 57 Abb. · 157 Seiten · ISBN 3-540-51724-3

21 *Sauerer, Ch.*
Beitrag für ein Zerspanprozeßmodell Metallbandsägen
1990 · 89 Abb. · 166 Seiten · ISBN 3-540-51868-1

22 *Karstedt, K.*
Positionsbestimmung von Objekten in der Montage- und Fertigungsautomatisierung
1990 · 92 Abb. · 157 Seiten · ISBN 3-540-51879-7

23 *Peiker, St.*
Entwicklung eines integrierten NC-Planungssystems
1990 · 66 Abb. · 180 Seiten · ISBN 3-540-51880-0

24 *Schugmann, R.*
Nachgiebige Werkzeugaufhängungen für die automatische Montage
1990 · 71 Abb. · 155 Seiten · ISBN 3-540-52138-0

25 *Wrba, P.*
Simulation als Werkzeug in der Handhabungstechnik
1990 · 125 Abb. · 178 Seiten · ISBN 3-540-52231-X

26 *Eibelshäuser, P.*
Rechnerunterstützte experimentelle Modalanalyse mittels gestufter Sinusanregung
1990 · 79 Abb. · 156 Seiten · ISBN 3-540-52451-7

27 *Prasch, J.*
Computerunterstützte Planung von chirurgischen Eingriffen in der Orthopädie
1990 · 113 Abb. · 164 Seiten · ISBN 3-540-52543-2

28 *Teich, K.*
Prozeßkommunikation und Rechnerverbund in der Produktion
1990 · 52 Abb. · 158 Seiten · ISBN 3-540-52764-8

29 *Pfrang, W.*
Rechnergestützte und graphische Planung manueller und teilautomatisierter Arbeitsplätze
1990 · 59 Abb. · 153 Seiten · ISBN 3-540-52829-6

30 *Tauber, A.*
Modellbildung kinematischer Strukturen als Komponente der Montageplanung
1990 · 93 Abb. · 190 Seiten · ISBN 3-540-52911-X

31 *Jäger, A.*
Systematische Planung komplexer Produktionssysteme
1991 · 75 Abb. · 148 Seiten · ISBN 3-540-53021-5

32 *Hartberger, H.*
Wissensbasierte Simulation komplexer Produktionssysteme
1991 · 58 Abb. · 154 Seiten · ISBN 3-540-53326-5

33 *Tuczek, H.*
Inspektion von Karosseriepreßteilen auf Risse und Einschnürungen mittels Methoden der Bildverarbeitung
1992 · 125 Abb. · 179 Seiten · ISBN 3-540-53965-4

34 *Fischbacher, J.*
Planungsstrategien zur stömungstechnischen Optimierung von Reinraum-Fertigungsgeräten
1991 · 60 Abb. · 166 Seiten · ISBN 3-540-54027-X

35 *Moser, O.*
3D-Echtzeitkollisionsschutz für Drehmaschinen
1991 · 66 Abb. · 177 Seiten · ISBN 3-540-54076-8

36 *Naber, H.*
Aufbau und Einsatz eines mobilen Roboters mit unabhängiger Lokomotions- und Manipulationskomponente
1991 · 85 Abb. · 139 Seiten · ISBN 3-540-54216-7

37 *Kupec, Th.*
Wissensbasiertes Leitsystem zur Steuerung flexibler Fertigungsanlagen
1991 · 68 Abb. · 150 Seiten · ISBN 3-540-54260-4

38 *Maulhardt, U.*
Dynamisches Verhalten von Kreissägen
1991 · 109 Abb. · 159 Seiten · ISBN 3-540-54365-1

39 *Götz, R.*
Strukturierte Planung flexibel automatisierter Montagesysteme für flächige Bauteile
1991 · 86 Abb. · 201 Seiten · ISBN 3-540-54401-1

40 *Koepfer, Th.*
3D-grafisch-interaktive Arbeitsplanung - ein Ansatz zur Aufhebung der Arbeitsteilung
1991 · 74 Abb. · 126 Seiten · ISBN 3-540-54436-4

41 *Schmidt, M.*
Konzeption und Einsatzplanung flexibel automatisierter Montagesysteme
1992 · 108 Abb. · 168 Seiten · ISBN 3-540-55025-9

42 *Burger, C.*
Produktionsregelung mit entscheidungsunterstützenden Informationssystemen
1992 · 94 Abb. · 186 Seiten · ISBN 3-540-55187-5

43 *Hoßmann, J.*
Methodik zur Planung der automatischen Montage von nicht formstabilen Bauteilen
1992 · 73 Abb. · 168 Seiten · ISBN 3-540-5520-0

44 *Petry, M.*
Systematik zur Entwicklung eines modularen Programmbaukastens für robotergeführte Klebeprozesse
1992 · 106 Abb. · 139 Seiten · ISBN 3-540-55374-6

45 *Schönecker, W.*
Integrierte Diagnose in Produktionszellen
1992 · 87 Abb. · 159 Seiten · ISBN 3-540-55375-4

46 *Bick, W.*
Systematische Planung hybrider Montagesysteme unter Berücksichtigung der Ermittlung des optimalen Automatisierungsgrades
1992 · 70 Abb. · 156 Seiten · ISBN 3-540-55377-0

47 *Gebauer, L.*
Prozeßuntersuchungen zur automatisierten Montage von optischen Linsen
1992 · 84 Abb. · 150 Seiten · ISBN 3-540-55378-9

48 *Schrüfer, N.*
Erstellung eines 3D-Simulationssystems zur Reduzierung von Rüstzeiten bei der NC-Bearbeitung
1992 · 103 Abb. · 161 Seiten · ISBN 3-540-55431-9

49 *Wisbacher, J.*
Methoden zur rationellen Automatisierung der Montage von Schnellbefestigungselementen
1992 · 77 Abb. · 176 Seiten · ISBN 3-540-55512-9

50 *Garnich, F.*
Laserbearbeitung mit Robotern
1992 · 110 Abb. · 184 Seiten · ISBN 3-540-55513-7

51 *Eubert, P.*
Digitale Zustandesregelung elektrischer Vorschubantriebe
1992 · 89 Abb. · 159 Seiten · ISBN 3-540-44441-2

52 *Glaas, W.*
Rechnerintegrierte Kabelsatzfertigung
1992 · 67 Abb. · 140 Seiten · ISBN 3-540-55749-0

53 *Helml, H.J.*
Ein Verfahren zur On-Line Fehlererkennung und Diagnose
1992 · 60 Abb. · 153 Seiten · ISBN 3-540-55750-4

54 *Lang, Ch.*
Wissensbasierte Unterstützung der Verfügbarkeitsplanung
1992 · 75 Abb. · 150 Seiten · ISBN 3-540-55751-2

55 *Schuster, G.*
Rechnergestütztes Planungssystem für die flexibel automatisierte Montage
1992 · 67 Abb. · 135 Seiten · ISBN 3-540-55830-6

56 *Bomm, H.*
Ein Ziel- und Kennzahlensystem zum Investitionscontrolling komplexer Produktionssysteme
1992 · 87 Abb. · 195 Seiten · ISBN 3-540-55964-7

57 *Wendt, A.*
Qualitätssicherung in flexibel automatisierten Montagesystemen
1992 · 74 Abb. · 179 Seiten · ISBN 3-540-56044-0

58 *Hansmaier, H.*
Rechnergestütztes Verfahren zur Geräuschminderung
1993 · 67 Abb. · 156 Seiten · ISBN 3-540-56053-2

59 *Dilling, U.*
Planung von Fertigungssystemen unterstützt durch Wirtschaftssimulationen
1993 · 72 Abb. · 146 Seiten · ISBN 3-540-56307-5

60 *Strohmayr, R.*
Rechnergestützte Auswahl und Konfiguration von Zubringeeinrichtungen
1993 · 80 Abb. · 152 Seiten · ISBN 3-540-56652-X

61 *Glas, J.*
Standardisierter Aufbau anwendungsspezifischer Zellenrechnersoftware
1993 · 80 Abb. · 145 Seiten · ISBN 3.540-56890-5

62 *Stetter, R.*
Rechnergestütze Simulationswerkzeuge zur Effizienzsteigerung des Industrierobotereinsatzes
1994 · 91 Abb. · 146 Seiten · ISBN 3-540-56889-1

63 *Dirndorfer, A.*
Robotersysteme zur förderbandsynchronen Montage
1993 · 76 Abb. · 144 Seiten · ISBN 3-540-57031-4

64 *Wiedemann, M.*
Simulation des Schwingungsverhaltens spanender Werkzeugmaschinen
1993 · 81 Abb. · 137 Seiten · ISBN 3-540-57177-9

65 *Woenckhaus, Ch.*
Rechnergestütztes System zur automatisierten 3D-Layoutoptimierung
1994 · 81 Abb. · 140 Seiten · ISBN 3-540-57284-8

66 *Kummetsteiner, G.*
3D-Bewegungssimulation als integratives Hilfsmittel zur Planung manueller Montagesysteme
1994 · 62 Abb. · 146 Seiten · ISBN 3-540-57535-9

67 *Kugelmann, F.*
Einsatz nachgiebiger Elemente zur wirtschaftlichen Automatisierung von Produktionssystemen
1993 · 76 Abb. · 144 Seiten · ISBN 3-540-57549-9

68 *Schwarz, H.*
Simulationsgestützte CAD/CAM-Kopplung für die 3D-Laserbearbeitung mit integrierter Sensorik
1994 · 96 Abb. · 148 Seiten · ISBN 3-540-57577-4

69 *Viethen, U.*
Systematik zum Prüfen in flexiblen Fertigungssystemen
1994 · 70 Abb. · 142 Seiten · ISBN 3-540-57794-7

70 *Seehuber, M.*
Automatische Inbetriebnahme geschwindigkeitsadaptiver Zustandsregler
1994 · 72 Abb. · 155 Seiten · ISBN 3-540-57896-X

71 *Amann, W.*
Eine Simulationsumgebung für Planung und Betrieb von Produktionssystemen
1994 · 71 Abb. · 129 Seiten · ISBN 3-540-57924-9

72 *Schöpf, M.*
Rechnergestütztes Projektinformations- und Koordinationssystem für das Fertigungsvorfeld
1997 · 63 Abb. · 130 Seiten · ISBN 3-540-58052-2

73 *Welling, A.*
Effizienter Einsatz bildgebender Sensoren zur Flexibilisierung automatisierter Handhabungsvorgänge
1994 · 66 Abb. · 139 Seiten · ISBN 3-540-580-0

74 *Zetlmayer, H.*
Verfahren zur simulationsgestützten Produktionsregelung in der Einzel- und Kleinserienproduktion
1994 · 62 Abb. · 143 Seiten · ISBN 3-540-58134-0

75 *Lindl, M.*
Auftragsleittechnik für Konstruktion und Arbeitsplanung
1994 · 66 Abb. · 147 Seiten · ISBN 3-540-58221-5

76 *Zipper, B.*
Das integrierte Betriebsmittelwesen - Baustein einer flexiblen Fertigung
1994 · 64 Abb. · 147 Seiten · ISBN 3-540-58222-3

77 *Raith, P.*
Programmierung und Simulation von Zellenabläufen in der Arbeitsvorbereitung
1995 · 51 Abb. · 130 Seiten · ISBN 3-540-58223-1

78 *Engel, A.*
Strömungstechnische Optimierung von Produktionssystemen durch Simulation
1994 · 69 Abb. · 160 Seiten · ISBN 3-540-58258-4

79 *Zäh, M. F.*
Dynamisches Prozeßmodell Kreissägen
1995 · 95 Abb. · 186 Seiten · ISBN 3-540-58624-5

80 *Zwanzer, N.*
Technologisches Prozeßmodell für die Kugelschleifbearbeitung
1995 · 65 Abb. · 150 Seiten · ISBN 3-540-58634-2

81 *Romanow, P.*
Konstruktionsbegleitende Kalkulation von Werkzeugmaschinen
1995 · 66 Abb. · 151 Seiten · ISBN 3-540-58771-3

82 *Kahlenberg, R.*
Integrierte Qualitätssicherung in flexiblen Fertigungszellen
1995 · 71 Abb. · 136 Seiten · ISBN 3-540-58772-1

83 *Huber, A.*
Arbeitsfolgenplanung mehrstufiger Prozesse in der Hartbearbeitung
1995 · 87 Abb. · 152 Seiten · ISBN 3-540-58773-X

84 *Birkel, G.*
Aufwandsminimierter Wissenserwerb für die Diagnose in flexiblen Produktionszellen
1995 · 64 Abb. · 137 Seiten · ISBN 3-540-58869-8

85 *Simon, D.*
Fertigungsregelung durch zielgrößenorientierte Planung und logistisches Störungsmanagement
1995 · 77 Abb. · 132 Seiten · ISBN 3-540-58942-2

86 *Nedeljkovic-Groha, V.*
Systematische Planung anwendungsspezifischer Materialflußsteuerungen
1995 · 94 Abb. · 188 Seiten · ISBN 3-540-58953-8

87 *Rockland, M.*
Flexibilisierung der automatischen Teilbereitstellung in Montageanlagen
1995 · 83 Abb. · 168 Seiten · ISBN 3-540-58999-6

88 *Linner, St.*
Konzept einer integrierten Produktentwicklung
1995 · 67 Abb. · 168 Seiten · ISBN 3-540-59016-1

89 *Eder, Th.*
Integrierte Planung von Informationssystemen für rechnergestützte Produktionssysteme
1995 · 62 Abb. · 150 Seiten · ISBN 3-540-59084-6

90 *Deutschle, U.*
Prozeßorientierte Organisation der Auftragsentwicklung in mittelständischen Unternehmen
1995 · 80 Abb. · 188 Seiten · ISBN 3-540-59337-3

91 *Dieterle, A.*
Recyclingintegrierte Produktentwicklung
1995 · 68 Abb. · 146 Seiten · ISBN 3-540-60120-1

92 *Hechl, Chr.*
Personalorientierte Montageplanung für komplexe und variantenreiche Produkte
1995 · 73 Abb. · 158 Seiten · ISBN 3-540-60325-5

93 *Albertz, F.*
Dynamikgerechter Entwurf von Werkzeugmaschinen - Gestellstrukturen
1995 · 83 Abb. · 156 Seiten · ISBN 3-540-60608-8

94 *Trunzer, W.*
Strategien zur On-Line Bahnplanung bei Robotern mit 3D-Konturfolgesensoren
1996 · 101 Abb. · 164 Seiten · ISBN 3-540-60961-X

95 *Fichtmüller, N.*
Rationalisierung durch flexible, hybride Montagesysteme
1996 · 83 Abb. · 145 Seiten · ISBN 3-540-60960-1

96 *Trucks, V.*
Rechnergestützte Beurteilung von Getriebestrukturen in Werkzeugmaschinen
1996 · 64 Abb. · 141 Seiten · ISBN 3-540-60599-8

97 *Schäffer, G.*
Systematische Integration adaptiver Produktionssysteme
1996 · 71 Abb. · 170 Seiten · ISBN 3-540-60958-X

98 *Koch, M. R.*
Autonome Fertigungszellen - Gestaltung, Steuerung und integrierte Störungsbehandlung
1996 · 67 Abb. · 138 Seiten · ISBN 3-540-61104-5

99 *Moctezuma de la Barrera, J.L.*
Ein durchgängiges System zur computer- und rechnergestützten Chirurgie
1996 · 99 Abb. · 175 Seiten · ISBN 3-540-61145-2

100 *Geuer, A.*
Einsatzpotential des Rapid Prototyping in der Produktentwicklung
1996 · 84 Abb. · 154 Seiten · ISBN 3-540-61495-8

101 *Ebner, C.*
Ganzheitliches Verfügbarkeits- und Qualitätsmanagment unter Verwendung von Felddaten
1996 · 67 Abb. · 132 Seiten · ISBN 3-540-61678-0

102 *Pischeltsrieder, K.*
Steuerung autonomer mobiler Roboter in der Produktion
1996 · 74 Abb. · 171 Seiten · ISBN 3-540-61714-0

103 *Köhler, R.*
Disposition und Materialbereitstellung bei komplexen variantenreichen Kleinprodukten
1997 · 62 Abb. · 177 Seiten · ISBN 3-540-62024-9

104 *Feldmann, Ch.*
Eine Methode für die integrierte rechnergestützte Montageplanung
1997 · 71 Abb. · 163 Seiten · ISBN 3-540-62059-1

105 *Lehmann, H.*
Integrierte Materialfluß- und Layoutplanung durch Kopplung von CAD- und Ablaufsimulationssystem
1997 · 96 Abb. · 191 Seiten · ISBN 3-540-62202-0

106 *Wagner, M.*
Steuerungsintegrierte Fehlerbehandlung für maschinennahe Abläufe
1997 · 94 Abb. · 164 Seiten · ISBN 3-540-62656-5

107 *Lorenzen, J.*
Simulationsgestützte Kostenanalyse in produktorientierten Fertigungsstrukturen
1997 · 63 Abb. · 129 Seiten · ISBN 3-540-62794-4

108 *Krönert, U.*
Systematik für die rechnergestützte Ähnlichteilsuche und Standardisierung
1997 · 53 Abb. · 127 Seiten · ISBN 3-540-63338-3

109 *Pfersdorf, I.*
Entwicklung eines systematischen Vorgehens zur Organisation des industriellen Service
1997 · 74 Abb. · 172 Seiten · ISBN 3-540-63615-3

110 *Kuba, R.*
Informations- und kommunikationstechnische Integration von Menschen in der Produktion
1997 · 77 Abb. · 155 Seiten · ISBN 3-540-63642-0

111 *Kaiser, J.*
Vernetztes Gestalten von Produkt und Produktionsprozeß mit Produktmodellen
1997 · 67 Abb. · 139 Seiten · ISBN 3-540-63999-3

112 *Geyer, M.*
Flexibles Planungssystem zur Berücksichtigung ergonomischer Aspekte bei der Produkt- und Arbeitssystemgestaltung
1997 · 85 Abb. · 154 Seiten · ISBN 3-540-64195-5

113 *Martin, C.*
Produktionsregelung - ein modularer, modellbasierter Ansatz
1998 · 73 Abb. · 162 Seiten · ISBN 3-540-64401-6

114 *Löffler, Th.*
Akustische Überwachung automatisierter Fügeprozesse
1998 · 85 Abb. · 136 Seiten · ISBN 3-540-64511-X

115 *Lindermaier, R.*
Qualitätsorientierte Entwicklung von Montagesystemen
1998 · 84 Abb. · 164 Seiten · ISBN 3-540-64686-8

116 *Koehrer, J.*
Prozeßorientierte Teamstrukturen in Betrieben mit Großserienfertigung
1998 · 75 Abb. · 185 Seiten · ISBN 3-540-65037-7

117 *Schuller, R. W.*
Leitfaden zum automatisierten Auftrag von hochviskosen Dichtmassen
1999 · 76 Abb. · 162 Seiten · ISBN 3-540-65320-1

118 *Debuschewitz, M.*
Integrierte Methodik und Werkzeuge zur herstellungsorientierten Produktentwicklung
1999 · 104 Abb. · 169 Seiten · ISBN 3-540-65350-3

119 *Bauer, L.*
Strategien zur rechnergestützten Offline-Programmierung von 3D-Laseranlagen
1999 · 98 Abb. · 145 Seiten · ISBN 3-540-65382-1

120 *Pfob, E.*
Modellgestützte Arbeitsplanung bei Fertigungsmaschinen
1999 · 69 Abb. · 154 Seiten · ISBN 3-540-65525-5

121 *Spitznagel, J.*
Erfahrungsgeleitete Planung von Laseranlagen
1999 · 63 Abb. · 156 Seiten · ISBN 3-540-65896-3

Seminarberichte iwb

herausgegeben von Prof. Dr.-Ing. Gunther Reinhart, Institut für Werkzeugmaschinen und Betriebswissenschaften der Technischen Universität München

Seminarberichte iwb sind erhältlich im Buchhandel oder beim
Herbert Utz Verlag, München, Fax 089-277791-01, utz@utzverlag.com

1 Innovative Montagesysteme · Anlagengestaltung, -bewertung und -überwachung
115 Seiten · ISBN 3-931327-01-9

2 Integriertes Produktmodell · Von der Idee zum fertigen Produkt
82 Seiten · ISBN 3-931327-02-7

3 Konstruktion von Werkzeugmaschinen · Berechnung, Simulation und Optimierung
110 Seiten · ISBN 3-931327-03-5

4 Simulation · Einsatzmöglichkeiten und Erfahrungsberichte
134 Seiten · ISBN 3-931327-04-3

5 Optimierung der Kooperation in der Produktentwicklung
95 Seiten · ISBN 3-931327-05-1

6 Materialbearbeitung mit Laser · von der Planung zur Anwendung
86 Seiten · ISBN 3-931327-76-0

7 Dynamisches Verhalten von Werkzeugmaschinen
80 Seiten · ISBN 3-931327-77-9

8 Qualitätsmanagement · der Weg ist das Ziel
130 Seiten · ISBN 3-931327-78-7

9 Installationstechnik an Werkzeugmaschinen · Analysen und Konzepte
120 Seiten · ISBN 3-931327-79-5

10 3D-Simulation · Schneller, sicherer und kostengünstiger zum Ziel
90 Seiten · ISBN 3-931327-10-8

11 Unternehmensorganisation · Schlüssel für eine effiziente Produktion
110 Seiten · ISBN 3-931327-11-6

12 Autonome Produktionssysteme
100 Seiten · ISBN 3-931327-12-4

13 Planung von Montageanlagen
130 Seiten · ISBN 3-931327-13-2

15 Flexible fluide Kleb/Dichtstoffe · Dosierung und Prozeßgestaltung
80 Seiten · ISBN 3-931327-15-9

16 Time to Market · Von der Idee zum Produktionsstart
80 Seiten · ISBN 3-931327-16-7

17 Industriekeramik in Forschung und Praxis · Probleme, Analysen und Lösungen
80 Seiten · ISBN 3-931327-17-5

18 Das Unternehmen im Internet · Chancen für produzierende Unternehmen
165 Seiten · ISBN 3-931327-18-3

19 Leittechnik und Informationslogistik · mehr Transparenz in der Fertigung
85 Seiten · ISBN 3-931327-19-1

20 Dezentrale Steuerungen in Produktionsanlagen · Plug & Play · Vereinfachung von Entwicklung und Inbetriebnahme
105 Seiten · ISBN 3-931327-20-5

21 Rapid Prototyping · Rapid Tooling · Schnell zu funktionalen Prototypen
95 Seiten · ISBN 3-931327-21-3

22 Mikrotechnik für die Produktion · Greifbare Produkte und Anwendungspotentiale
95 Seiten · ISBN 3-931327-22-1

24 EDM Engineering Data Management
195 Seiten · ISBN 3-931327-24-8

25 Rationelle Nutzung der Simulationstechnik · Entwicklungstrends und Praxisbeispiele
152 Seiten · ISBN 3-931327-25-6

26 Alternative Dichtungssysteme · Konzepte zur Dichtungsmontage und zum Dichtmittelauftrag
110 Seiten · ISBN 3-931327-26-4

27 Rapid Prototyping · Mit neuen Technologien schnell vom Entwurf zum Serienprodukt
111 Seiten · ISBN 3-931327-27-2

28 Rapid Tooling · Mit neuen Technologien schnell vom Entwurf zum Serienprodukt
154 Seiten · ISBN 3-931327-28-0

29 Installationstechnik an Werkzeugmaschinen · Abschlußseminar
156 Seiten · ISBN 3-931327-29-9

31 Engineering Data Management (EDM) · Erfahrungsberichte und Trends
183 Seiten · ISBN 3-931327-31-0

33 3D-CAD · Mehr als nur eine dritte Dimension
181 Seiten · ISBN 3-931327-33-7

34 Laser in der Produktion · Technologische Randbedingungen für den wirtschaftlichen Einsatz
102 Seiten · ISBN 3-931327-34-5

35 Ablaufsimulation · Anlagen effizient und sicher planen und betreiben
129 Seiten · ISBN 3-931327-35-3

36 Moderne Methoden zur Montageplanung · Schlüssel für eine effiziente Produktion
124 Seiten · ISBN 3-931327-36-1

37 Wettbewerbsfaktor Verfügbarkeit · Produktivitätsteigerung durch technische und organisatorische Ansätze
95 Seiten · ISBN 3-931327-37-X

38 Rapid Prototyping · Effizienter Einsatz von Modellen in der Produktentwicklung
128 Seiten · ISBN 3-931327-38-8

39 Rapid Tooling · Neue Strategien für den Werkzeug- und Formenbau
130 Seiten · ISBN 3-931327-39-6

40 Erfolgreich kooperieren in der produzierenden Industrie · Flexibler und schneller mit modernen Kooperationen
160 Seiten · ISBN 3-931327-40-X

41 Innovative Entwicklung von Produktionsmaschinen
146 Seiten · ISBN 3-89675-041-0

42 Stückzahlflexible Montagesysteme
139 Seiten · ISBN 3-89675-042-9

43 Produktivität und Verfügbarkeit · ...durch Kooperation steigern
120 Seiten · ISBN 3-89675-043-7

44 Automatisierte Mikromontage · Handhaben und Positionieren von Mikrobauteilen
125 Seiten · ISBN 3-89675-044-5

45 Produzieren in Netzwerken · Lösungsansätze, Methoden, Praxisbeispiele
173 Seiten · ISBN 3-89675-045-3

46 Virtuelle Produktion · Ablaufsimulation
108 Seiten · ISBN 3-89675-046-1

47 Virtuelle Produktion · Prozeß- und Produktsimulation
131 Seiten · ISBN 3-89675-047-X

48 Sicherheitstechnik an Werkzeugmaschinen
106 Seiten · ISBN 3-89675-048-8

49 Rapid Prototyping · Methoden für die reaktionsfähige Produktentwicklung
150 Seiten · ISBN 3-89675-049-6

50 Rapid Manufacturing · Methoden für die reaktionsfähige Produktion
121 Seiten · ISBN 3-89675-050-X

51 Flexibles Kleben und Dichten · Produkt-& Prozeßgestaltung, Mischverbindungen, Qualitätskontrolle
137 Seiten · ISBN 3-89675-051-8

52 Rapid Manufacturing · Schnelle Herstellung von Klein- und Prototypenserien
124 Seiten · ISBN 3-89675-052-6

53 Mischverbindungen · Werkstoffauswahl, Verfahrensauswahl, Umsetzung
107 Seiten · ISBN 3-89675-054-2

54 Virtuelle Produktion · Integrierte Prozess- und Produktsimulation
133 Seiten · ISBN 3-89675-054-2

55 e-Business in der Produktion · Organisationskonzepte, IT-Lösungen, Praxisbeispiele
150 Seiten · ISBN 3-89675-055-0

56 Virtuelle Produktion – Ablaufsimulation als planungsbegleitendes Werkzeug
150 Seiten · ISBN 3-89675-056-9

57 Virtuelle Produktion – Datenintegration und Benutzerschnittstellen
150 Seiten · ISBN 3-89675-057-7

58 Rapid Manufacturing · Schnelle Herstellung qualitativ hochwertiger Bauteile oder Kleinserien
169 Seiten · ISBN 3-89675-058-7

59 Automatisierte Mikromontage · Werkzeuge und Fügetechnologien für die Mikrosystemtechnik
114 Seiten · ISBN 3-89675-059-3

60 Mechatronische Produktionssysteme · Genauigkeit gezielt entwickeln
131 Seiten · ISBN 3-89675-060-7

Forschungsberichte iwb

herausgegeben von Prof. Dr.-Ing. Gunther Reinhart, Institut für Werkzeugmaschinen und Betriebswissenschaften der Technischen Universität München

Forschungsberichte iwb ab Band 122 sind erhältlich im Buchhandel oder beim Herbert Utz Verlag, München, Fax 089-277791-01, utz@utzverlag.com

122 Schneider, Burghard
Prozesskettenorientierte Bereitstellung nicht formstabiler Bauteile
1999 · 183 Seiten · 98 Abb. · 14 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-559-5

123 Goldstein, Bernd
Modellgestützte Geschäftsprozeßgestaltung in der Produktentwicklung
1999 · 170 Seiten · 65 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-546-3

124 Mößmer, Helmut E.
Methode zur simulationsbasierten Regelung zeitvarianter Produktionssysteme
1999 · 164 Seiten · 67 Abb. · 5 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-585-4

125 Gräser, Ralf-Gunter
Ein Verfahren zur Kompensation temperaturinduzierter Verformungen an Industrierobotern
1999 · 167 Seiten · 63 Abb. · 5 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-603-6

126 Trossin, Hans-Jürgen
Nutzung der Ähnlichkeitstheorie zur Modellbildung in der Produktionstechnik
1999 · 162 Seiten · 75 Abb. · 11 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-614-1

127 Kugelmann, Doris
Aufgabenorientierte Offline-Programmierung von Industrierobotern
1999 · 168 Seiten · 68 Abb. · 2 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-615-X

128 Diesch, Rolf
Steigerung der organisatorischen Verfügbarkeit von Fertigungszellen
1999 · 160 Seiten · 69 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-618-4

129 Lulay, Werner E.
Hybrid-hierarchische Simulationsmodelle zur Koordination teilautonomer Produktionsstrukturen
1999 · 182 Seiten · 51 Abb. · 14 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-620-6

130 Murr, Otto
Adaptive Planung und Steuerung von integrierten Entwicklungs- und Planungsprozessen
1999 · 178 Seiten · 85 Abb. · 3 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-636-2

131 Macht, Michael
Ein Vorgehensmodell für den Einsatz von Rapid Prototyping
1999 · 170 Seiten · 87 Abb. · 5 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-638-9

132 Mehler, Bruno H.
Aufbau virtueller Fabriken aus dezentralen Partnerverbünden
1999 · 152 Seiten · 44 Abb. · 27 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-645-1

133 Heitmann, Knut
Sichere Prognosen für die Produktionsptimierung mittels stochastischer Modelle
1999 · 146 Seiten · 60 Abb. · 13 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-675-3

134 Blessing, Stefan
Gestaltung der Materialflußsteuerung in dynamischen Produktionsstrukturen
1999 · 160 Seiten · 67 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-690-7

135 Abay, Can
Numerische Optimierung multivariater mehrstufiger Prozesse am Beispiel der Hartbearbeitung von Industriekeramik
2000 · 159 Seiten · 46 Abb. · 5 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-697-4

136 Brandner, Stefan
Integriertes Produktdaten- und Prozeßmanagement in virtuellen Fabriken
2000 · 172 Seiten · 61 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-715-6

137 Hirschberg, Arnd G.
Verbindung der Produkt- und Funktionsorientierung in der Fertigung
2000 · 165 Seiten · 49 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-729-6

138 Reek, Alexandra
Strategien zur Fokuspositionierung beim Laserstrahlschweißen
2000 · 193 Seiten · 103 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-730-X

139 Sabbah, Khalid-Alexander
Methodische Entwicklung störungstoleranter Steuerungen
2000 · 148 Seiten · 75 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-739-3

140 Schliffenbacher, Klaus U.
Konfiguration virtueller Wertschöpfungsketten in dynamischen, heterarchischen Kompetenznetzwerken
2000 · 187 Seiten · 70 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-754-7

141 Sprenzel, Andreas
Integrierte Kostenkalkulationsverfahren für die Werkzeugmaschinenentwicklung
2000 · 144 Seiten · 55 Abb. · 6 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-757-1

142 Gallasch, Andreas
Informationstechnische Architektur zur Unterstützung des Wandels in der Produktion
2000 · 150 Seiten · 69 Abb. · 6 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-781-4

143 Cuiper, Ralf
Durchgängige rechnergestützte Planung und Steuerung von automatisierten Montagevorgängen
2000 · 168 Seiten · 75 Abb. · 3 Tab. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-783-0 · lieferbar ab ca. 02/01

144 Schneider, Christian
Strukturmechanische Berechnungen in der Werkzeugmaschinenkonstruktion
2000 · 180 Seiten · 66 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-789-X

145 Jonas, Christian
Konzept einer durchgängigen, rechnergestützten Planung von Montageanlagen
2000 · 183 Seiten · 82 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-870-5

146 Willnecker, Ulrich
Gestaltung und Planung leistungsorientierter manueller Fließmontagen
2001 · 175 Seiten · 67 Abb. · broschiert · 20,5 x 14,5 cm · ISBN 3-89675-891-8

147 Lehner, Christof
Beschreibung des Nd:Yag-Laserstrahlschweißprozesses von Magnesiumdruckguss
2001 · 205 Seiten · 94 Abb. · 24 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0004-X

148 Rick, Frank
Simulationsgestützte Gestaltung von Produkt und Prozess am Beispiel Laserstrahlschweißen
2001 · 145 Seiten · 57 Abb. · 2 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0008-2

149 Höhn, Michael
Sensorgeführte Montage hybrider Mikrosysteme
2001 · 171 Seiten · 74 Abb. · 7 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0012-0

150 Böhl, Jörn
Wissensmanagement im Klein- und mittelständischen Unternehmen der Einzel- und Kleinserienfertigung
2001 · 179 Seiten · 88 Abb. · 20,5 x 14,5 cm · ISBN 3-8316-0020-1

151 Bürgel, Robert
Prozessanalyse an spanenden Werkzeugmaschinen mit digital geregelten Antrieben
2001 · 185 Seiten · 60 Abb. · 10 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0021-X
lieferbar ab ca. 09/01

152 Stephan Dürrschmidt
Planung und Betrieb wandlungsfähiger Logistiksysteme in der variantenreichen Serienproduktion
2001 · 914 Seiten · 61 Abb. · 20,5 x 14,5 cm · ISBN 3-8316-0023-6

153 Bernhard Eich
Methode zur prozesskettenorientierten Planung der Teilebereitstellung
2001 · 132 Seiten · 48 Abb. · 6 Tabellen · 20,5 x 14,5 cm · ISBN 3-8316-0028-7

154 Wolfgang Rudorfer
Eine Methode zur Qualifizierung von produzierenden Unternehmen für Kompetenznetzwerke
2001 · 207 Seiten · 89 Abb. · 20,5 x 14,5 cm · ISBN 3-8316-0037-6

155 Hans Meier
Verteilte kooperative Steuerung maschinennaher Abläufe
2001 · 162 Seiten · 85 Abb. · 20,5 x 14,5 cm · ISBN 3-8316-0044-9

156 Gerhard Nowak
Informationstechnische Integration des industriellen Service in das Unternehmen
2001 · 203 Seiten · 95 Abb. · 20,5 x 14,5 cm · ISBN 3-8316-0055-4

157 Martin Werner
Simulationsgestützte Reorganisation von Produktions- und Logistikprozessen
2001 · 191 Seiten · 20,5 x 14,5 cm · ISBN 3-8316-0058-9

158 Bernhard Lenz
Finite Elemente-Modellierung des Laserstrahlschweißens für den Einsatz in der Fertigungsplanung
2001 · 150 Seiten · 47 Abb. · 5 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0094-5

159 Stefan Grunwald
Methode zur Anwendung der flexiblen integrierten Produktentwicklung und Montageplanung
2002 · 206 Seiten · 80 Abb. · 25 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0095-3

160 Josef Gartner
Qualitätssicherung bei der automatisierten Applikation hochviskoser Dichtungen
2002 · 165 Seiten · 74 Abb. · 21 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0096-1

161 Wolfgang Zeller
Gesamtheitliches Sicherheitskonzept für die Antriebs- und Steuerungstechnik bei Werkzeugmaschinen
2002 · 192 Seiten · 54 Abb. · 15 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0100-3

162 Michael Loferer
Rechnergestützte Gestaltung von Montagesystemen
2002 · 178 Seiten · 80 Abb. · 20,5 x 14,5 cm · ISBN 3-8316-0118-6

163 Jörg Fährer
Ganzheitliche Optimierung des indirekten Metall-Lasersinterprozesses
2002 · 176 Seiten · 69 Abb. · 13 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0124-0

164 Jürgen Höppner
Verfahren zur berührungslosen Handhabung mittels leistungsstarker Schallwandler
2002 · 132 Seiten · 24 Abb. · 3 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0125-9

165 Hubert Götte
Entwicklung eines Assistenzrobotersystems für die Knieendoprothetik
2002 · 258 Seiten · 123 Abb. · 5 Tab. · 20,5 x 14,5 cm · ISBN 3-8316-0126-7