Matthias Wenk

Entwicklung eines konfigurierbaren Steuerungssystems für die flexible Sensorführung von Industrierobotern

Matthias Wenk

Entwicklung eines konfigurierbaren Steuerungssystems für die flexible Sensorführung von Industrierobotern

Herausgegeben von
Professor Dr.-Ing. Klaus Feldmann,
Lehrstuhl für
Fertigungsautomatisierung und Produktionssystematik

FAPS

Meisenbach Verlag Bamberg

Als Dissertation genehmigt von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der Einreichung:	07. Januar 2002
Tag der Promotion:	29. Mai 2002
Dekan:	Prof. Dr. rer. nat. A. Winnacker
Berichterstatter:	Prof. Dr.-Ing. K. Feldmann
	apl. Prof. Dr.-Ing. habil. Dr. h.c. W. Bär

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Wenk, Matthias:
Entwicklung eines konfigurierbaren Steuerungssystems
für die flexible Sensorführung von Industrierobotern /
Matthias Wenk. - Bamberg : Meisenbach, 2002
(Fertigungstechnik - Erlangen ; 131)
ISBN 3-87525-174-1

ISSN 1431-6226

Herstellung: Gruner Druck GmbH, Erlangen-Eltersdorf
Printed in Germany

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik der Friedrich-Alexander-Universität Erlangen-Nürnberg.

Herrn Prof. Dr.-Ing. Klaus Feldmann, dem Leiter dieses Lehrstuhls am Institut für Fertigungstechnik und Maschinenbau, danke ich für die Förderung bei der Durchführung meiner Arbeit sowie für den wissenschaftlichen Freiraum und die Unterstützung, die er mir bei der Bearbeitung der praxisorientierten Problemstellungen gewährte.

Herrn apl. Prof. Dr.-Ing. Dr. h.c. Wolfgang Bär möchte ich meinen Dank aussprechen für die engagierte Übernahme des Koreferats sowie die interessanten fachlichen Diskussionen im Rahmen von gemeinsamen Studien- und Diplomarbeiten am Lehrstuhl für Regelungstechnik der Universität Erlangen-Nürnberg.

Diese Dissertation beruht zum großen Teil auf Arbeiten, die im Rahmen der bayerischen Forschungsverbünde FORLAS I und II durchgeführt wurden. Der Bayerischen Forschungsstiftung und den Industriepartnern danke ich für die finanzielle Förderung. Die fachlichen Diskussionen mit den beteiligten Industriepartnern haben die Praxisrelevanz dieser Arbeit wesentlich gefördert. Dafür danke ich Herrn Dr.-Ing. Arif Kazi und Herrn Dr.-Ing. Gerd Schneider (KUKA Roboter GmbH), Herrn Dipl.-Ing. Hans-Richard Tradt (KUKA Schweißanlagen GmbH), Herrn Dipl.-Phys. Markus Kogel-Hollacher und Herrn Christoph Dietz (JURCA Optoelektronik GmbH).

Mein besonderer Dank gilt Herrn Dr.-Ing. Peter Hoffmann und Herrn Dipl.-Ing. Peter Kugler (Erlanger Lasertechnik GmbH) für die enge Zusammenarbeit in FORLAS II und Herrn Dipl.-Ing. Detlef Krause und Herrn Dipl.-Ing. Sven Wiegand (Photon Laser Engineering GmbH) für die gute Zusammenarbeit beim Technologietransfer meiner Arbeit.

Weiterhin danke ich allen Kollegen und Mitarbeitern am Lehrstuhl FAPS für die fachlichen Diskussionen und die gute Zusammenarbeit. Hervorheben möchte ich besonders Herrn Dr.-Ing. Markus Koch, Herrn Dipl.-Ing. Uli Wenger und Herrn Dipl.-Ing. Bernd Müller.

Ich möchte mich auch bei allen Studenten und wissenschaftlichen Hilfskräften, die mich bei der Bearbeitung meiner Projekte unterstützt haben bedanken, insbesondere bei Herrn cand.-Ing. Gert Stumpf, Herrn cand.-Ing. Kenan Halilovic, Herrn Dipl.-Ing. Frank Faltenbacher und Herrn Dipl.-Ing. Oliver Schmidt.

Mein herzlichster Dank gilt meiner Frau Monika und meinen Töchtern Anja und Verena, die durch ihre stete Motivation und Unterstützung den erfolgreichen Abschluß dieser Arbeit möglich gemacht haben.

Erlangen, Juni 2002 Matthias Wenk

Entwicklung eines konfigurierbaren Steuerungssystems für die flexible Sensorführung von Industrierobotern

- Inhaltsverzeichnis -

1 Einführung

Die zunehmende Bedeutung des Faktors Zeit kennzeichnet die technische Entwicklung in der Produktionstechnik [9]. Dies äußert sich in einer stetigen Verkürzung der Time-to-Market innovativer Produkte. Der gesamte Produktlebenszyklus würde sich ebenfalls spürbar verkürzen, wenn nicht geeignete Produktpflege (z.B. Designmodifikationen, neue Funktionen) und Marketingmaßnahmen erfolgreich dagegen gesetzt werden können. Dadurch steigt allerdings die Zahl der Varianten und es sinken entsprechend die Losgrößen pro gefertigter Variante [94]. Für die Unternehmen stellt sich damit vor allem die Herausforderung der wirtschaftlichen Fertigung auch bei kleinen und kleinsten Losgrößen [49].

Diese Situation zwingt Unternehmen in Hochlohnländern in zunehmenden Maße zur Rationalisierung, um im internationalen Wettbewerb bestehen zu können [116]. Im Bereich der Produktion ist neben technologischen und organisatorischen Verbesserungen eine Steigerung der Flexibilität der Produktionsanlagen und Automatisierungssysteme von entscheidender Bedeutung [102].

Roboter sind die geeigneten Werkzeuge, mit denen dieser Forderung entsprochen werden kann. Den steigenden Anforderungen hinsichtlich Positioniergenauigkeit, Steifigkeit und Bahntreue seitens der industriellen Anwender wurde in den letzten Jahren vor allem durch Verbesserung der konstruktiven Ausführung der Robotermechanik und Verwendung hochdynamischer, wartungsarmer Gelenkantriebe in Verbindung mit leistungsfähigen Robotersteuerungen Rechnung getragen [105].

Weitgehend unberücksichtigt bleibt dabei die Unterstützung von sensorischen Eigenschaften. Dieses Faktum resultiert nicht zuletzt aus dem Selbstverständnis von Roboterherstellern, die die Sensorik als reine Zuliefertechnik betrachten, die sich daher weitgehend an den Gegebenheiten der Roboter orientieren muß. Dies führte in der Vergangenheit lediglich zu suboptimalen Lösungen.

Die Erschließung neuer, zunehmend komplexerer Aufgabenfelder für Roboter setzt jedoch den verstärkten Einsatz hochflexibler Sensortechnik voraus. Nur durch den Sensoreinsatz sind die vielfältigen Fehlereinflüsse in einer roboterbasierten Bearbeitungszelle im erforderlichen Maße kompensierbar [34].

Trotz der Fortschritte in der Roboter- und Sensortechnik entwickelt sich der Einsatz von Sensorsystemen im Produktionsbereich nur zögerlich. Dem Zugewinn an Robustheit durch die Verwendung von aufwendiger und unflexibler Spanntechnik wird in vielen Bearbeitungsanwendungen der Vorzug vor der Erhöhung der Flexibilität und Bearbeitungsqualität mit sensorgestützten Verfahren gegeben. Die Gründe hierfür liegen in den hohen Aufwendungen, die die Integration und Verarbeitung von Sensordaten in Robotersteuerungen heute noch erfordern [36].

Fehlende Schnittstellenstandards in physikalischer und logischer Hinsicht erschweren die Integration von Sensordaten. Auf der Verarbeitungsseite verlieren Leistungsbeschränkungen durch die Hardwarekomponenten angesichts ständig steigender Mikroprozessorrechenleistung und Kapazität von Halbleiterspeichern zunehmend an Bedeutung [89]. Der Entwicklungsaufwand für die Verarbeitungssoftware stellt heute den zentralen Kostenfaktor dar [15]. Generell gilt für Software, daß sich nur durch die Wiederverwendung bestehender Lösungen längerfristig kostengünstige Systeme herstellen lassen. Aus Sicht der Anwender muß ein Softwaresystem darüberhinaus adaptierbar sein, um die geforderte Flexibilität bei hoher Variantenvielfalt erreichen zu können. Beide Forderungen - Wiederverwendbarkeit und Adaptierbarkeit - sind nur mit offenen Systemkonzepten umsetzbar [110].

Infolgedessen, besteht das Ziel dieser Arbeit darin, mit neuen Ansätzen aus dem Bereich der Komponententechnologie, ein offenes und an anwendungsspezifische Anforderungen anpaßbares Softwaresystem zur Integration und Verarbeitung von Sensordaten für die flexible Sensorführung von Industrierobotern zu entwickeln.

Um die Industrietauglichkeit der konzipierten Lösungen nachzuweisen, soll eine anspruchsvolle Nahtfolgeanwendung aus dem Bereich der Lasermaterialbearbeitung realisiert werden. Dabei soll die Offenheit des Systemkonzepts auch in der Gestaltung der Laufzeitumgebung Berücksichtigung finden.

Die Simulationstechnologie stellt heute ein Standardwerkzeug für Planungs-, Entwicklungs- und Optimierungsaufgaben im produktionstechnischen Umfeld dar [48]. Für die Entwicklung und Optimierung des Nahtfolgealgorithmus soll deshalb ein 3D-Kinematik-Simulationssystem eingesetzt werden. Damit ist es möglich, bereits in der Entwicklungsphase algorithmische Fehler zu erkennen und zu beseitigen. Durch die Berücksichtigung von dynamischen Systemeigenschaften innerhalb der Simulationsumgebung sollen die freien Parameter des Nahtfolgealgorithmus geeignet eingestellt werden.

Weitere Rationalisierungspotentiale bestehen darin, eine Produktionszelle nahtlos in die innerbetriebliche Informationsverarbeitung zu integrieren. Informationstransparenz ist heute die Voraussetzung für eine effiziente Produktionssteuerung. Die Entscheidungsträger müssen mit allen notwendigen Informationen versorgt werden, um fundierte Entscheidungen treffen zu können [95].

Information wird in Zukunft einer der wichtigsten Produktionsfaktoren. Aus diesem Grund sollen Konzepte entwickelt und umgesetzt werden, um die informationstechnische Integration der roboterbasierten Bearbeitungszelle zu ermöglichen.

2 Analyse der Fehlereinflußmöglichkeiten in Roboterzellen

Robotersysteme sind unverzichtbarer Bestandteil vieler Automatisierungslösungen in der verarbeitenden Industrie. Roboter nehmen dem Menschen körperlich anstrengende und gefährliche Arbeiten ab und erhöhen die Produktivität und Qualität der Prozesse und Produkte. Der größte strategische Vorteil liegt jedoch in der erreichbaren Flexibilität. Diese verschafft Robotern einen hohen Wiederverwendungswert, verringert die Time-to-Market und ermöglicht eine Produktion bis zu Losgröße eins.

Die Robotik gilt als Vorzeigebranche des deutschen Maschinenbaus. Seit Mitte der neunziger Jahre ist ein regelrechter Boom auf dem Robotermarkt beobachtbar. In den Jahren von 1996 bis 2000 wurden Wachstumsraten bei den Neuinstallationen von durchschnittlich 13% erreicht [17], während der allgemeine Maschinenbau nur moderate Zuwächse verzeichnen konnte. Im Jahr 2000 wurden 12800 Roboter in Deutschland neu installiert. Damit sind in Deutschland über hunderttausend Roboter im Einsatz.

Der größte Abnehmer von Robotern ist die Automobilindustrie und deren Zulieferer. Der Automobilanteil am Robotermarkt beträgt ca. 57% [83]. Aus diesem Grund dominiert bei den Anwendungsfeldern das Punktschweißen deutlich (Bild 1).

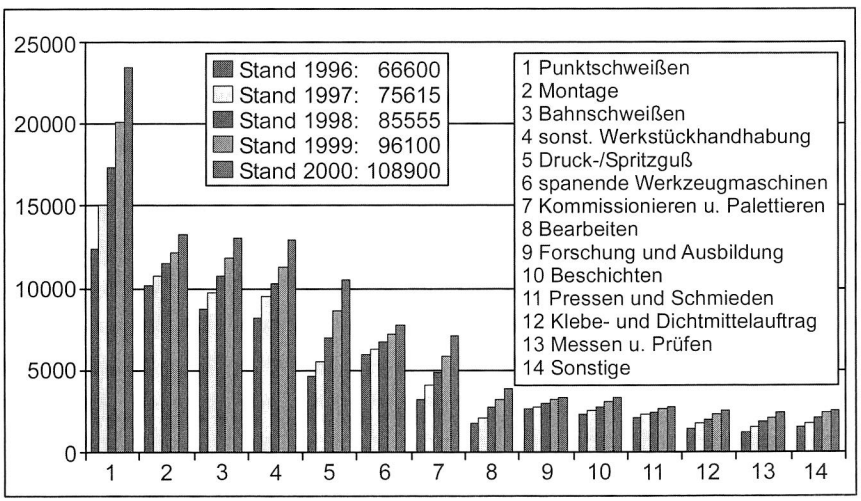


Bild 1: Robotereinsatzzahlen in Deutschland 1996-2000 (nach [17])

Dies könnte sich in den nächsten Jahren ändern, da der Trend im Automobilbau in Richtung Leichtbauweise geht. Der verstärkte Einsatz von Kunststoffen, Aluminium oder der Kombination unterschiedlicher Werkstoffe begünstigt mechanische Fügetechniken. Die Anwendungsbereiche Werkstückhandhabung und Kleben werden deshalb in Zukunft an Bedeutung gewinnen. Ein weiterer Konkurrent des Punktschweißens ist das Bahnschweißen. Moderne Verfahren, wie das Laserstrahlschweißen, werden aufgrund der erzielbaren Qualität auch im Automobilbereich, beim Fügen sichtbarer Karosserieteile, immer häufiger eingesetzt.

In der Spitzengruppe der Einsatzzahlen etabliert sich traditionell die Montage. Infolge des starken Preisverfalls für Roboter in den letzten Jahren werden in automatisierten Montagesystemen statt unflexibler Bewegungsmodule zunehmend Industrieroboter eingesetzt. Der Trend zu hybriden Montagekonzepten [33] fördert den Robotereinsatz zusätzlich. Anwendungen, die bisher allein durch Robotereinsatz technisch bzw. wirtschaftlich nicht automatisierbar waren, können mit der Einbeziehung des Menschen in den kritischen Montageschritten roboterbasiert automatisiert werden.

Zur Aufrechterhaltung der bisherigen Zuwachsraten dringt der Robotereinsatz in immer anspruchsvollere Anwendungsfelder vor. Als limitierende Faktoren gelten jedoch die Positionier- und Bahngenauigkeit eines Roboters. Beispielsweise gestaltet sich der Fügeprozeß beim Punktschweißen in Bezug auf Positionierungenauigkeiten der Fügestelle vergleichsweise unkritisch. Im Gegensatz dazu stellt das Bahnschweißen mit dem Laser sehr hohe Anforderungen an die Bahngenauigkeit des Roboters. Derartige Prozesse fordern das genaue Einhalten definierter geometrischer Beziehungen zwischen Werkzeug und Werkstück. Ungenauigkeiten in diesem Bereich wirken sich direkt auf die Prozeßqualität aus.

Bei der Materialbearbeitung mit Robotern treten aufgrund verschiedener Einflüsse Abweichungen zwischen der programmierten Bahn, die das Werkzeug im Raum abfährt, und der zu bearbeitenden Werkstückkontur auf. Diese Einflüsse sollen im folgenden genauer analysiert werden.

2.1 Roboter- und Werkzeugungenauigkeiten

Im Vergleich zu einer Werkzeugmaschine besitzen Industrieroboter eine wesentlich geringere Genauigkeit. Dies liegt vor allem in dem unterschiedlichen kinematischen Aufbau begründet. Bei einer Werkzeugmaschine werden das Werkstück und die Antriebsstränge in einem gemeinsamen, sehr steifen Maschinenbett gehalten. Im Gegensatz dazu ist ein Roboter als kinematische Kette aus bis zu sechs Achsen aufgebaut. Vor allem Knickarmroboter weisen damit eine wesentlich geringere Steifigkeit auf. Die Bewegung des Werkzeugs in einer Werkzeugmaschine erfolgt überwiegend auf Linearführungen, die mit hochgenauen Meßsystemen, wie z.B. Glasmaßstäben, ausgerüstet sind. Ein Roboter hingegen verfügt vornehmlich über

rotatorische Achsen. Die Stellung einer rotatorischen Achse wird über vergleichsweise ungenauere Winkelmeßsysteme erfaßt als dies bei einer linearen Achse möglich ist. Als weitere Einflußfaktoren auf die Genauigkeit sind der wesentlich größere Arbeitsraum eines Roboters und die höheren Bahngeschwindigkeiten zu nennen.

Unter der Genauigkeit eines Industrieroboters ist im wesentlichen die Wiederholgenauigkeit beim Positionieren und/oder Orientieren bzw. beim Nachfahren einer Bahn zu verstehen [96]. Diese Genauigkeitskenngrößen sind in nationalen und internationalen Richtlinien, wie der VDI-Richtlinie 2861 [96], der ISO 9283 [66] oder der RIA 15.05 [2] definiert. Da die Güte dieser Kenngrößen sich unmittelbar auf die Fertigungsqualität auswirkt, sollen diese im folgenden näher erläutert werden.

2.1.1 Fehlereinflüsse auf die Wiederholgenauigkeit

Die Wiederholgenauigkeit gibt an, mit welcher maximalen Abweichung ein numerisch im Raum vorgegebener Punkt, bei stets gleicher Anfahrrichtung und Geschwindigkeit, von der Maschinenkinematik angefahren werden kann. Dabei wird nicht nur die kartesische Positionsabweichung zwischen der programmierten Sollposition und der erreichten Istposition einbezogen. Die Wiederholgenauigkeit umfaßt ebenso die Orientierungsabweichungen. In der Praxis haben die Orientierungsabweichungen jedoch nur einen geringen Stellenwert und werden deshalb in den Datenblättern häufig nicht angegeben.

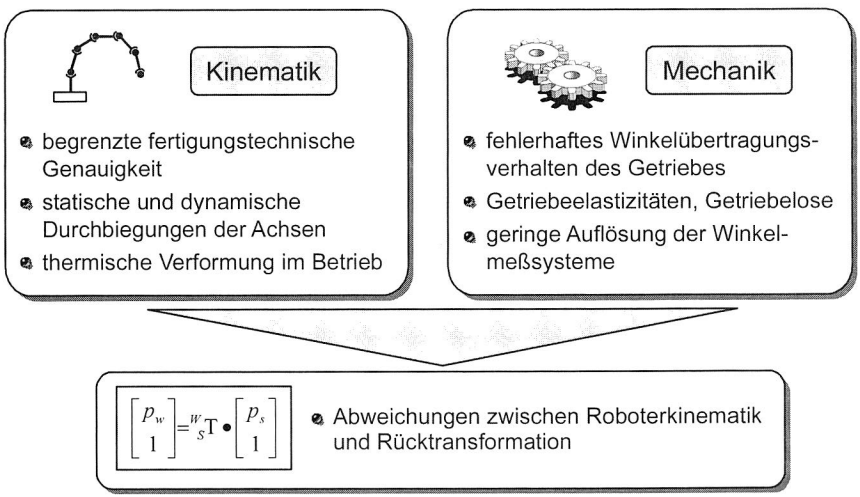


Bild 2: Kinematische und mechanische Einflüsse auf die Wiederholgenauigkeit

Die Wiederholgenauigkeit wird von Fehlereinflüssen aus der Kinematik und der Mechanik bestimmt (Bild 2). Eine begrenzte fertigungstechnische Genauigkeit führt zu Abweichungen in den Roboterarmlängen, Lage- und Orientierungsfehlern der Achsmittelpunkte und zu Nullagefehlern der Achsen. Statische und dynamische Durchbiegungen der Achsen, infolge von Störkräften, wie Gravitationskräfte und Reaktionskräfte resultieren in Verformungen des kinematischen Aufbaus. Eine Verformung der Achskörper als Folge von Temperaturänderungen ist vor allem bei hochpräzisen Anwendungen von Bedeutung [53].

Zusätzlich zu den Kinematikfehlern treten mechanische Fehler im Antriebsstrang auf. Zu nennen sind hier ein fehlerhaftes Winkelübertragungsverhalten des Getriebes, Achswinkelfehler durch Getriebeelastizitäten, Hystereseeinflüsse durch Getriebelose und die relative geringe Auflösung der Winkelmeßsysteme in den Achsen [99].

Die Summe der kinematischen und mechanischen Fehlereinflüsse führen zu Abweichungen zwischen der realen Roboterkinematik und dem der Rücktransformation zugrunde liegenden Kinematikmodell. Bedingt durch diese Modellabweichungen werden programmierte Positionen und Orientierungen mit einem maximalen Fehler in Höhe der Wiederholgenauigkeit angefahren. Das Ausmaß, mit dem die beschriebenen Fehlereinflüsse auf die Posefehler wirken, hängt vor allem von den aktuellen Achsstellungen und der am Roboterflansch angebrachten Nutzlast ab. Die in heutigen Robotersteuerungen hinterlegten Kinematikmodelle sind jedoch starre Modelle, die nicht an den momentanen Kinematikstatus angepaßt werden können. Adaptive Kinematikmodelle wären in der Lage die Posefehler deutlich zu reduzieren. Untersuchungen zur Kompensation von Temperatureinflüssen belegen dies [45, 53].

2.1.2 Fehlereinflüsse auf die Bahngenauigkeit

Die Bahngenauigkeit gibt die Größe der Abweichungen an, die bei Bewegungen entlang von Raumkurven entstehen. Auch hier führen kinematische und mechanische Fehler zu Bahnabweichungen [56]. Bei geringen Bahngeschwindigkeiten dominiert der Einfluß von Getriebefehlern [90]. Bei höheren Bahngeschwindigkeiten führen vor allem die dynamischen Eigenschaften der Achsregelkreise zu zusätzlichen Fehlereinflüssen. In heutigen Robotersteuerungen werden überwiegend P-Lageregler eingesetzt. Diese führen zu Schleppabständen in den einzelnen Achsen. Die Schleppabstände werden über die Roboterkinematik nichtlinear übertragen und führen zu Bahnabweichungen, die mit wachsender Geschwindigkeit zunehmen [10]. Leistungsfähigere Antriebe und fortschrittliche Regelungskonzepte für die Achsregelung bieten Potential für die Verbesserung der Bahngenauigkeit [73].

2.1.3 Werkzeugbedingte Fehlereinflüsse

Die Roboterkinematik dient dazu, ein Werkzeug, das am Roboterflansch angebracht ist, im Arbeitsraum definiert zu bewegen. Der sog. Werkzeugbezugspunkt (TCP, engl. Tool Center Point) definiert hierbei den Eingriffspunkt des Werkzeugs. Dies kann z.B. die Spitze eines Bohrwerkzeugs, der Fokuspunkt eines Laserbearbeitungswerkzeugs oder der Greifermittelpunkt sein. Die programmierten Positionen und Orientierungen, sowie die programmierte Bahngeschwindigkeit, beziehen sich auf den TCP. Die exakte Vermessung des TCP ist deshalb von großer Bedeutung, vor allem im Hinblick auf die Programmierverfahren (Bild 3). Beim Teach-In-Verfahren wird dies vor allem bei der Umorientierung des Werkzeugs deutlich. Wurde der TCP nicht exakt vermessen, so wird das Werkzeug nicht in Bezug auf den Eingriffspunkt umorientiert. Das erfordert eine Nachkorrektur in der kartesischen Position und damit eine Verlängerung der Programmierzeiten. Noch gravierender ist der Einfluß bei der Offline-Programmierung. Weicht der Werkzeugbezugspunkt im Rechnermodell von dem realen TCP ab, so erfordert dies ein aufwendiges Nachteachen der programmierten Posen.

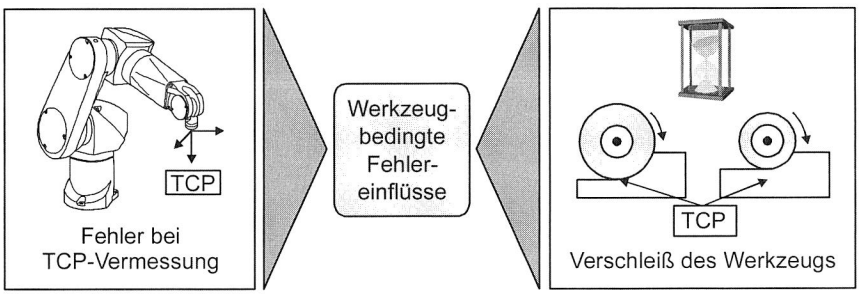


Bild 3: Ursachen für werkzeugbedingte Fehlereinflüsse

Bei berührendem Werkzeugeingriff führt der Verschleiß des Werkzeugs zu Abweichungen zwischen dem realen und dem vermessenen TCP [104]. Ebenso können Kollisionen des Werkzeugs mit der Zellenperipherie zu Abweichungen führen. Um die Bearbeitungsqualität sicher zu stellen, muß der TCP in zeitlichen Abständen nachgemessen werden.

2.2 Werkstückbedingte Fehlereinflüsse

Beim Ausführen eines Roboterprogramms wird davon ausgegangen, daß es keine Abweichungen zwischen dem realen Werkstück und dem bei der Programmerstellung verwendeten Werkstück bzw. dem rechnerintern vorliegendem Werkstückmodell bestehen. In der Praxis treten jedoch werkstückbedingte Fehlereinflüsse auf, die diese

idealisierte Annahme wiederlegen. Die auftretenden Abweichungen können erheblich größer werden als die durch die Roboterungenauigkeiten bedingten Fehler. In bestimmten Anwendungsfällen können sie aus fertigungstechnischer Sicht toleriert werden. Erhöhte Genauigkeitsanforderungen, z.B. bei der Lasermaterialbearbeitung, zwingen dagegen zum Einhalten enger Toleranzen. Werkstückbedingte Fehlereinflüsse sind auf das Werkstück selbst, aber auch auf die Werkstückzuführung zurückzuführen.

2.2.1 Werkstücktoleranzen durch vorgeschaltete Fertigungsschritte

Bevor ein Werkstück in einer Roboterzelle bearbeitet wird, hat es bereits eine Reihe vorgeschalteter Fertigungsschritte durchlaufen. Hierbei handelt es sich häufig um Umform- oder Trennprozesse, wie dem Tiefziehen oder Stanzen. Bei diesen Prozessen unterliegen die zum Einsatz kommenden Werkzeuge einem hohen Verschleiß. Aber auch chargenabhängige Werkstoffeigenschaften, wie beispielsweise das Rückfederungsverhalten nach dem Umformen, führt zu Fertigungstoleranzen. Diese können sich bei mehreren aufeinanderfolgenden Fertigungsschritten aufaddieren und zu relevanten Werkstückabweichungen führen [115].

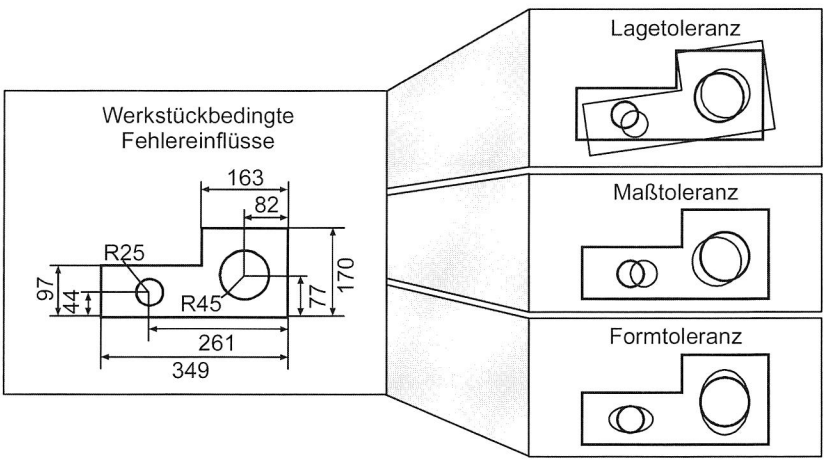


Bild 4: Ursachen für werkstückbedingte Fehlereinflüsse (nach [121])

Man unterscheidet Maßabweichungen, Lageabweichungen und Formabweichungen (Bild 4) [59]. Bei Maßabweichungen ist ein einzelnes Maß des Werkstücks fehlerbehaftet. So kann beispielsweise der Durchmesser einer Bohrung, infolge von Bohrerverschleiß, fehlerhaft sein. Bei Lageabweichungen ist die gesamte Kontur des zu bearbeiteten Kontursegments durch eine Translation oder Rotation im Raum

verschoben. Lageabweichungen können sich auf das gesamte Werkstück oder nur auf Teilsegmente beziehen. Die programmierte Bearbeitungsbahn kann durch Verschiebung wieder mit der Werkstückkontur zur Deckung gebracht werden [92]. Dies erfolgt steuerungsintern durch die Anwendung von Transformationsoperationen auf die programmierten Raumpunkte. Unter Formabweichungen werden Abweichungen verstanden, bei denen die ursprünglich vorgesehene Gestalt einer Kontur, z.B. einer Geraden oder eines Kreissegments, nicht eingehalten wird.

2.2.2 Werkstücktoleranzen durch Prozeßeinflüsse

Neben den Fertigungstoleranzen in den vorgeschalteten Fertigungsschritten, können bei thermischen Fügeprozessen, während des Bearbeitungsprozesses, zusätzliche Abweichungen in der Werkstückgeometrie auftreten [92]. Vor allem bei Schweißprozessen im Dünnblechbereich stellt der thermische Verzug einen wesentlichen Fehlereinflußfaktor dar. Das Aufbringen hoher Spannkräfte durch aufwendige Spannvorrichtungen kann den Fehlereinfluß begrenzen. Als problematisch kann sich allerdings die dadurch eingeschränkte Zugänglichkeitssituation erweisen.

Bei kraftschlüssigen Fertigungsverfahren, wie dem Stanznieten oder Clinchen, können die auftretenden Prozeßkräfte zu einer unbestimmten Deformation des Werkstücks führen [59]. Verlassen relevante Werkstückmerkmale, wie z.B. Bohrlochpositionen, ihre Toleranzbereiche, so kann die Fertigungsqualität nachfolgender Prozesse nicht mehr sichergestellt werden.

2.2.3 Fehlereinflußfaktor Spannvorrichtung

Die exakte Positionierung des Werkstücks zum Industrieroboter muß durch entsprechende Spannvorrichtungen sichergestellt werden. Außerdem muß ein bei thermischen Fertigungsverfahren auftretender Verzug des Werkstücks unterbunden werden. Die einzuhaltenden Fertigungstoleranzen bestimmen dabei den Aufwand für die einzusetzende Spanntechnik.

Um eine gleichbleibende Aufspannqualität sicherzustellen, müssen Abweichungen der Aufspannung vermieden werden. Abweichungen der Aufspannung können entweder durch Spannfehler oder durch die Spannmittel selbst hervorgerufen werden. Spannfehler entstehen durch mangelhafte Positionierung des Werkstücks vor dem Betätigen der Spannelemente sowie durch Verschmutzungen zwischen Werkstück und Spannmittel [99]. Abweichungen der Spannmittel lassen sich in Form- und Lageabweichungen unterteilen. Formabweichungen werden durch zu große Nachgiebigkeit, Verzug der Spannmittel oder durch Temperatureinflüsse verursacht und lassen sich durch richtige Auslegung der Spannmittel vermeiden. Lageabweichungen werden besonders durch Kollisionen hervorgerufen. Bei der Repositionierung der Spannmittel können anschließend Lageabweichungen auftreten.

2.3 Auftreten von Programmierfehlern

Für die Programmierung von Industrierobotern werden im industriellen Umfeld zwei Verfahren eingesetzt. Dies sind das Teach-In-Verfahren und die Offline-Programmierung. Beide Verfahren beinhalten spezifische Fehlereinflußpotentiale. Die Folge sind das Auftreten von Programmierfehlern, die bei der Programmausführung zu Bearbeitungsfehlern und damit zu Qualitätsverlusten führen (Bild 5).

2.3.1 Fehlereinflüsse bei der Online-Programmierung

Beim Teach-In-Verfahren wird der Roboter als Programmierhilfsmittel verwendet. Die erforderlichen Bewegungsschritte des Roboters werden vom Programmierer durch manuelles Verfahren der Achsen festgelegt. Die kartesische Position des Werkzeugbezugspunktes und die Orientierung des Werkzeugs in Bezug auf das Roboterkoordinatensystem werden direkt in das Roboterprogramm übernommen. Die zeitaufwendige Programmerstellung und die Belegung des Roboters während der Programmierung sind die wesentlichen Nachteile dieses Programmierverfahrens. In der Großserienfertigung ist dies noch zu tolerieren, da die Programmierung in diesem Fall als Nebenzeit der Fertigung nur einen Bruchteil der Hauptzeit ausmacht. Bei der Fertigung kleiner Losgrößen oder im Prototypenbau stellt die Programmierung mittels Teach-In allerdings einen erheblichen Kostenfaktor dar [99].

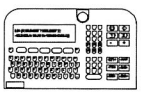
Online-Programmierung	Offline-Programmierung
	
• Programmiergenauigkeit abhängig von “Augenmaß” des Bedieners • Nichtberücksichtigung von dynamischen Abweichungen • Mangelnde Unterstützung durch Interpolationsverfahren	• Aufwendige Vermessung der Zellenkomponenten • Abweichungen zwischen Rechnermodell und realen Komponenten • Nachträgliche Programmanpassungen sind notwendig

Bild 5: Fehlereinflüsse bei der Online- und Offlineprogrammierung

Fehlereinflüsse bei der Teach-In-Programmierung ergeben sich dadurch, daß aus terminlichen Zwängen häufig ungenau programmiert wird, bzw. die erreichbare Genauigkeit bei Handprogrammierung prinzipiell von der Erfahrung und dem "Augenmaß" des Programmierers abhängt. Der stete Leistungszuwachs in der

Antriebs- und Steuerungstechnik ermöglicht in der Materialbearbeitung mit robotergeführten Werkzeugen immer höhere Bahngeschwindigkeiten. Bei starken Umorientierungen in der Bewegungsbahn und hohen Bahngeschwindigkeiten weicht die verfahrene Istbahn aufgrund dynamischer Restriktionen von der programmierten Sollbahn ab. Nur ein sehr erfahrener Programmierer kann durch geeignete Programmierung der Bewegungssätze im Roboterprogramm diesen Bahnabweichungen entgegenwirken. Neben der Programmierung der anzufahrenden Positionen, ist die Programmierung der Orientierung des Werkzeugs zum Werkstück von ebenso großer Bedeutung. Vor allem bei Bearbeitungstechnologien, die die Zuführung von Zusatzwerkstoffen erfordern, wie z.B. das Löten oder Kleben, wird dies deutlich. Während die Einhaltung von Abständen durch das Unter- bzw. Anlegen von Meßnormalen relativ unproblematisch sichergestellt werden kann, ist die Messung der drei kartesischen Orientierungswinkel zu jedem geteachten Punkt sehr aufwendig. Deshalb werden die Orientierungen häufig mit "Augenmaß" eingestellt.

Weiterhin ist die Auswahl an Interpolationsverfahren bei heutigen Robotersteuerungen sehr begrenzt. Die Programmiersprachen von Industrierobotern bieten lediglich Geraden und Kreisbögen zur Generierung von interpolierten Bewegungsbahnen. Komplexe Bahnverläufe können deshalb nur ungenau angenähert werden. Leistungsfähigere Verfahren, wie z.B. die Verwendung von Splines, die bei NC-Steuerungen schon lange zur Verfügung stehen, konnten sich in der Roboterprogrammierung noch nicht durchsetzen.

2.3.2 Fehlereinflüsse bei der Offline-Programmierung

Bei der Offline-Programmierung handelt es sich um ein indirektes Verfahren. Die Programmierung findet nicht am realen Roboter, sondern auf einem Rechner statt. Während der Programmerstellung kann in der Roboterzelle weiter produziert werden. Dieser wirtschaftliche Aspekt stellt einen der wesentlichen Vorteile der Offline-Programmierung dar. Die Erstellung des Roboterprogramms am Rechner erfordert ein Offline-Programmiersystem [4]. Voraussetzung ist die exakte Abbildung der Roboterkinematik, des Werkstücks und der Fertigungsumgebung in den rechnerinternen Modellen des Programmiersystems. Die Programmierung kann wie beim Teach-In-Verfahren benutzergeführt erfolgen. Dazu wird innerhalb der grafischen Bedienerschnittstelle der virtuelle Roboter zu den anzufahrenden Positionen bewegt. Offline-Programmiersysteme der neuesten Generation ermöglichen eine weitgehend automatische Generierung des Roboterprogramms [4,99]. Dabei wird das Roboterprogramm in Bezug auf Kollisionsfreiheit, zeitoptimale Bewegungsabläufe und prozeßtechnische Anforderungen optimiert.

Fehlereinflüsse ergeben sich durch die nicht vermeidbaren Abweichungen zwischen den rechnerinternen, idealisierten Modellen und den störungsbehaften realen Zellen-

komponenten [7]. Die erreichbare Genauigkeit, mit der die realen Zellenkomponenten in dem Rechnermodell des Offline-Programmiersystems abgebildet werden, hängt von dem Aufwand ab, den man der Vermessung der Zellenkomponenten widmet. Diesem wird durch wirtschaftliche Zwänge ein enger Rahmen gesetzt. Während der Vermessung der Zellenkomponenten kann die Roboterzelle nicht für Produktionszwecke genutzt werden. Ebenso erfordert jede Umbaumaßnahme eine Neuvermessung der betroffenen Komponenten [4].

In Bezug auf die robotergestützte Materialbearbeitung ist die Abbildung der exakten geometrischen Beziehungen zwischen dem vom Roboter geführten Werkzeug und dem zu bearbeitenden Werkstück in den Rechnermodellen von entscheidender Bedeutung. Aus Genauigkeitsgründen müssen deshalb offline programmierte Roboterprogramme in den meisten Fällen noch manuell nachgeteacht werden.

2.4 Schlußfolgerungen

Die diskutierten Fehlereinflußmöglichkeiten erschweren den Einsatz von Industrierobotern bei Anwendungen, die hohe Anforderungen an die Einhaltung exakter geometrischer Beziehungen zwischen Werkzeug und Werkstück stellen. In der industriellen Praxis wird versucht diese Problematik durch aufwendige Spanntechnik oder die Verwendung alternativer Kinematiken zu lösen. Als Beispiel sei hier der Einsatz von Portalanlagen bei Laseranwendungen zu nennen. Diese Lösungen sind jedoch mit erheblichen Investitionsaufwendungen verbunden, die sich nur bei hohen Stückzahlen amortisieren.

Eine wirtschaftliche Lösung stellt der Einsatz eines Regelungssystems dar. Dazu müssen die Roboterzellen mit leistungsfähigen Sensoren ausgerüstet werden. Die rasante Entwicklung der Mikroelektronik führte im Bereich der Sensortechnik zu einer deutlichen Leistungssteigerung bei sinkenden Preisen. Mit dem erreichten Leistungspotential ist es heute möglich die auftretenden Fehlereinflüsse mit der erforderlichen Genauigkeit zu erfassen. Dies ermöglicht den Einsatz von Standard-Industrierobotern auch in anspruchsvollen Anwendungsfeldern.

Die Kombination von Industrierobotern und Sensorsystemen führt zu Systemlösungen, die eine hohe Flexibilität mit der Sicherstellung der geforderten Bearbeitungsqualität verbinden. Folgerichtig besteht die Zielsetzung dieser Arbeit darin, die Integration von Sensorsystemen in Roboterzellen durch neue Ansätze weiter voranzutreiben.

3 Sensordatenintegration in Robotersteuerungen

Die Fehlereinflußfaktoren in roboterbasierten Produktionszellen sind vielfältig. In Kapitel 2 wurde diese Problematik ausführlich dargestellt. Ein Großteil der Fehlerquellen ist mit einem hohen bis sehr hohen Aufwand kompensierbar. So sind beispielsweise Roboter- und Programmierungenauigkeiten durch umfangreiche Kalibriermaßnahmen minimierbar, Werkstücktoleranzen können durch exaktes Vermessen jedes einzelnen Werkstückes erfaßt werden und Spannfehler können durch komplexe Spannvorrichtungen vermieden werden. Das technisch Machbare muß jedoch auch unter wirtschaftlichen Rahmenbedingungen umsetzbar sein. Der hohe personelle und zeitliche Aufwand, der mit diesen Kompensationsmaßnahmen zwangsläufig verbunden ist, ist nur bei einigen wenigen Anwendungsfeldern, wie z.B. in der Mikromontage, wirtschaftlich vertretbar.

Eine wirtschaftliche Alternative stellt der Einsatz von Sensorsystemen dar. Sie tragen dazu bei, die Flexibilität eines Robotersystems zu erhöhen und die erforderliche Produktionsqualität zu gewährleisten.

3.1 Definitionen zum Begriff Sensor

Nach [14] kann ein Sensor folgendermaßen charakterisiert werden:

> Die Aufgabe eines Sensors besteht darin, eine zu messende physikalische, nichtelektrische Größe und ihre Änderungen in ein eindeutiges elektrisches Signal umzuwandeln.

Die Umwandlung einer physikalischen Größe in ein elektrisches Signal wird durch das Kernstück eines Sensors, dem Sensorelement bewerkstelligt [74]. Damit scheint die Funktionalität eines Sensors, nach der obigen Definition, erfüllt zu sein. Ein Sensor ist jedoch nicht auf des Sensorelement beschränkt. Die vom Sensorelement gelieferte elektrische Größe muß in der Regel noch aufbereitet werden. Die Signalaufbereitung umfaßt die Verstärkung der meist sehr geringen Amplitude der elektrischen Größe. Dies ist notwendig, um den Signal/Rauschabstand für die Übertragung des Signals zu einer Verarbeitungseinheit zu verbessern. Der Signalverstärkung kann eine Filterung folgen, um Störeinflüsse, die im Meßsignal enthalten sein können, zu minimieren. Erst die Kombination aus Sensorelement und Aufbereitungselektronik stellt einen Sensor dar. Die moderne Mikroelektronik ermöglicht es durch Miniaturisierung der Aufbereitungselektronik diese in den Sensor zu integrieren. Man spricht dann von einem "integrierten Sensor" (Bild 6).

Das elektrische Signal des Sensors wird zu einer Verarbeitungseinheit übertragen, die die weitere Verarbeitung des Sensorsignals durchführt. Dies kann im einfachen Fall eine Linearisierung sein und bis zur Extraktion von relevanten Signalmerkmalen

reichen. Die Bildung der Fouriertransformierten aus dem Sensorsignal ist ein Beispiel für einen komplexen Verarbeitungsprozeß. Basiert die Verarbeitungseinheit auf einem Mikrocontroller, so kann durch dessen hohe Integrationsdichte bei zunehmender Miniaturisierung auch die Verarbeitungseinheit in den Sensor integriert werden. Daraus resultiert ein sog. "Intelligenter Sensor", der sogar über eine digitale Schnittstelle in Form eines Feldbusanschlusses verfügen kann. Komplexe Anwendungen, wie sie beispielsweise eine Bildverarbeitung darstellt, erfordert mehr Rechenleistung von der Verarbeitungseinheit. In diesem Fall kommt ein vom Sensor getrenntes Rechnersystem auf Basis eines Mikroprozessors als Verarbeitungseinheit zur Anwendung. Die Verbindung eines der drei vorgestellten Sensortypen mit einem Rechnersystem zur Verarbeitung der Sensorsignale bezeichnet man als "Sensorsystem".

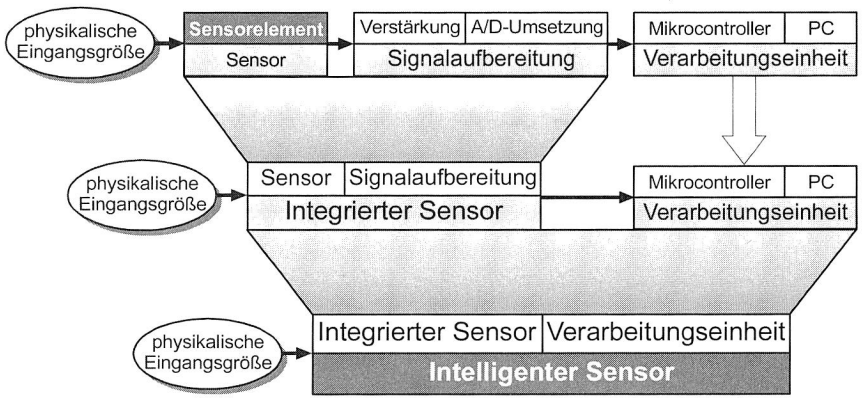


Bild 6: Darstellung der Integrationsstufen von Sensoren

Sensoren nehmen in der industriellen Robotertechnik eine Schlüsselrolle zur Erschließung komplexer Bearbeitungsaufgaben ein. Sie dienen zur Erfassung von Prozeß- oder Zustandsdaten bzw. zur Erfassung physikalischer Eigenschaften von Werkstücken, mit oder an denen fertigungstechnische Operationen durchgeführt werden. Sie wurden und werden entwickelt, um dem Roboter Flexibilität und Anpassungsvermögen zu verleihen.

3.2 Klassifikation von Sensoren für Roboteranwendungen

Eine Klassifikation von Sensoren kann nach unterschiedlichen Kriterien erfolgen. Häufig wird eine Einteilung nach Meßprinzip oder Meßgröße gewählt. Ein derartiges Vorgehen berücksichtigt jedoch keine roboterspezifischen Randbedingungen. In dieser Arbeit soll deshalb die Art der steuerungstechnischen Verarbeitung der Sensormeßwerte in der Robotersteuerung als Klassifikationskriterium herangezogen werden.

Generell können zwei unterschiedliche Verarbeitungsweisen unterschieden werden. Bei der ersten Verarbeitungsart werden die Sensordaten vor der Ausführung einer Bearbeitungsaufgabe verarbeitet (Offline-Verarbeitung). Im Gegensatz dazu werden die Sensordaten bei der zweiten Verarbeitungsart während der Ausführung einer Bearbeitungsaufgabe verarbeitet (Online-Verarbeitung). Die jeweils eingesetzten Sensoren unterscheiden sich je nach Verarbeitungsart deutlich. Die Verarbeitungsart ist damit ein sinnvolles Klassifikationskriterium.

3.2.1 Sensoren für die Offline-Verarbeitung

Die Aufgabe von Sensoren für die Offline-Verarbeitung besteht darin, die Position bzw. Orientierung von Werkstückmerkmalen zu bestimmen, die für die Bearbeitungsaufgabe relevant sind. Die gewonnen Sensordaten werden zur Modifikation eines bereits vorhandenen Roboterprogramms verwendet. Dabei werden einzelne Bewegungsbefehle an die aktuellen Werkstückgegebenheiten angepaßt. Ein typischer Anwendungsfall ist beispielsweise die Modifikation eines Positionierbefehls zum Greifen eines ungeordnet bereitgestellten Werkstücks. Andererseits können mehrere zusammenhängende Bewegungsbefehle eines Unterprogramms durch Translations- und Rotationsoperationen angepaßt werden. Dies ist beispielsweise notwendig, wenn zwei Werkstücke durch mehrere Schraubverbindungen gefügt werden sollen, die Fügepartner jedoch durch Spannfehler falsch positioniert sind. In diesem Fall müssen sämtliche Roboterbefehle zur Positionierung des Schraubers angepaßt werden. Charakteristisch für diese Anwendungsfelder ist, daß die Vermessung des Werkstücks vor der jeweiligen Bearbeitung stattfindet.

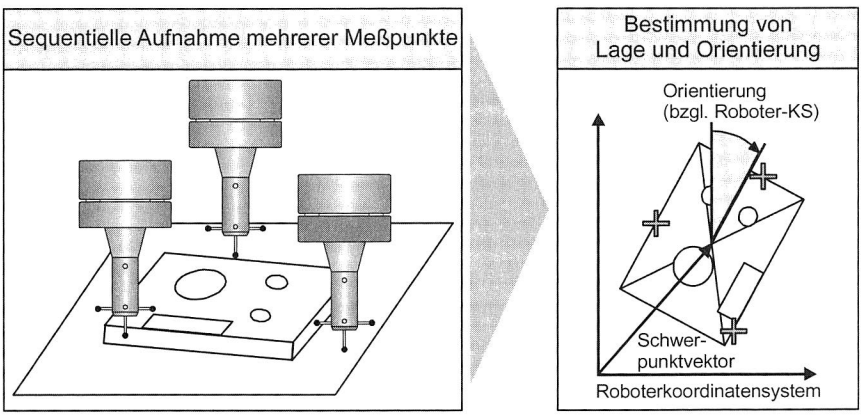


Bild 7: Bestimmung von Lage und Orientierung eines Werkstücks aus mehreren Messungen mit einem taktilen Sensor

Zur Lösung der damit zusammenhängenden Meßaufgaben stehen eine Vielzahl von Sensoren zur Verfügung, die sich in den zugrundeliegenden Meßprinzipien unterscheiden. Das Spektrum reicht von einfachen binären Sensoren, bis hin zu komplexen Sensorsystemen, die mehrdimensional aufbereitete Meßgrößen bereitstellen. Vor allem aus wirtschaftlichen Überlegungen werden in der industriellen Praxis zwei Sensortypen bevorzugt eingesetzt: Taster und Bildverarbeitungssysteme.

Ein Taster besteht im wesentlichen aus einem Mikroschalter, der betätigt wird, sobald der daran angebrachte Meßtaster mit einem Meßobjekt in Berührung kommt. Dieser Sensor zeichnet sich durch einen äußerst geringen Preis und eine hohe Robustheit aus. Zur Durchführung einer Messung muß der Roboter den Sensor zum Meßobjekt bewegen, bis eine Berührung stattfindet (Bild 7). Aus den kartesischen Positionsdaten der Roboterhand kann die räumliche Position des Berührpunktes zwischen dem Sensor und dem Meßobjekt bestimmt werden.

Den geringen Kosten des Sensors steht jedoch ein hoher Meßaufwand gegenüber. Sind zur Bestimmung der Positionen relevanter Werkstückmerkmale viele einzelne sensorische Positionsbestimmungen notwendig, so erfordert dies, bei Verwendung eines taktilen Sensors, eine Vielzahl von Roboterbewegungen. Dies führt zu einer Verlängerung der Taktzeit, die die Wirtschaftlichkeit der jeweiligen Roboteranwendung, trotz geringer Sensorkosten, nicht erfüllt. Dies ist vor allem dann der Fall, wenn die eigentliche Bearbeitungszeit für die Durchführung der Roboteranwendung in einem ungünstigen Verhältnis zum Zeitaufwand für die Messung steht. Zusammenfassend gilt, daß ein Taster dann wirtschaftlich sinnvoll einsetzbar ist, wenn pro Werkstück nur wenige Messungen durchzuführen sind.

Erfordert die Roboteranwendung, aufgrund einer kurzen Bearbeitungszeit, einen minimalen Zeitaufwand für die Bestimmung der relevanten Werkstückmerkmale, so bietet sich der Einsatz eines Bildverarbeitungssystems an. Unter der industriellen Bildverarbeitung ist die berührungslose Erfassung, visuelle Darstellung und automatische Auswertung bildhafter Szenen unter industriellen Umgebungsbedingungen (z.B. Staub, Wärme, Lichtschwankungen, Echtzeitanforderungen) zu verstehen.

Ein Bildverarbeitungssystem ist wesentlich komplexer aufgebaut, als ein taktiler Sensor. Es besteht aus den Komponenten Beleuchtung, Optik, Kamera und Bildverarbeitungsrechner mit Softwarepaket. Den eigentlichen Bildverarbeitungssensor bildet die Kamera, mit der zugehörigen Optik (Bild 8). Zur Verbesserung der Bildqualität ist auf eine ausreichende Beleuchtung besonderer Wert zu legen. Das Sensorelement ist ein Halbleiterbaustein, der linear oder matrixartig angeordnete, lichtempfindliche Bereiche aufweist, die Lichtintensitäten in elektrische Ströme umwandeln. Je nach verwendeter Technologie handelt es sich dabei um einen CCD-Chip (engl. Charged Coupled Device) oder einen CMOS-Chip (engl. Complementary Metal Oxide Semiconductor). Das von der Kameraelektronik aufbereitete Videosignal kann erst nach

erfolgter Digitalisierung in einem Rechnersystem weiterverarbeitet werden. Die dazu notwendige Hardwarekomponente bezeichnet man als Framegrabber. Die Auswertung der Bilddaten erfolgt durch spezielle Bildverarbeitungsalgorithmen.

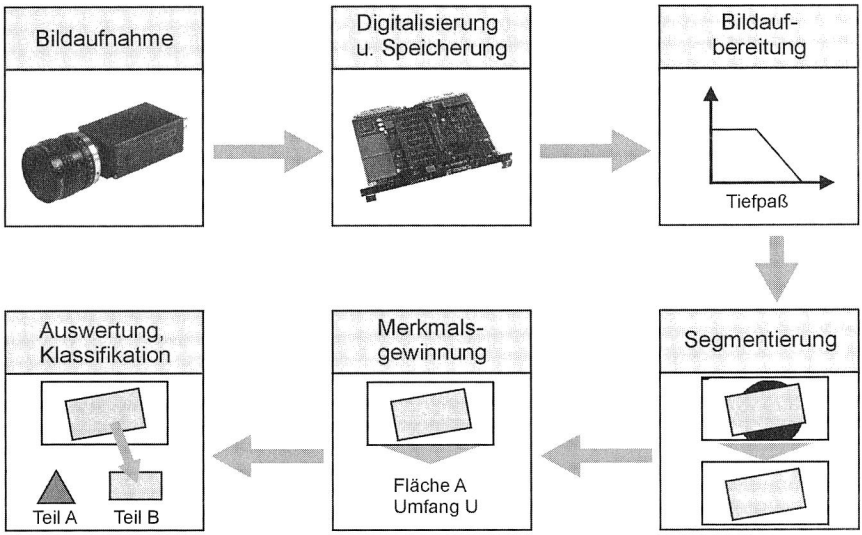


Bild 8: Ablauf einer typischen Bildverarbeitungsaufgabe

Die Bilddatenauswertung erfolgt in mehreren Schritten (Bild 8). Durch den Einsatz von Filterroutinen wird das Bild einer Szene aufbereitet und für die späteren Auswerteschritte optimiert. Bei der daran anschließenden Segmentierung werden die interessierenden Details der Szene aus dem Hintergrund extrahiert (z.B. durch Binärisierung). An Hand der segmentierten Szene werden die Merkmale für die spätere Auswertung gewonnen und an die Klassifikation weiter gegeben. Die Klassifikation bewertet die gefundenen Merkmale und trifft dann Entscheidungen wie z.B. Gutteil, Ausschußteil, Teil B wurde erkannt. Aufgrund einer derartigen Entscheidung kann letztlich der Ablauf bzw. die Bewegungsbefehle des Roboterprogramms beeinflußt werden.

Im Vergleich zu einem taktilen Sensor ist ein Bildverarbeitungssystem mit hohen Investitions- und Inbetriebnahmekosten verbunden. Andererseits besteht der Vorteil eines Bildverarbeitungssystems darin, das Abbild einer Szene mit einer Bildaufnahme zu erfassen und daraus die relevanten Werkstückmerkmale zu bestimmen. Die Taktzeit für die Durchführung einer Messung kann damit deutlich reduziert werden. Allein der Taktzeitgewinn rechtfertigt häufig die hohen Kosten. Aufgrund der gestiegenen Rechenleistung moderner Prozessoren, bei gleichzeitig starkem Preisverfall,

setzen sich Bildverarbeitungssysteme immer stärker durch und verdrängen andere optische Sensoren, wie den Triangulationssensor oder den Laserscanner.

3.2.2 Sensoren für die Online-Verarbeitung

Das zweite wichtige Anwendungsfeld für Sensoren in Roboteranwendungen ist die Adaption der Bewegungsbahn des Endeffektors. Der Endeffektor bewirkt die Interaktion des Robotersystems mit der Umwelt. Endeffektor ist der Oberbegriff für Greifersysteme zur Handhabung und Manipulation von Objekten, für Werkzeuge zur Werkstückbearbeitung und für Meßmittel zur Ausführung von Prüfaufträgen.

Das in Kapitel 3.2.1 vorgestellte Anwendungsfeld führt zu einer Adaption der Zielposition eines oder mehrerer Bewegungssätze *vor* der Durchführung der eigentlichen Bearbeitungsaufgabe. Im Gegensatz dazu werden bei dem hier besprochenen Anwendungsfeld *während* der Bewegung des Effektors die Stützwerte der Effektorbahn bestimmt. Dies kann auf zwei Arten erfolgen. Die erste Möglichkeit besteht darin, die Stützwerte eines programmierten Bewegungssatzes in Abhängigkeit der Sensorsignale zu korrigieren. Alternativ kann auf eine Programmierung verzichtet werden und die Stützwerte werden direkt aus den Sensorsignalen generiert. Die Bahnplanung des Roboters wird dabei deaktiviert.

Bei der überwiegenden Anzahl von Roboteranwendungen, die eine Adaption der Bewegungsbahn erfordern, soll der Effektor entlang markanter Werkstückkonturen mit definierter Lage und Orientierung geführt werden. Dies können z.B. Stoßformen für Schweißaufgaben oder Grate an Gußteilen sein. Ein geeignetes Sensorsystem muß deshalb in der Lage sein, vorhandene Werkstückkonturen zu erfassen. Dies kann sowohl taktil, als auch berührungslos erfolgen. Eine kontinuierliche, sensordatengestütze Bewegungsbeeinflussung eines robotergestützten Bearbeitungswerkzeugs erfordert die Erfassung von bis zu sechs Korrekturdimensionen:

- der Abstand und die seitliche Abweichung des Werkzeugs vom Werkstück
- die Geschwindigkeitskomponente in Vorschubrichtung (als dritte translatorische Korrekturgröße) und
- die räumliche Orientierung.

Bei rotationssymmetrischem Werkzeug entfällt ein rotatorischer Freiheitsgrad.

Bei der Online-Verarbeitung von Sensordaten werden in der überwiegenden Zahl von industriellen Anwendungen vor allem zwei Sensorsysteme eingesetzt. Es handelt sich dabei um Kraft/Momenten-Sensoren und Konturfolgesensoren.

Fügevorgänge bei der Montage oder Bearbeitungsvorgänge, insbesondere Entgraten, erfordern eine kraftschlüssige Bewegung zwischen Effektor und Werkstück. Ungenaue Werkstückpositionen sowie Toleranzstreuungen der Werkstückgeometrien können

große Reaktionskräfte und -momente bewirken, die sowohl auf den Roboterarm, als auch auf das Werkstück wirken. Die Folge sind hohe Gelenkbelastungen des Roboters und im Extremfall Beschädigungen an Effektor oder Werkstück. Um dies zu vermeiden müssen definierte Wirkkräfte, die für den technischen Prozeß erforderlich sind, eingestellt und eingehalten werden. Ein Sensor, der diese Anforderungen sicherstellt, ist der Kraft/Momenten-Sensor.

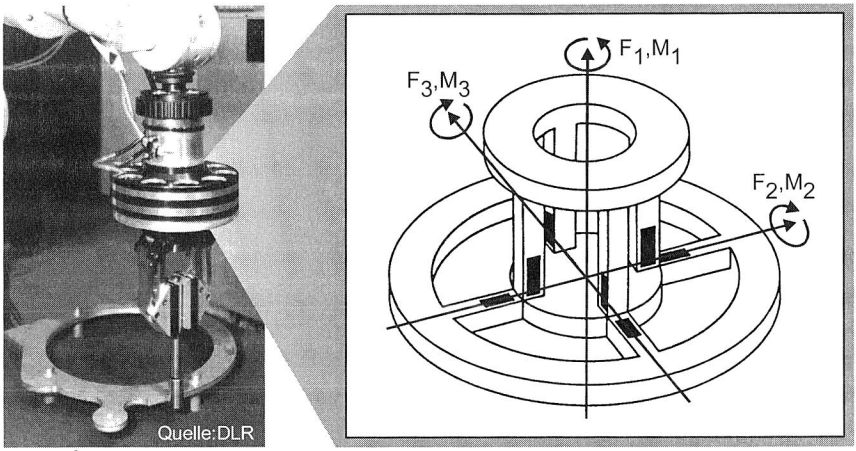


Bild 9: Schematischer Aufbau eines Kraft/Momenten-Sensors (nach [79])

Alle Verfahren zur Messung einer Kraft werden auf die Messung der auftretenden Verformung zurückgeführt. Die hierbei üblichen Methoden der Verformungsmessung beruhen darauf, daß sich bestimmte physikalische Eigenschaften bei kristallinen und elektrisch leitfähigen Körpern unter einer von außen eingeprägten Dehnung ändern [79]. Bei einfachen Aufgabenstellungen, bei denen der Ort des Krafteingriffs und die Wirkrichtung unverändert bleiben, reicht im allgemeinen die Erfassung von zwei Kräften und einem Drehmoment aus. Liegen jedoch Aufgabenstellungen vor, bei denen sich der Krafteingriffspunkt und auch die Richtung der Kraftwirkung ändern können, ist die Berücksichtigung aller Freiheitsgrade durch sechs Kraft- bzw. Momentenkomponenten erforderlich. Kraft/Momenten-Sensoren müssen so am Roboter adaptiert werden, daß der gesamte Kraftfluß über sie geleitet wird. Dies ist im allgemeinen die Lage zwischen dem Flansch der letzten Roboterachse und dem Effektor. Der mechanische Grundkörper eines Kraft/Momenten-Sensors auf Basis von Dehnungsmeßstreifen (DMS) hat meist den in Bild 9 dargestellten Aufbau. Werden Kräfte oder Momente über den oberen und unteren Ring eingeleitet, bewirken Sie je nach Wirkrichtung in den Speichen elastische Verformungen. Diese Verformungen

verursachen eine Widerstandsänderung der DMS. Die daraus resultierenden Spannungsänderungen werden nach einer Signalverstärkung über eine Matrix-Multiplikation in Kraft- und Momentenkomponenten umgerechnet. Aufgrund der umfangreichen Verarbeitung der ursprünglichen Meßgrößen handelt es sich hierbei um einen intelligenten Sensor, bzw. ein Sensorsystem.

Die von dem Kraft/Momenten-Sensorsystem bereitgestellten Kraft- und Momentenkomponenten dienen dazu, die Stützstellen der Bewegungstrajektorie des Roboterefektors zu adaptieren. Die Bewegungstrajektorie kann beispielsweise eine Fügebewegung in der Montage oder die Bearbeitungsbahn beim Entgraten eines Gußwerkstücks beschreiben. Ziel ist es, daß während der Bewegung die technologisch erforderlichen Wirkkräfte an dem Werkstück angreifen.

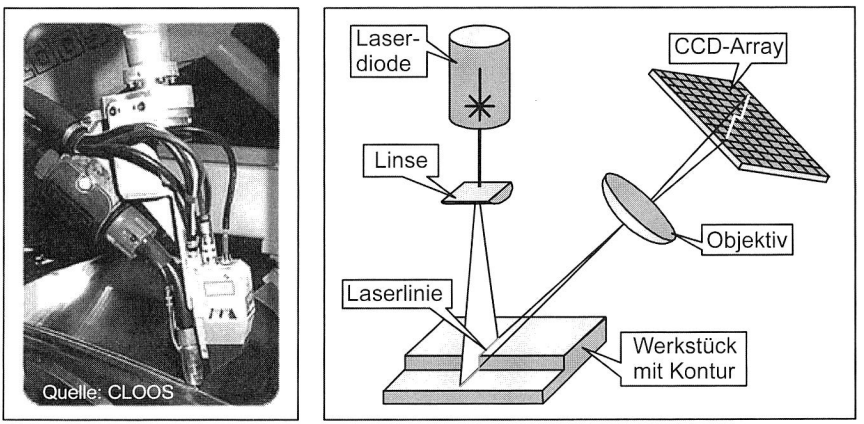


Bild 10: Anwendungsbeispiel und struktureller Aufbau eines Konturfolgesensors

Konturgeführte Fertigungsverfahren, wie das Bahnschweißen, Kleben oder Nahtdichten, erfordern keinen Kraftfluß zwischen Effektor und Werkstück. Die Aufgabenstellung besteht vielmehr darin, den Effektor entlang einer Werkstückkontur mit definiertem Abstand und mit definierter Geschwindigkeit und Orientierung zu führen. Die Erfassung der Werkstückkontur erfolgt berührungslos. Moderne Konturfolge-Sensoren setzen die industriell gut eingeführte Methode des Lichtschnittverfahrens ein (Bild 10). Dabei wird eine Lichtebene von einer Lichtquelle entsprechender Intensität und einem Kollimator, mit einer zylindrischen Linse zur Aufspreizung des Lichtstrahls, erzeugt. An allen den Punkten im Raum, an denen die Lichtebene auf die Oberfläche des Werkstücks (als zweite Ebene) trifft, entsteht als Projektion eine gerade Linie, die auf der Oberfläche des Objektes von einer Videokamera erfaßt werden kann. Die relative Position von projizierter Lichtebene und Kamera ist während des Betriebes fest

und die relative Lage wird durch Kalibrierung vor Beginn des eigentlichen Meßlaufes festgelegt. Da eine eindeutige Abbildung zwischen projizierter Lichtebene und Bildebene existiert, kann die Position eines Oberflächenpunktes des Objektes direkt aus der Bildposition eines Punktes der Laserlinie berechnet werden. Die Anordnung aus Bildpunkt, Laserlinienpunkt und Beleuchtung wird durch ein Dreieck beschrieben. Dieses Dreieck kann als Vermessungsdreieck aufgefaßt werden. Daher wird auch von Triangulation gesprochen.

Das Lichtschnittverfahren stellt sehr hohe Anforderungen an die Bildverarbeitung. Um eine ausreichend hohe Meßpunktdichte bei den gebräuchlichen Bearbeitungsgeschwindigkeiten zu erzielen, ist eine hohe Verarbeitungsgeschwindigkeit erforderlich. Moderne Signalprozessoren erfüllen die Anforderungen aus dem industriellen Einsatzfeld. Höhere Verarbeitungsleistung bei gleichzeitig sinkenden Preisen fördern den Einsatz dieser Sensorsysteme in der industriellen Anwendung.

3.3 Datenschnittstellen von Robotersteuerungen

Die Hauptaufgabe einer Robotersteuerung besteht im wesentlichen darin, die Achsen einer Roboterkinematik zu steuern. Dies geschieht in Abhängigkeit von den Bewegungsbefehlen in einem frei programmierten Roboterprogramm. Störeinflüsse aus der Arbeitsumgebung des Roboters werden dabei nicht berücksichtigt.

Die zunehmend komplexer gestalteten Handhabungs- und Bearbeitungsprozesse, die von Roboteranlagen durchgeführt werden, erfordern aus Gründen der Qualitätssicherung eine Kompensation der Störeinflüsse. Diese erfolgt durch Korrekturgrößen, die die programmtechnisch vorgegebene Bewegungsfolge beeinflussen. Geeignete Korrekturgrößen sind Positions-, Orientierungs- und Bahngeschwindigkeitsänderungen.

Die Korrekturgrößen werden aus den Meßwerten von Sensoren abgeleitet. Von einigen Ausnahmen abgesehen, müssen die Sensormeßwerte erst vorverarbeitet werden, um daraus die Korrekturgrößen zu bestimmen. Die Vorverarbeitung kann bei intelligenten Sensoren (z.B. Bildverarbeitungssystem) bereits im Sensorsystem erfolgen. Ansonsten ist zusätzliche Rechenleistung in Form eines Mikrocontrollers bzw. eines Mikroprozessors erforderlich. Eine Vorverarbeitung der Sensormeßwerte durch die Robotersteuerung ist aus Rechenzeitgründen in den meisten Anwendungsfällen nicht realisierbar [72], da es sich hierbei um komplexe Transformations- und Regelungsalgorithmen handelt.

Um die Korrekturgrößen bestimmen zu können, werden neben den Sensormeßwerten auch roboterinterne Daten benötigen. Dabei handelt es sich beispielsweise um die momentane kartesische Position und Orientierung des Endeffektors.

Zum Auslesen der benötigten Roboterdaten und zum Übertragen der Korrekturgrößen muß die Robotersteuerung über geeignete Schnittstellen verfügen. Unter einer Schnittstelle versteht man nach DIN 19240 [27] den Übergang zwischen einer elektronischen Steuerung und ihrer Peripherie. Man unterscheidet hierbei zwischen einfachen und intelligenten Schnittstellen. Eine einfache Schnittstelle dient der Übertragung von binären, digitalen und analogen Signalen nach DIN 19240 [27]. Intelligente Schnittstellen basieren auf seriellen bzw. parallelen Übertragungsverfahren. Die meist verwendeten intelligenten Schnittstellen sind die serielle Schnittstelle nach DIN 66259 [31] und die Feldbusschnittstellen (z.B. Profibus [29]). Erst wenige Robotersteuerungen verfügen über eine sog. Koppelspeicher-Schnittstelle in Form eines Dual-Ported-Memory (DPM). Bei dieser Schnittstelle treten vernachlässigbare Übertragungstotzeiten auf, da es sich um Speicherzugriffe handelt. Die DPM-Schnittstelle ermöglicht den Datenaustausch mit einer Einsteckkarte, die in die Robotersteuerung integriert wurde [35].

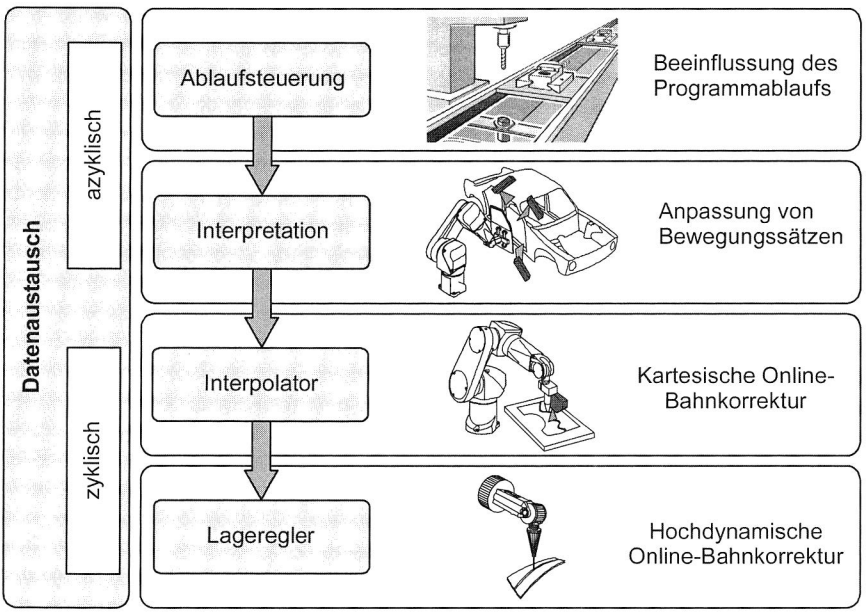


Bild 11: Verschiedene Ebenen des Sensoreingriffs in einer Robotersteuerung

Die Steuerungsstruktur einer Robotersteuerung ist hierarchisch aufgebaut. Die Schnittstellen einer modernen Robotersteuerung gehören unterschiedlichen Hierarchieebenen an (Bild 11). Auf der jeweiligen Ebene können systeminterne Roboter-

daten an eine externe Verarbeitungseinheit übertragen werden. Dies wird im folgenden nicht mehr explizit erwähnt. Der Schwerpunkt soll im weiteren auf den Integrationsmöglichkeiten von Korrekturgrößen liegen.

An der Spitze der Hierarchie steht die Ablaufsteuerung. Sie bestimmt die Abarbeitungsreihenfolge und den Abarbeitungszeitpunkt der Befehle des Roboterprogramms. Eine Schnittstelle auf dieser Ebene kann dazu genutzt werden den Programmablauf zu beeinflussen. Eingesetzt werden hier vor allem binäre Sensoren, wie z.B. induktive Näherungsschalter. Deshalb genügen als hardwaretechnische Schnittstelle die binären I/O-Kanäle der Robotersteuerung. Ein Anwendungsbeispiel ist die Detektion der Anwesenheit einer Werkstückpalette im Arbeitsraum des Roboters durch einen induktiven Näherungsschalter, der in das Materialflußsystem integriert ist. Daraufhin kann der Roboter mit einem geeigneten Endeffektor ein Werkstück entnehmen.

Die Ablaufsteuerung übergibt den nächsten abzuarbeitenden Befehl des Roboterprogramms an den Interpreter. Bewegungsbefehle gibt der Interpreter an die Präparation weiter. Diese berechnet aus den Bewegungsbefehlen den Bahnverlauf in parametrischer Form. Die Interpreterschnittstelle erlaubt es die Bewegungssätze vor der Weitergabe an die Präparation zu verändern. Änderungen werden durch eine translatorische bzw. rotatorische Transformation der Positions- und Orientierungswerte des aktuellen Bewegungssatzes bzw. einer Gruppe von Bewegungssätzen vorgenommen. Die Korrekturwerte werden meist auf Basis der Meßwerte eines taktilen Sensors oder eines Bildverarbeitungssystems generiert. Da die Korrekturwerte, für jeden Programmlauf, nur einmal verarbeitet werden, handelt es sich um eine azyklische Schnittstelle.

Aus dem von der Präparation berechneten Bahnverlauf und den vorgegebenen Beschleunigungs- und Geschwindigkeitsprofilen berechnet der Interpolator die anzufahrenden Stützstellen. Dies erfolgt zyklisch im Interpolationstakt. Moderne Robotersteuerungen realisieren Interpolationstaktzyklen zwischen 6 ms und 24 ms. Eine Interpolatorschnittstelle ermöglicht es die Stützstellen zu verändern und damit online die Bewegungsbahn des Roboters zu korrigieren. Durch die kurzen Interpolationszyklen sind heute wesentlich dynamischere Sensorregelungen realisierbar, als noch vor wenigen Jahren.

Wird eine noch höhere Regelungsdynamik benötigt, so kann die Schnittstelle der Lageregelung genutzt werden [34,121]. Die Lageregelung setzt die Führungsgrößen des Interpolators in die Bewegung des Roboters um. Damit dies möglichst exakt durchgeführt werden kann, liegen die Zykluszeiten von Lageregelkreisen heute im Bereich von 1 ms bis 2 ms. Da ein Lageregler auf axialer Ebene arbeitet, müssen auch die Korrekturgrößen axial vorliegen. Dies erfordert eine Transformation der meist kartesisch vorliegenden Korrekturgrößen.

3.4 Defizite von Sensordatenintegrationsmechanismen

Die rasante Entwicklung der Mikroelektronik in den vergangenen Jahren führte zu einer wesentlichen Leistungssteigerung bei Sensorsystemen [100]. Diese äußert sich vor allem in einer Verkürzung der Totzeit zwischen Meßwertaufnahme und Meßwertausgabe. Vor allem die hohen Totzeiten haben früher den Sensoreinsatz erschwert. Auf Seiten der Robotersteuerungen konnten in den letzten Jahren die Zykluszeiten des Interpolationstaktes weiter reduziert werden. Damit ist es für die meisten Anwendungen nicht mehr erforderlich die schwer beherrschbare Lagereglerschnittstelle zu nutzen. Weiterhin ist ein deutlicher Trend zu offenen Robotersteuerungen zu erkennen, die über eine Vielzahl von Datenschnittstellen verfügen [105].

Die genannten Punkte sollten eigentlich zu einer starken Zunahme des Sensoreinsatzes in Roboteranlagen führen. Es existieren jedoch immer noch starke Hindernisse, die diese Entwicklung blockieren. Die wesentlichen Gründe hierfür sind die unzureichende Standardisierung der Datenschnittstellen und der hohe Engineeringaufwand, der mit der Sensordatenintegration verbunden ist.

3.4.1 Unzureichende Standardisierung der Datenschnittstellen

Bezüglich der Standardisierung der Datenschnittstellen sind analoge, digitale und binäre Schnittstellen unproblematisch, da deren Signalpegel einheitlich genormt sind [27]. Auch die logische Weiterverarbeitung in der Robotersteuerung ist problemlos handhabbar, da jede Schnittstelle jeweils nur eine Information überträgt. Dies kann beispielsweise ein analoger Meßwert oder ein digitaler Zustandswert sein.

Der Datenaustausch zwischen der Robotersteuerung und einem intelligentem Sensorsystem funktioniert gemäß dem ISO-Referenzmodell [65]. Dieses sieht allerdings nur für die sichere Datenübertragung in den Ebenen 1 und 2 einheitliche Definitionen vor. Da zwischen Robotersteuerung und intelligentem Sensorsystem nur Punkt-zu-Punkt-Verbindungen betrachtet werden, können die Ebenen 3 bis 6 außer Acht gelassen werden (Bild 12). Eine besondere Bedeutung hat die Anwendungsschicht 7. Sie stellt das Bindeglied zwischen dem eigentlichen Prozeß (Roboterprozeß) und der Datenübertragung (Datensicherungsebene und physikalische Ebene) dar. Die Schwierigkeit besteht darin, daß bei einem intelligenten Sensorsystem sämtliche Informationen bzw. Daten über einen gemeinsamen Kanal übertragen werden. Dies bedeutet, daß durch die Anwendungsebene 7 eine Vereinheitlichung von

- Protokollaufbau
- Kommunikationssteuerung
- Datendarstellung und
- Aktionskennung.

erfolgen muß. Allerdings führt eine Standardisierung und damit Verallgemeinerung immer zu einem Overhead, der einen Performanceverlust zur Folge hat. Dies stellte in früheren Jahren ein nicht zu vernachlässigendes Problem dar, da die zur Verfügung stehenden Rechenleistungen noch sehr begrenzt waren. Heute tritt dieses Problem zunehmend in den Hintergrund, da moderne Prozessoren und breitbandige Übertragungskanäle ausreichend Performance bereitstellen.

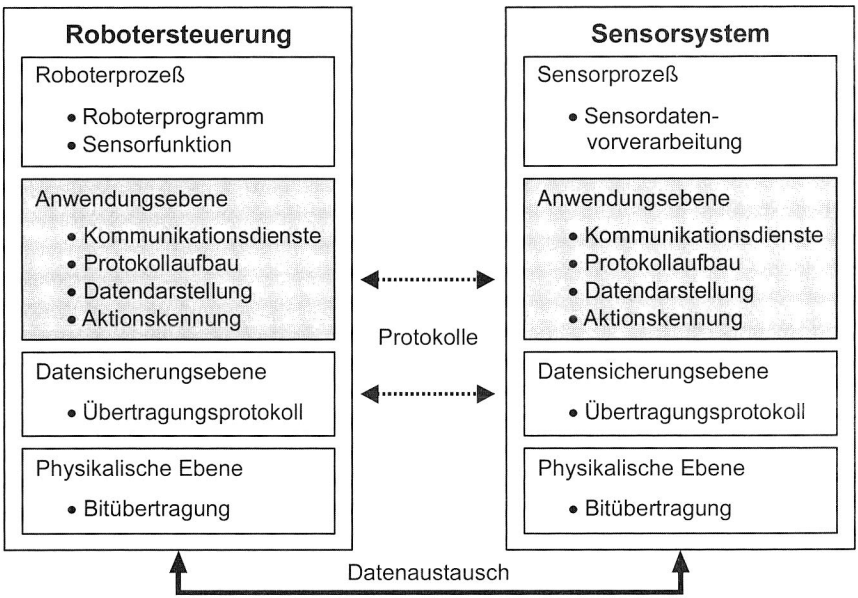


Bild 12: Kommunikationsstruktur zwischen intelligentem Sensorsystem und Robotersteuerung nach dem ISO-Referenzmodell

Speziell im Bereich der Sensor-Roboter-Kopplung wurde bereits im Jahre 1984 vom Projektträger Fertigungstechnik im KFK Fördermittel zur Erarbeitung eines Schnittstellenkonzepts für eine Sensorschnittstelle eingesetzt. 1986 konnte das Ergebnis der Arbeit dem zuständigen Normungsausschuß Maschinenbau im DIN vorgestellt werden. Auf Beschluß des Fachbereichs-Beirats wurde im Januar 1987 das Konzept schließlich als Deutsche Vornorm DIN V 66311 herausgegeben [98].

Trotz umfangreicher Bemühungen konnten die Normierungsbemühungen in der Industrie nicht durchgesetzt werden. Dafür gab es mehrere Gründe. Zum einen war damals die Bereitschaft zur Öffnung der eigenen Steuerung, durch normierte Schnittstellen, bei den Herstellern noch nicht vorhanden. Durch Abschottungsstrategien hoffte

man die Kunden an sich zu binden und damit die eigenen Marktanteile dauerhaft zu sichern. Außerdem liegt es im Interesse jedes Herstellers erst die Entwicklungskosten für die eigenen proprietären Schnittstellen zu erlösen, bevor in neue Entwicklungen investiert wird. Der Hauptgrund war jedoch, daß die Industrie große Hoffnungen in die damals eingeführte Feldbustechnologie setzte. Mit einem einheitlichen Feldbusprotokoll sollten die Anforderungen, die die Anwendungsschicht stellt, erfüllt werden. Nach nunmehr über zehnjährigen Diskussionen und der Marktetablierung vieler proprietärer Feldbussysteme ist die Idee eines Standardfeldbusses endgültig gescheitert [84].

3.4.2 Hoher Engineeringaufwand für Entwicklung und Wartung

Das Fehlen eines einheitlichen Standards für Datenschnittstellen zwischen Robotern und Sensorsystemen führte zu aufgaben- und sensorspezifischen Protokollen und Datendarstellungen. Die Folge ist ein hoher Entwicklungswand bei den Sensorherstellern, die ihre eigenen Protokolle und Kommunikationsmechanismen an die Datenschnittstellen der jeweiligen Robotersteuerungen anpassen müssen. Die Roboterhersteller hingegen müssen die Sensorfunktionen ihrer Steuerungen ebenfalls an die Aufgabenstellung und das verwendete Sensorsystem adaptieren. Der Wiederverwendungsgrad ist dabei gering. Die Integration eines neuen Sensorsystems erfordert deshalb einen nicht zu vernachlässigenden Entwicklungsaufwand in der Inbetriebnahmephase. Hinzu kommt, daß jede Realisierung individuellen Charakter hat. Dies erhöht den Schulungs- und Wartungaufwand für das Bedienpersonal.

Die Folge ist eine geringe Bereitschaft zur Nutzung der Potentiale, die ein Sensoreinsatz heute bietet. In der Industrie werden nur bei den Anwendungen Sensorsysteme eingesetzt, bei denen es der Prozeß unbedingt erfordert. Ansonsten wurden bisher Investitionen in aufwendige und unflexible Spanntechnik bevorzugt.

Seit kurzem ist in der Industrie eine gewisse Sensibilisierung für dieses Problemfeld zu beobachten, da moderne Fertigungstechnologien ohne Sensoreinsatz zunehmend schwieriger beherrschbar sind. Die Antwort der Roboterhersteller beschränkt sich jedoch auf die enge Kooperation mit wenigen Partnern aus der Sensorbranche. Konkret unterstützt eine Robotersteuerung nur ein bis zwei Sensorsysteme der Systempartner. Für die Sensorsysteme anderer Hersteller werden keine Integrationsmöglichkeiten angeboten. Die von den Endnutzern geforderte Offenheit ist auf diese Art und Weise nicht erreichbar.

Zur Lösung der Integrationsproblematik sind neue Ansätze erforderlich, die konsequent an der Erreichung eines maximalen Grades an Offenheit, Leistungsfähigkeit und Bedienerfreundlichkeit ausgerichtet sind. Moderne Softwaretechnologien bieten heute Möglichkeiten diese Zielsetzung zu erreichen. Diese Arbeit leistet dazu einen wesentlichen Beitrag.

4 Offenheit durch Komponententechnologie

Offenheit ist heute der Schlüsselbegriff im Bereich der informationstechnischen Systeme. Auf Anwenderseite werden damit hohe Rationalisierungserwartungen verknüpft. Von Offenen Systemen verspricht man sich eine schnellere Marktverfügbarkeit, ein verbessertes Preis/Leistungsverhältnis, eine bessere Anpaßbarkeit an spezifische Anwendungsanforderungen und vieles mehr [69]. Diese Ziele sind durchaus erreichbar. Die beiden erfolgreichsten Offenen Systeme der Technikgeschichte – der Personalcomputer und das Internet – haben dies auf beeindruckende Weise deutlich gemacht.

Nach dem Siegeszug der Offenen Systeme im Bereich der Geschäftsprozesse, ist nun auch ein deutlicher Trend zu Offenen Systemen in der Automatisierungstechnik spürbar [101]. Der zunehmende Einsatz PC-basierter Steuerungstechnik macht dies deutlich [46,80]. Es genügt jedoch nicht, konventionelle Automatisierungsgeräte durch Industrie-PCs zu ersetzen, um damit Offene Steuerungssysteme zu erhalten [112]. Der Schlüssel zu Offenen Systemen liegt heute mehr denn je in den Softwareanteilen. Einen vielversprechenden Ansatz stellt die Komponententechnologie dar.

Dieses Kapitel diskutiert die Kriterien für Offene Systeme und stellt aktuelle Entwicklungen zu deren Realisierung, auf Basis von Komponenten, vor.

4.1 Kriterien für Offenheit

In der Praxis wird dem Begriff "offen" meistens als Gegenpol "proprietär" gegenübergestellt. Das ist falsch, denn auch ein proprietäres System kann einen gewissen Grad an Offenheit beinhalten.

Die polaren Begriffspaare, die es hier zu betrachten gilt, sind [20]:

- Offen ⇔ Geschlossen
- Proprietär ⇔ Public Domain .

Bild 13 macht die Zusammenhänge durch eine Einteilung in vier Zonen (I bis IV) deutlich. Die Vergangenheit ist durch Zone I geprägt. Etablierte Hersteller haben versucht durch Abschottung ihrer proprietären Systeme ihre Marktanteile zu sichern. Der starke Zuwachs in der Komplexität moderner Systeme führt jedoch dazu, daß sich Systeme öffnen müssen, um vorhandene Standardfunktionalitäten integrieren zu können. Deshalb ist ein starker Trend von Zone I zu Zone II zu beobachten. Das Ziel bilden Systeme, die vollkommen auf einer offenen Architektur basieren. Diese Migration verläuft meist in mehreren Stufen über einen längeren zeitlichen Horizont.

Ein erster Schritt in Richtung Offener Systeme besteht darin, proprietäre Systeme mit offenen Schnittstellen auszustatten. Hierbei soll betont werden, daß Offenheit nicht als

absoluter Begriff zu verstehen ist, nach der Devise: Ein System ist entweder offen oder geschlossen. Vielmehr existieren verschiedene Ausprägungen von Offenheit.

Die Zone III ist inhaltlich durch die vorwiegend PC-basierte Public-Domain-Software geprägt und spielt im industriellen Umfeld keine beachtenswerte Rolle.

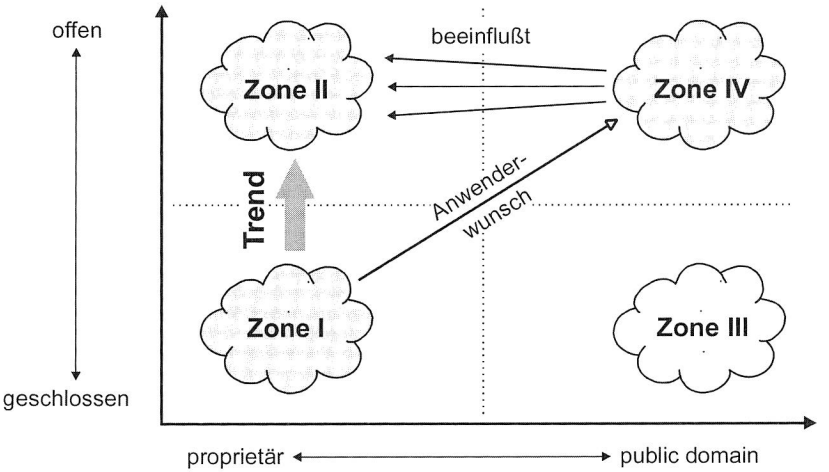


Bild 13: Darstellung der polaren Begriffspaare zur Offenheit (nach [20])

Die Zone IV ist als Wunschvorstellung für Offene Systeme zu betrachten. Sie charakterisiert Software und Systeme, die gleichermaßen offen wie public domain (also kostenlos) sind. Zwar gibt es einige beachtliche Entwicklungen in diesem Bereich, die vorwiegend durch amerikanische Universitäten getragen sind. Das GNU-Projekt der Free Software Foundation [40] ist ein Beispiel für derartige Produkte. Aber die Erfahrung zeigt, daß es ohne kommerzielle Anreize in der Regel auch keine Weiterentwicklung und keinen Support gibt. Schon deswegen muß industriellen Anwendern abgeraten werden, derartige Produkte in Betracht zu ziehen. Bemerkenswert ist allerdings der nicht unerhebliche Einfluß dieser ideellen Bemühungen auf die Weiterentwicklung der Konzepte zur Realisierung Offener Systeme.

4.1.1 Definition des Begriffs Offenheit

Es gibt eine Vielzahl unterschiedlicher Definitionen für den Begriff der Offenheit. Die wesentlichen Aspekte werden von der folgenden IEEE-Definiton [63] sehr gut beschrieben.

"Open System Environment: a comprehensive and consistent set of international information technology standards and functional standards that specify interfaces, services

and supporting formats to accomplish interoperability and portability of applications, data and people...".

Bemerkenswert an dieser Definition ist, daß sie betont, daß sich die Offenheit von Systemen nicht allein auf technische Komponenten, wie Hardware und Software, beschränken darf. Sie schließt auch die Daten und vor allem die Menschen, die mit diesen Systemen arbeiten, ein.

Konkret werden zwei Kriterien genannt, die jedes vollständig offene System erfüllen muß. Dies ist zum einen die Interoperabilität und zum anderen die Portabilität.

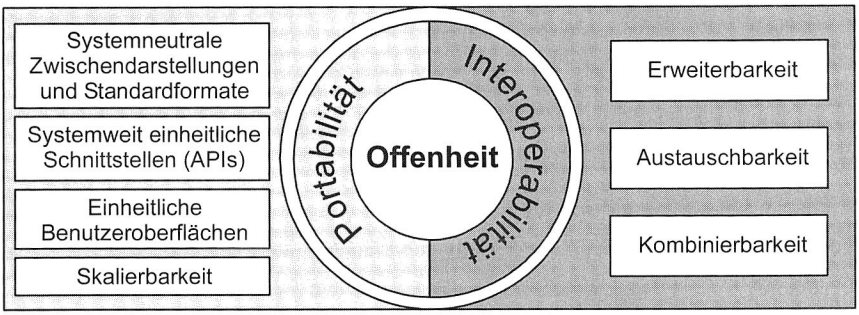


Bild 14: Interoperabilität und Portabilität als wichtigste Ziele von Offenheit

4.1.2 Definition von Interoperabilität

Interoperabilität ist das wichtigste Ziel Offener Systeme (Bild 14). Sie stellt die unbedingt erforderliche Basis für ein Offenes System dar. Interoperabilität ist gegeben, wenn heterogene Systeme mit unterschiedlichen Zweckbestimmungen in einem Verbund zusammenwirken, so daß sie sich wie ein einziges homogenes Leistungsgefüge darstellen [20,64].

Häufig wird Interoperabilität mit Kommunikationsfähigkeit (Konnektivität) gleichgesetzt. Dies ist nur bedingt richtig, denn Konnektivität ist eine notwendige, aber keine hinreichende Voraussetzung. Interoperabilität fordert mehr. Auf Basis der Konnektivität wird die gemeinsame Nutzung von verfügbaren Ressourcen, wie Daten oder Funktionalitäten der Teilsysteme, angestrebt.

Die Interoperabilität umfaßt die Teilbereiche [51]

- Erweiterbarkeit: Es muß möglich sein, nachträglich ein System um Teilsysteme zu ergänzen/erweitern.
- Austauschbarkeit: Teilsysteme können durch vergleichbare Teilsysteme, z.B. von einem anderen Hersteller, ersetzt werden.

- Kombinierbarkeit: Teilsysteme sind in der Lage, mit anderen Teilsystemen zusammenzuarbeiten, um ein leistungsfähiges Gesamtsystem zu bilden.

4.1.3 Definition von Portabilität

Portabilität ist das zweite wichtige Ziel Offener Systeme. Es beschreibt die Fähigkeit von Teilsystemen, diese ohne Änderung in anderen Systemen einsetzen zu können. Dieser Punkt ist vor allem für Softwaresysteme relevant. Voraussetzung für Portabilität ist eine plattform- und betriebssystemunabhängige Implementierung.

Portabilität bezieht sich dabei nicht ausschließlich auf Anwendungen. Gemäß der IEEE-Definition schließt Portabilität auch die Daten und den Menschen mit ein. Die Portabilität von Daten erfordert systemneutrale Zwischendarstellungen und die Verwendung von Standardformaten [63]. Außerdem müssen Zugriffsmöglichkeiten mit einheitlichen Schnittstellen über den gesamten Systemverbund geschaffen werden. Portabilität in Hinblick auf den Menschen bedeutet die Vereinheitlichung der Benutzeroberflächen und damit der dahinterstehenden Bedienphilosophie. Ein Wechsel zwischen unterschiedlichen Systemen soll ohne einen zusätzlichen Lernaufwand und kurzfristige Effizienzverluste vollzogen werden können. Portabilität schließt auch Skalierbarkeit ein. Darunter versteht man die Anpassung der Systemressourcen an die wirtschaftlichen, technischen und organisatorischen Anforderungen.

4.2 Industrielle Herstellung von Software

Die bisher gemachten Aussagen über Offenheit bezogen sich auf die relativ abstrakte Systemebene. Im dem technischen Umfeld, das für diese Arbeit relevant ist, sind Systeme durch Hardware- und Softwareanteile geprägt. Die Entwicklung eines Offenen Systems unterscheidet sich zwischen den Hardware- und den Softwareanteilen wesentlich.

Prinzipiell gilt, daß Standards die Fundamente bilden, auf denen Interoperabilität und Portabilität aufbauen. In Bezug auf die Hardware existieren eine Reihe internationaler Standards, die das Zusammenwirken unterschiedlicher Hardwarekomponenten definieren. Bestes Beispiel ist der Personalcomputer. Dieser kann mit einer Vielzahl von Komponenten unterschiedlicher Hersteller aufgebaut werden, da die hardwaretechnischen Schnittstellen standardisiert sind. Die Realisierung eines Offenen Softwaresystems ist um ein Vielfaches schwieriger. Dies liegt vor allem, aber nicht ausschließlich, an dem wesentlich geringeren Standardisierungsgrad in der Softwaretechnik. Vielmehr gestaltet sich der gesamte Entwicklungsprozeß wesentlich komplexer, als bei der Hardwareentwicklung. Zum besseren Verständnis der folgenden Kapitel wird auf diesen Sachverhalt im folgenden näher eingegangen.

Software unterscheidet sich von Hardware vor allem durch die Immaterialität und den hohen Abstraktionsgrad. Die Gegenständlichkeit der Hardwareentwicklung führt zu

meßbaren Zwischenergebnissen in der Entwicklungsphase (Bild 15). Außerdem findet eine Rückkopplung zwischen Entwicklung und Produktion statt, die ein Lernen am Objekt ermöglicht. In der Hardwareentwicklung erfordert eine Veränderung am Produkt häufig Anpassungen in den Bereichen Konstruktion und Produktionsmittel, die mit entsprechend hohen Aufwendungen verbunden sind. Allerdings geben die physikalischen Gesetzmäßigkeiten einen Rahmen für realisierbare Lösungen vor und ermöglichen den Einsatz mathematischer Hilfsmittel als Unterstützung bei Entwicklung und Erprobung. Wie bereits erwähnt, erleichtern vorhandene Standards die Hardwareentwicklung zusätzlich und fördern darüber hinaus die Entwicklung Offener Systeme.

Bei der Softwareentwicklung ist der Quellcode leicht veränderbar und damit prinzipiell leichter an Veränderungen anpaßbar. Eine Überprüfung der Auswirkungen von Quellcodeänderungen erfordert aber einen enorm hohen Testaufwand. Erschwerend kommt hinzu, daß Software im Gegensatz zu Hardware keine Toleranzen zuläßt. Als Konsequenz ist die Erfüllung der Funktionalitäten eines Softwaresystems idealerweise nur durch eine vollständige Fehlerfreiheit erreichbar, die vor allem bei größeren Projekten kaum umsetzbar ist.

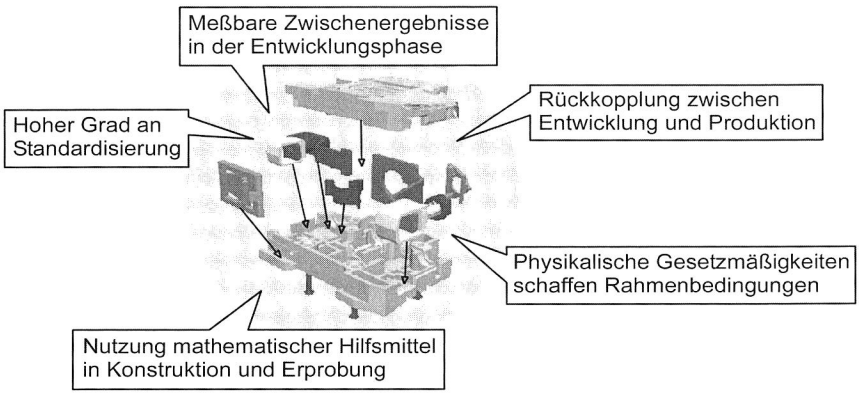


Bild 15: Vorteile, die den Entwicklungsprozesses von Hardwarebaugruppen erleichtern

Heute stellt Softwareentwicklung in weiten Bereichen noch eine kreative, fast künstlerische Tätigkeit dar, da für eine gegebene Aufgabenstellung eine Vielzahl individueller Realisierungsmöglichkeiten bestehen. Die Anforderungen des industriellen Softwareeinsatzes verlangen jedoch nach einem ingenieurgemäßen Vorgehen. Ansätze, wie das sog. Software Engineering, definieren lediglich die Reihenfolge und den Umfang von durchzuführenden Arbeitsschritten während der Softwareerstellung. Dies umfaßt z.B. die Planung und Durchführung von Tests oder die Erstellung unterstützender Dokumentation. Die Praxis zeigt, daß damit die angestrebte "industrielle Herstellung"

von Software, die vor allem durch immer kürzere Produktlebenszyklen bei steigender Komplexität erschwert wird, nicht realisierbar ist. In der Informatik spricht man in diesem Zusammenhang von einer "Softwarekrise". Einen Lösungsweg zeigt der Vergleich mit dem industriellen Produktionsprozeß auf.

Die Komplexität bei der Produktentwicklung und Produktfertigung wird durch einen modularen Produktaufbau beherrschbar. Die Idee, Systeme durch das Zusammensetzen kleinerer Teile zu bauen, gibt es in den mechanischen Ingenieurdisziplinen spätestens seitdem die ersten Maschinen aus Schrauben, Zahnrädern und Bolzen gebaut wurden. Ein weiteres Beispiel zeigt die Elektrotechnik auf. Dort werden Schaltungen aus Widerständen, Dioden und ICs zusammengesetzt. Man kann beobachten, daß die Entwurfsmethoden in allen Disziplinen Systeme in kleinere funktionale Einheiten herunterbrechen. Bekannt ist dieses Verfahren unter der Bezeichnung "funktionelle Dekomposition" [88]. Dieses funktionale Herunterbrechen ist ein wichtiges Mittel, weil dadurch erst die Handhabbarkeit und Übersichtlichkeit eines Systems gegeben ist. Außerdem ist es möglich, daß bereits bestehende Teile wiederverwendet werden und die Herstellung bestimmter Teile von anderen Zulieferfirmen übernommen werden. Voraussetzung hierfür ist auch die Einhaltung strenger Qualitätsrichtlinien. Die Dekomposition erleichtert natürlich auch die Qualitätssicherung, da Teilsysteme über weniger zu überprüfende Funktionalität verfügen als das Gesamtsystem. Die von Henry Ford begründete Fließbandfertigung beruht genau auf diesen Dekompositions-Prinzipien. Sie stellt auch heute noch die Grundlage für die industrielle Arbeitsteilung und damit für die Massenfertigung qualitativ hochwertiger Produkte dar.

Auch im Bereich der Softwareentwicklung wurden die Vorteile des funktionalen Herunterbrechens bereits früh erkannt. In den frühen prozeduralen Programmiersprachen wurden die Konzepte von Subroutinen bereits implementiert. Die Wiederverwendung von Softwareteilen war jedoch auf die umständliche Handhabung über Funktionsbibliotheken jahrzehntelang beschränkt. Erst die Einführung des objektorientierten Programmierparadigmas schaffte eine konzeptionelle Basis für die umfassende Wiederverwendung von Softwareteilen. Allerdings ist die Wiederverwendbarkeit auf die Quellcodeebene beschränkt. Ein Objekt ist stark durch eine bestimmte Programmiersprache, in der es implementiert ist, geprägt. Lauffähig ist ein Objekt nur, wenn es an die richtige Stelle in seiner Klassenhierarchie eingebettet ist. Die Wiederverwendung eines Objekts ist nur in einem Softwaresystem möglich, das objektorientiert mit der passenden Programmiersprache realisiert wird. Die Schnittstellen des Objekts entsprechen den Standards der jeweiligen Programmiersprache und sind auch nur mit ihrer Hilfe ansprechbar. Objekte sind immer Teil eines Softwaresystems und nicht alleine verwendbar.

Das Ziel besteht jedoch darin, dasselbe "Plug&Play" zu bieten, wie es für die Hardwareindustrie längst Standard ist. Dieses Ziel scheint seit einigen Jahren mit der Komponententechnologie in greifbare Nähe zu rücken.

4.3 Einführung in die Komponententechnologie

Bisher wurden in der Softwareentwicklung strukturierte und objektorientierte Programmierverfahren eingesetzt. Die Komponententechnologie stellt ein völlig neues Programmierparadigma dar. Die zugrundeliegenden Konzepte basieren auf den Erfahrungen, die in den letzten einhundert Jahren bei der industriellen Herstellung von Massengütern gewonnen wurden. Der naheliegende Ansatz der funktionalen Dekomposition konnte bisher aufgrund der fehlenden Basistechnologien noch nicht im erforderlichen Maße umgesetzt werden. Mit den heute zur Verfügung stehenden Softwaretechnologien rückt die Vision von der industriellen Herstellung von Software in greifbare Nähe. Damit wird jedoch nicht nur der Vorgang der Softwareerstellung erleichtert, sondern vor allem die Entwicklung Offener Softwaresysteme gefördert.

4.3.1 Definition des Komponentenbegriffes

Für den Begriff "Komponente" existiert keine eindeutige Definition. Jeder Experte assoziiert damit, in Abhängigkeit des Anwendungskontextes, verschiedene Begrifflichkeiten. Für diese Arbeit soll folgende Definition nach [108] verwendet werden:

Eine Komponente ist ein binärer, funktional in sich abgeschlossener Softwarebaustein, der über wohl definierte Schnittstellen mit seiner Umgebung in Beziehung tritt und dabei in der Lage sein muß, in unterschiedliche, nicht vorhersagbare Anwendungskontexte eingebettet zu werden.

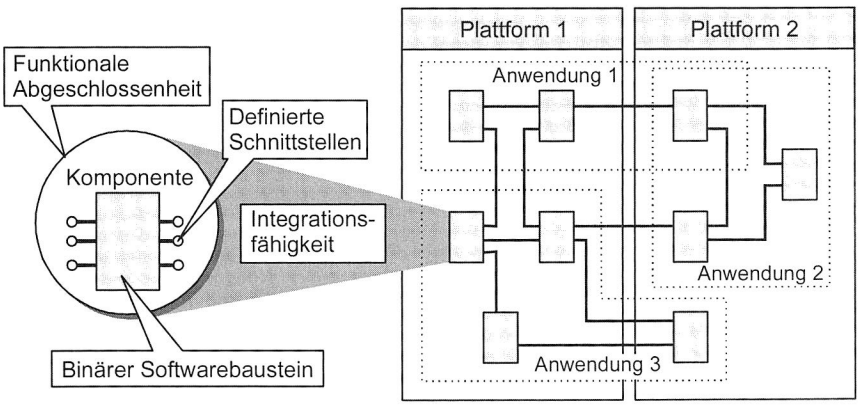


Bild 16: Charakterisierung einer Softwarekomponente

Eine Komponente liegt als binärer Softwarebaustein vor. Das bedeutet, daß es sich um bereits kompilierten, also ausführbaren Code handelt (Bild 16). Komponenten sind deshalb unabhängig von einer Programmiersprache oder einer Implementierungsweise. So kann eine Komponente objektorientiert oder prozedural implementiert sein, unter der Verwendung einer beliebigen Programmiersprache. Bekannte Konzepte aus der Objekttechnologie, vor allem Vererbungsmechanismen, sind im Komponentenansatz nicht enthalten. Die mit Vererbungsmechanismen einhergehenden engen Verbindungsabhängigkeiten stehen der Forderung nach unabhängig voneinander entwickelbaren Softwarekomponenten und deren breiter Wiederverwendung entgegen.

4.3.2 Vorteile der komponentenbasierten Softwareentwicklung

Die Komponententechnologie kann den Softwareentwicklungsprozeß entscheidend effizienter gestalten. Sie hat das Potential zu signifikant besserer Produktivität, Wartbarkeit und Qualität. Im folgenden werden die vier wichtigsten Vorteile, die durch die Komponententechnologie ermöglicht werden, vorgestellt:

- Software kann schneller und mit weniger Aufwand erstellt werden, weil passende, bereits bestehende Komponenten wiederverwendet werden können.
- Die Verwendung von qualitativ hochwertigen, am Markt erhältlichen Komponenten führt zu höherer Qualität in den Gesamtsystemen.
- Der Wartbarkeitsaufwand kann deutlich verringert werden, weil das Warten einer Komponente dazu führt, daß die behobenen Fehler allen diese Komponente benutzenden Softwaresystemen zugute kommt.
- Die strikte Modularisierung von Softwaresystemen in Komponenten und die Standardisierung ihrer Schnittstellen machen sie flexibler und leichter erweiterbar.

Die funktionale Abgeschlossenheit von Komponenten und deren Integrierbarkeit in anwendungsspezifische Anwendungskontexte führt dazu, daß sich Unternehmen damit beschäftigen, Komponenten professionell zu entwickeln und zu vertreiben. Fachleute sehen darin einen vielversprechenden Markt für die Zukunft und sprechen bereits von einer Komponentenindustrie.

4.3.3 Essentielle Eigenschaften von Komponenten

Die komponentenbasierte Softwareentwicklung bringt die erwähnten Vorteile jedoch nicht automatisch. Einen wirklichen Nutzen bringt eine Komponentenindustrie nur, wenn die kommerziellen Komponenten über die folgenden, essentiellen Eigenschaften verfügen:

- Interessante Funktionalität

  Die Komponenten müssen für die Kunden interessante Funktionalität bieten. Sie

müssen funktional in die Systeme der Kunden passen und mit anderen Komponenten (auch anderer Hersteller) kooperieren.

- Leichte Auffindbarkeit

  Der Aufwand, um Komponenten mit gesuchter Funktionalität aufzufinden, muß gering sein. Programmierer verwenden Komponenten nur wieder, wenn der Aufwand, sie zu finden, zu verstehen und zu integrieren, geringer ist, als das Implementieren der entsprechenden Funktionalität.

- Ausreichende Performanz

  Das aus Komponenten zusammengesetzte Gesamtsystem muß den Anforderungen entsprechende, ausreichende Performanz haben. Besondere Bedeutung erlangt diese Aussage bei verteilten Systemen, da die Kommunikation über Netzwerke zu nicht vorhersagbaren Verzögerungen führen kann.

- Mittlere Granularität

  Die Größe einer Komponente ist immer ein Kompromiß zwischen ihrer Wiederverwendbarkeit (Wahrscheinlichkeit der Wiederverwendung) und ihrem Wiederverwendungsgrad (Menge an wiederverwendetem Code). Kleine Komponenten bieten einfache Funktionalität. Deswegen sind sie einfach zu beschreiben, zu verstehen, aufzufinden und dadurch wiederzuverwenden. Ihre Wiederverwendbarkeit ist hoch. Das Problem von kleinen Komponenten ist aber, daß eine sehr große Anzahl von Komponenten erforderlich ist, um die gesamte Funktionalität eines durchschnittlich komplexen Systems zu implementieren. Der Aufwand des Zusammensetzens zu einem System ist sehr groß. Ebenso steigt der Kommunikationsaufwand mit der Anzahl an Komponenten.

  Große Komponenten wiederum verringern den Aufwand des Zusammensetzens und für die Kommunikation, weil sie wesentlich mehr Funktionalität bieten. Der Wiederverwendungsgrad großer Komponenten ist hoch. Das Problem von großen Komponenten ist aber, daß ihre Funktionalität viel spezifischer ist und die Wahrscheinlichkeit ihrer Wiederverwendung dadurch viel geringer ist. Die Wahrscheinlichkeit, daß eine große Komponente in ein Gesamtsystem paßt, ist deutlich geringer als bei kleinen Komponenten.

  Bedenkt man diesen Kompromiß, so liegt die Erkenntnis nahe, daß Gesamtsysteme nur dann die geforderte Flexibilität, Qualität und Wartbarkeit besitzen können, wenn sie aus einer "mittleren" Anzahl von (mittel-)großen Komponenten zusammengesetzt werden.

Beim Umstieg auf die Komponententechnologie ist der volle Nutzen einer komponentenbasierten Vorgehensweise kurzfristig nicht sichtbar. Zunächst sind sogar bestimmte Investitionen und organisatorische Maßnahmen erforderlich, die erhöhte Kosten verur-

sachen. Die umfangreiche Wiederverwendung von Komponenten kann erst mittelfristig erfolgen, wenn im Laufe der Zeit immer mehr Komponenten zur Verfügung stehen und ein Anwendungsentwickler die Einsatzmöglichkeiten und Funktionalität der Komponenten mehr oder weniger vollständig exploriert hat.

4.4 Verteilte Komponenten-Architekturen

Um ein komponentenbasiertes Softwaresystem aufbauen zu können, benötigt man neben geeigneten Komponenten auch eine detaillierte Beschreibung der internen Struktur des Softwaresystems. Eine derartige Beschreibung bezeichnet man als Architektur [106].

Eines der Fundamentalprinzipien Offener Systeme ist die Client/Server-Ausrichtung (Bild 17). Diese als Auftraggeber/Auftragnehmer-Prinzip bekannte Technik beschreibt ganz allgemein einen Verbund kooperierender Systemkomponenten, wobei die als Clients bezeichneten Auftraggeber die von Auftragnehmern (Server) angebotenen Dienste anfordern. Unter einem Dienst versteht man die Funktionalität, die ein Protokoll seinem Benutzer anbietet. Ein Dienst wird dabei immer von einem Protokoll erbracht [18]. Ein Protokoll definiert eine Menge von Regeln, die das Verhalten bei Kommunikationsvorgängen festlegen. Zu den Regeln gehören u.a. solche für den Aufbau und Abbau von Verbindungen, über die Formate der auszutauschenden Nachrichten, über den zu benutzenden Code mitsamt den Absprachen über Fehlererkennung und Fehlerkorrektur [106].

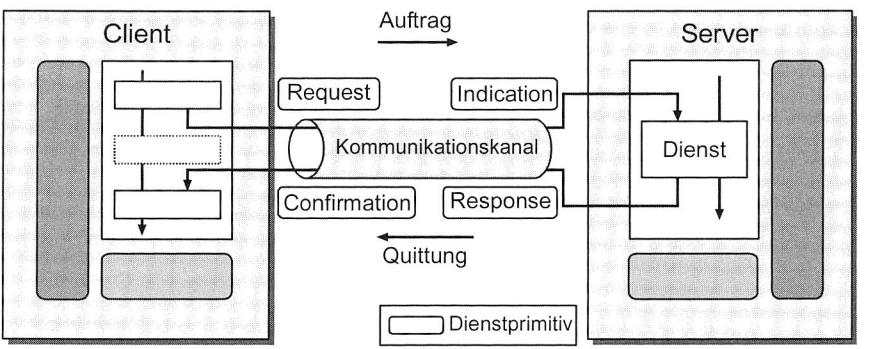


Bild 17: Abläufe bei der Client/Server-Kommunikation

In der Informationstechnik haben sich bisher eine Vielzahl unterschiedlicher Protokolle für die Kommunikation zwischen Client und Server etabliert. Diese Vielfalt erschwert die Integration von Komponenten. Aus diesem Grund ist es wichtig, die Komponenten

von den technischen Details der Kommunikation abzuschirmen. Dies wird durch die sog. Middleware erreicht.

Nach [12] versteht man unter Middleware eine transparente Vermittlungsschicht, die Kommunikationsmechanismen zwischen Client und Server bereitstellt. Dazu besitzt sie standardisierte Programmierschnittstellen (API - Application Programming Interface) und basiert auf standardisierten Übertragungsprotokollen.

Konkret stellt eine Middleware APIs bereit, die ein Client nutzen kann, um einen Dienst des Servers anzufordern. Weiterhin übernimmt die Middleware die Übermittlung der Dienstanforderung sowie die Rückmeldung des Servers. Die Middleware setzt auf dem Betriebssystem auf und entkoppelt damit die Kommunikationspartner von den spezifischen Eigenschaften der Hardware und des Betriebssystems. Dieses Konzept unterstützt die Verteilung von Anwendungskomponenten auf heterogene Hardwareplattformen, die über heterogene Netzwerke verbunden sein können.

Middlewarearchitekturen sind auf der Basis unterschiedlicher Technologien realisierbar. Neben einfachen nachrichten-basierten Mechanismen finden auch Mechanismen auf der Basis von Remote Procedure Calls (RPC) [110] Verwendung. Die weite Verbreitung des objektorientierten Paradigmas führte jedoch dazu, daß sich heute objekt-basierte Middlewarearchitekturen durchgesetzt haben.

Da eine Komponente in binärer Form vorliegt, ist das zugrundeliegende Programmierparadigma frei wählbar. Komponenten können sowohl strukturiert als auch objektorientiert implementiert werden. In der Regel herrscht jedoch eine objektorientierte Implementierung vor. Bei Verwendung der Objektorientierten Programmierung stellt eine Komponente Klassen zur Verfügung.

Ein Objektmodell standardisiert Mechanismen, mit denen die in den Komponenten implementierten Klassen angesprochen und verwendet werden können [97]. Mit der Definition eines Objektmodells soll garantiert werden, daß Objekte, die von verschiedenen Entwicklern in aller Welt stammen und in unterschiedlichen Programmiersprachen entwickelt wurden, in binärer Form miteinander zusammenarbeiten. Außerdem muß es möglich sein, binär vorliegende Objekte nicht nur innerhalb eines gemeinsamen Prozeßraumes, sondern auch über Rechnergrenzen hinweg zu verwenden. Ein verteiltes Objektmodell übernimmt zusätzlich zu der Programmiersprachenintegration die Aufgabe, Methodenaufrufe an ein entferntes Objekt zu vermitteln. Es stellt die Grundlage für verteilte Softwaresysteme zur Verfügung. Aufgrund der elementaren Bedeutung des Objektmodells wird dieses in der Fachliteratur auch allgemein als Komponententechnologie bezeichnet.

Die beiden wichtigsten Komponententechnologien sind CORBA (Common Object Request Broker Architecture) von der OMG (Object Management Group) und COM

(Component Object Model) von Microsoft. Zur Einordnung dieser Technologien sollen diese im folgenden kurz vorgestellt werden.

4.4.1 Die Komponententechnologie CORBA

CORBA [86] ist eine von dem Standardisierungskonsortium OMG erarbeitete Spezifikation, die die Zusammenarbeit von Softwaresystemen über die Grenzen von Programmiersprachen, Betriebssystemen und Netzwerken hinweg regelt [86]. Die OMG setzt sich aus über 800 Herstellern und Anwendern zusammen. Die Schnittstellen von CORBA sind plattformunabhängig. Aus diesem Grund bietet sich CORBA vor allem als Middleware in heterogenen Rechnerwelten an. CORBA hat eine verteilte Registrierung (Interface Repository), was die Administration vereinfacht. Die Standardisierung der CORBA-Spezifikation durch die OMG erlaubt es einer großen Anzahl von IT-Unternehmen, Einfluß auf die weitere Zukunft von CORBA zu nehmen. Dabei müssen allerdings auch die Nachteile der sich zum Teil widersprechenden Vorstellungen der Mitgliedsfirmen in Kauf genommen werden.

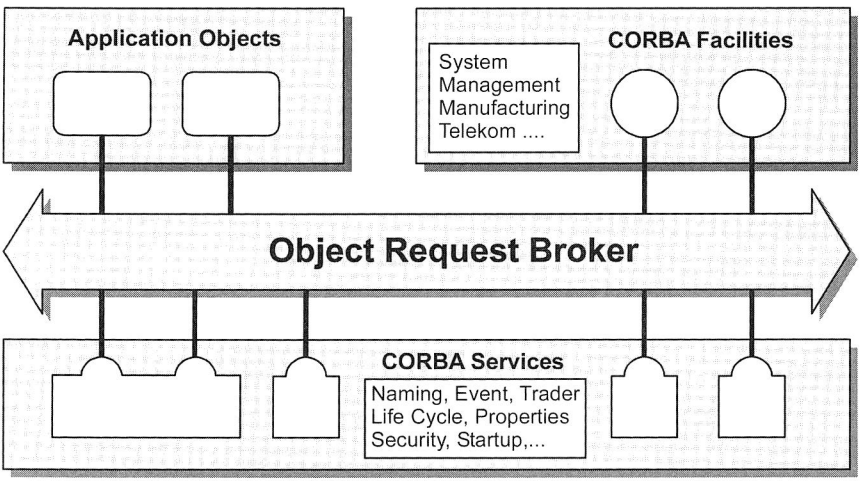


Bild 18: Aufbau der Common Object Request Broker Architecture (nach [86])

Eine verteilte Anwendung mit CORBA muß explizit aufgebaut werden, und die notwendige Infrastruktur muß in jedem Fall gesondert erworben und installiert werden. Sie ist nicht, wie DCOM, auf einer großen Anzahl der typischen Rechner automatisch vorhanden. CORBA basiert auf einem Object Request Broker (ORB), der als Softwarebus eine Reihe von Systemdiensten anbietet (Bild 18). Diese Systemdienste ermöglichen die Interaktion zwischen verteilten Komponenten. CORBA stellt den

Entwicklern von ORBs lediglich das Konzept zur Verfügung, das den Funktionsumfang dieser Dienste beschreibt. Wie der ORB aber tatsächlich implementiert wird, bleibt ihnen überlassen.

4.4.2 Die Komponententechnologie COM

COM ist nicht CORBA-konform. Es basiert vielmehr auf dem Distributed Computing Environment (DCE) und den dort verwendeten Remote Procedure Calls (RPC) [110]. Die DCE-Architektur hat einige Ähnlichkeit mit einem CORBA-ORB, allerdings wird hier nicht mit Methoden von Objekten, sondern mit Prozessen gearbeitet. Methodenaufrufe werden über Referenzen (Pointer) ausgeführt. Damit ist COM im Gegensatz zu CORBA kein klassisches Objektmodell, da es keine Vererbung auf Schnittstellenbasis unterstützt. Für die Integration von Komponenten bietet sich COM deshalb vor allem wegen der guten Kapselung an, bei der man sich nur mit Schnittstellen, nicht aber mit Vererbungsstrukturen befassen muß.

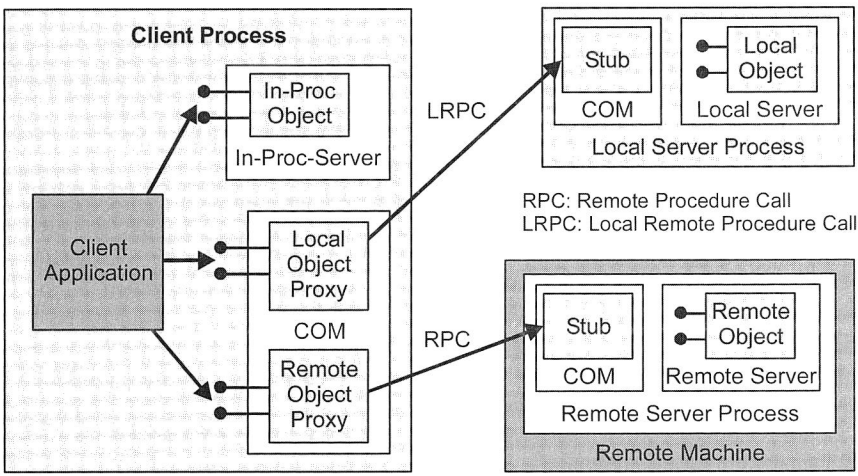


Bild 19: Kommunikation in COM/DCOM über Remote Procedure Calls (nach [110])

Die Komponententechnologie COM [32] ist für die Vermittlung von Methodenaufrufen zwischen binären Objekten zuständig, die sich sowohl im selben Prozeßraum als auch in separaten Prozeßräumen auf demselben Rechner befinden (Bild 19). COM stellt somit einen ORB für einen einzelnen Rechner zur Verfügung. Nach Einführung von DCOM (Distributed Component Object Model) ist auch die Verteilung von COM-Objekten über Rechnergrenzen hinweg möglich. DCOM stellt damit einen vollständigen ORB dar.

COM hat eine große Verbreitung im Desktop-Bereich sowie im Bereich Low-End-Server-Systeme, da es von jedem Windows-Betriebssystem unterstützt wird. Durch die weltweite Verbreitung der Windows-Systeme hat sich COM als De-facto-Standard etabliert. Die Weiterentwicklung der COM-Technologie ist von den Entscheidungen Microsofts abhängig. Somit hat die IT-Industrie nur geringen Einfluß auf die Zukunft von COM.

4.4.3 Trennung von Schnittstelle und Implementierung

Erst durch eine strikte Trennung von Schnittstelle und Implementierung ist die Portierung einer Komponente von einer auf eine andere Plattform unter Beibehaltung derselben Schnittstelle möglich. Sowohl CORBA als auch DCOM erlauben die Spezifikation von Schnittstellen schon während der Entwurfsphase eines Systems in einer implementierungsunabhängigen Schnittstellenbeschreibungssprache (engl. Interface Definition Language, Abk. IDL) [60]. Durch die Trennung von Schnittstellendefinition und Klassenimplementierung wird der Zustand eines Objektes gekapselt und kann nur durch verschiedene, in einer Schnittstellendefinition enthaltenen Methoden verändert werden. Je strikter das Modell diese Kapselung einhält, um so transparenter ist für den Entwickler der Umgang mit den Objekten.

Eine moderne Middleware, auf Basis einer Komponententechnologie, muß über eine erweiterte Funktionalität verfügen, als dies früher in konventionellen Client/Serversystemen der Fall war. Middleware ist heute mehr, als die Kapselung der Datenübertragung zwischen Client und Server. Ein verteiltes Objektmodell muß darüber hinaus Mechanismen für die Instanzierung, Verwaltung und Löschung lokaler und verteilter Objekte zur Verfügung stellen.

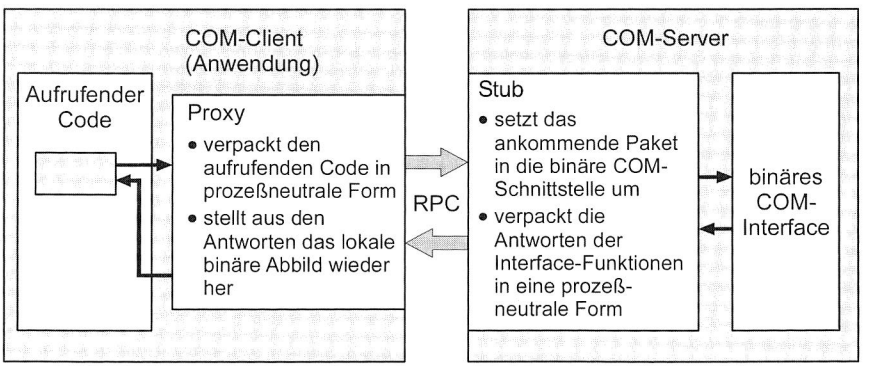


Bild 20: Marshalling/Unmarschalling-Prozeß am Beispiel von COM/DCOM

Ein entfernter Methodenaufruf ist mit dem bloßen Versenden einer Nachricht nicht realisierbar (Bild 20). Befindet sich das Serverobjekt außerhalb des Prozeßraumes des Methodenaufrufers (Client), so besteht ein entfernter Methodenaufruf im wesentlichen aus dem Auffinden des entfernten Objekts, der Zusammenstellung und Verpackung des Methodenaufrufes (Marshalling), dem Entpacken des Methodenaufrufs (Unmarshalling), der Ausführung der Methode und der Zurücksendung des Ergebnisses [110].

4.5 Entkopplung von Funktionalität und Struktur

Der Charakter komponentenbasierter Software legt stärker als jedes andere Softwareparadigma die Unterscheidung zwischen einer strukturellen Sicht und einer funktionalen Sicht auf ein Softwaresystem nahe. Die Funktionalität wird von den einzelnen Komponenten erbracht, während die Struktur, die den Rahmen für den Programmablauf festlegend, von den Verbindungen der Softwarekomponenten untereinander bestimmt wird. Die strukturellen Gesichtspunkte komponentenbasierter Software sollen im folgenden näher beleuchtet werden.

Die Komponentendefinition verlangt, daß eine Komponente eine möglichst abgeschlossene Funktionalität beinhaltet. Bei Funktionalitäten geringen Komplexitätsgrades ist diese Forderung leicht zu erfüllen. Nimmt die Komplexität jedoch zu, stellt sich die bereits diskutierte Frage der optimalen Granularität. Der Komponentenentwickler muß dann entscheiden, ob er die Komplexität auf mehrere Komponenten aufteilt (funktionale Dekomposition). Zur Erfüllung ihrer Funktionalität muß eine Komponente selbst die Rolle eines Clients übernehmen und Dienste einer anderen Komponente nutzen. Bei der Verteilung der Funktionalität auf mehrere Komponenten entsteht damit eine feste Kopplung auf Quellcodeebene zwischen den beteiligten Komponenten. Durch diesen Kopplungsprozeß wird die Struktur des Softwaresystems festgelegt. Erfordert die Anwendung eine strukturelle Veränderung, so müssen die Kopplungen auf Quellcodeebene neu gestaltet werden.

Bisher wurde diesem Problem wenig Beachtung gewidmet, da einmal erstellte Businessanwendungen, die heute noch bei weitem die Mehrheit komponentenbasierter Softwaresysteme darstellen, in der Regel keine strukturellen Änderungen erfahren. Dies liegt darin begründet, daß die Abläufe im Banken- und Versicherungswesen, aber auch im E-Commerce, weitgehend vereinheitlicht wurden. Sollte dennoch eine strukturelle Änderung notwendig werden, so wird der Quellcode der betroffenen Komponenten angepaßt und diese durch eine Updateprozedur ausgetauscht.

Ein anderes Bild bietet sich im Maschinen- und Anlagenbau. Sinkende Losgrößen und steigende Variantenvielfalt erfordern Systeme die schnell und einfach an wechselnde Anwendungsanforderungen adaptierbar sind. Modulare Hardwarekonzepte ermöglichen eine einfache Erweiterbarkeit und Austauschbarkeit von Hardwarekomponenten. So können beispielsweise rekonfigurierbare Bearbeitungsmaschinen um

zusätzliche Achsen erweitert werden [107] oder in flexiblen Montagesystemen Automatisierungsmodule durch manuelle Bearbeitungsstationen ersetzt werden [33]. Andererseits ist es für einen Maschinenbauer bzw. Endkunden aus Zeit und Kostengründen nicht akzeptabel, bei Veränderungen an der Hardwarestruktur Softwarekomponenten im Quellcode anzupassen und neu zu compilieren.

Die Lösung dieser Problemstellung liegt darin, die festen Kopplungen zwischen Softwarekomponenten durch lose Kopplungen zu ersetzen. Unter einer losen Kopplung ist zu verstehen, daß sie nicht im Quellcode festgeschrieben wird, sondern parametrierbar gestaltet wird. Konkret spezifiziert ein derartiger Parameter eine spezifische Schnittstelle, die die benötigte Komponente anbietet. Zur Erstellung einer Anwendung aus kommunizierenden Softwarekomponenten wird deshalb eine kompositorische Beschreibung benötigt, die die einzelnen Schnittstellen in Bezug setzt. Hierfür bietet sich die Verwendung einer Scriptsprache an. Voraussetzung ist allerdings, daß sie eine einheitliche Beschreibungsform bietet und unabhängig von den einzelnen eingesetzten Implementierungssprachen der Komponenten ist. Bei einer Scriptsprache handelt es sich nicht um eine Programmiersprache, die Funktionalitäten implementiert, sondern um eine Beschreibungsform, die Funktionalitäten spezifiziert. Ihre Syntax sieht deshalb keine typisierten Variablen oder Referenzen vor. Neben der Beschreibung der Kopplungen zwischen Komponenten besteht eine weitere Aufgabe der Scriptsprachen darin, die Parametrierung der Komponenten zu beschreiben. Es handelt sich hierbei um eine verallgemeinerte Sichtweise, da die Kopplungsbeziehungen ebenfalls als Parameter verstanden werden können.

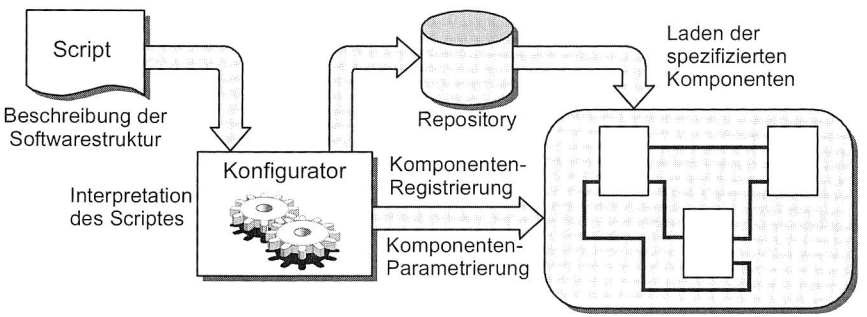


Bild 21: Konfiguration der Softwarestruktur nach Scriptvorgaben

Scriptsprachen sind interpretierte Sprachen. Die Interpretation eines Scriptes und die Umsetzung der im Script beschriebenen Funktionalitäten erfolgt durch den Konfigurator (Bild 21). Dazu bedient er sich Systemdiensten des Betriebssystems und der Middleware. Über Betriebssystemdienste werden die, auf Datenträgern abgelegten,

binären Komponenten in die Laufzeitumgebung geladen. Über Dienste der Middleware werden die Komponenten registriert und parametriert. Danach können sie von Client-Anwendungen genutzt werden.

Die Trennung von Beschreibung und Ausführung ermöglicht es die Syntax der Scriptsprache unabhängig von der verwendeten Plattform und Komponententechnologie zu gestalten. Trotzdem ist es bisher noch nicht gelungen einen Standard für die Syntax einer universell einsetzbaren Scriptsprache zu definieren. Bisher ist es üblich, daß die Entwickler eines komponentenbasierten Softwaresystems auch eine individuelle Syntax für die Scriptsprache entwickeln.

Zum Abschluß dieses Kapitels soll der Stand der Technik zu komponentenbasierten Softwaresystemen im fertigungstechnischen Umfeld vorgestellt werden.

4.6 Komponentensysteme in der Produktionstechnik

In steuerungstechnischer Hinsicht sind im Produktionsbereich im wesentlichen zwei zentrale Automatisierungsgeräte anzutreffen. Die Speicherprogrammierbare Steuerung (SPS) und die Numerische Steuerung (NC bzw. CNC).

Die Hauptaufgabe einer SPS besteht darin, über ihre I/O-Module den Ablauf eines technischen Prozesses zu steuern. Bei einer Ablaufsteuerung ist der Steuerungsvorgang in einzelne Schritte gegliedert, die entsprechend einem Programm nacheinander abgearbeitet werden. Das Steuerprogramm ist dabei nicht im Quellcode verankert, sondern wird anwendungsspezifisch von einem SPS-Programmierer in einem geeigneten Entwicklungssystem erstellt und anschließend auf das SPS-Laufzeitsystem überspielt. Die dabei verwendeten Programmiersprachen sind international in der IEC 61131-3 [61,119] genormt und bieten damit einen gewissen Grad an Offenheit. Da die gesamte Steuerungsfunktionalität allein durch das SPS-Programm bestimmt wird, ist ein Komponentenansatz hier nicht anwendbar.

Ein anderes Bild ergibt sich bei den Numerischen Steuerungen. Die Kernfunktionalität einer derartigen Steuerung besteht darin, auf Basis eines NC-Programms nach DIN 66025 [30], ein oder mehrere Achssystemen derart anzusteuern, daß eine gewünschte Bewegungssequenz umgesetzt wird. Dazu sind eine Reihe von Teilfunktionalitäten, wie z.B. Bahnplanung, Interpreter, Transformator und Lageregler auszuführen. Hinzu kommen sekundäre Funktionalitäten, wie z.B. Werkzeug- und Werkstückverwaltung, NC-Programmverwaltung, Simulation der NC-Programme, Logikfunktionen, usw. Numerische Steuerungen werden zur Ansteuerung einer Vielzahl von Maschinentypen mit unterschiedlichen Kinematiken eingesetzt. Aufgrund der stark variierenden Anforderungen werden konventionelle Steuerungen von den Maschinenherstellern an deren Maschinen angepaßt.

4.6.1 Internationale Entwicklungen

Durch den steigenden Kostendruck und die wachsenden technologischen Anforderungen wird es immer wichtiger Teilfunktionalitäten eines Steuerungssystems wiederzuverwenden bzw. austauschen zu können. Aufgrund der hohen volkswirtschaftlichen Bedeutung des Maschinenbaus wurden Anfang der neunziger Jahre auf nationaler und internationaler Ebene Forschungsprojekte zur Entwicklung offener Systemarchitekturen für Numerische Steuerungen angestoßen [71]. Dazu gehört auf europäischer Ebene das ESPRIT-Projekt OSACA (Open System Architecture for Controls within Automation) [6] sowie das daran anschließende Projekt HÜMNOS (Herstellerübergreifende Module für den nutzerorientierten Einsatz der offenen Steuerungsarchitektur). Im amerikanischen Raum existieren ähnliche Ansätze in den Projekten NGC (Next Generation Controller) [77,81], EMC (Enhanced Machine Controller) [91] und OMAC (Open Modular Architecture Controller) [120]. Weitere Aktivitäten die sich mit offenen Steuerungsarchitekturen beschäftigen sind das japanische OSEC-Projekt (Open System Environment for Controllers) [41] oder das kanadische ORTS (Open Real Time Operating System for CNC Design) [1].

Beispielhaft für Offene Systeme im Bereich der Numerischen Steuerungen soll im folgenden die Steuerungsarchitektur des Europäischen Projektes OSACA vorgestellt werden.

4.6.2 Beschreibung der OSACA-Steuerungsarchitektur

Die Grundlage eines Offenen Systems ist eine standardisierte Systemplattform und darauf aufsetzend einheitliche Schnittstellendefinitionen zu den Anwendungsprogrammen. In dem ESPRIT III-Projekt OSACA [6,75] wurden die erforderlichen Spezifikationen ausgearbeitet. Darüber hinaus wurde eine Systemplattform entwickelt und beispielhafte Softwaremodule implementiert.

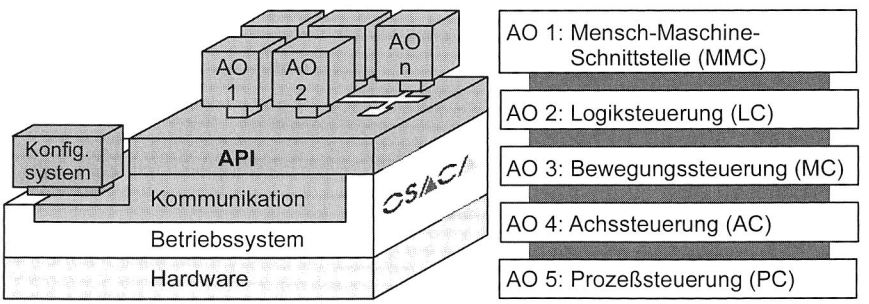


Bild 22: Systemplattform und Architekturobjekte der OSACA-Architektur (nach [75])

Die OSACA-Systemplattform ist in einer Schichtenarchitektur gegliedert und besteht aus Systemhardware und Systemsoftware (Bild 22). Die unterste Schicht bildet die Systemhardware, die elektronische Komponenten, wie Prozessorboard oder I/O-Karten, beinhaltet. Die Systemsoftware wird gebildet durch das Betriebssystem, Treibern für Erweiterungskarten und vor allem dem Kommunikationssystem. Auf die Funktionalitäten der Systemplattform kann über Dienste durch einen einheitlichen Satz von Schnittstellen (API) zugegriffen werden. Das API versteckt die spezifische Implementierung von Plattformdiensten und macht damit den Zugriff auf Plattformdienste unabhängig von der verwendeten Hardware und dem Betriebssystem.

Anwendungsspezifische Softwaremodule tragen bei OSACA den Namen Architekturobjekte (AO). Diese werden objektorientiert implementiert und in AO-Klassen bereitgestellt. Insgesamt sind fünf Architekturobjekte spezifiziert: Mensch-Maschine-Schnittstelle (MMC), Logiksteuerung (LC), Bewegungssteuerung (MC), Achssteuerung (AC) und Prozeßsteuerung (PC). Jedes dieser Architekturobjekte ist wiederum unterteilt in Submodule. Beispielsweise besteht die Bewegungssteuerung aus den Submodulen: Bahnplanung, Interpreter, Vorbereitung und Durchführung.

Interoperabilität und Portabilität wurden bereits als wesentliche Eigenschaften einer Offenen Systemarchitektur definiert. Diese Eigenschaften werden durch die drei Hauptbestandteile der OSACA-Architektur unterstützt (Bild 23).

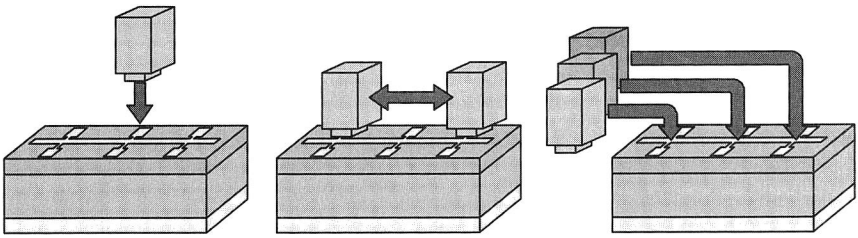


Bild 23: Hauptbestandteile der OSACA-Architektur (nach [75])

Die Referenzarchitektur beschreibt, welche Funktionalität von welchen AOs angeboten wird, indem sie Attribute und Dienste definiert. Weiterhin werden zusätzliche Vereinbarungen festgelegt, die eine eindeutige Interpretation und Nutzung der ausgetauschten Daten ermöglichen. Nur mit diesen zusätzlichen Regeln kann Interoperabilität zwischen AOs unterschiedlicher Hersteller erreicht werden.

Die Interoperabilität von AOs wird durch die Bereitstellung eines standardisierten Kommunikationssystems ermöglicht. Das Kommunikationssystem erlaubt nicht nur den Datenaustausch zwischen AOs, sondern definiert auch die notwendigen Proto-

kolle. Das Kommunikationssystem unterstützt den Datenaustausch zwischen AOs, die sich entweder auf dem selben Prozessorboard befinden oder die auf unterschiedlichen Boards verteilt sind. Verteilte Systeme sind über ein Bussystem oder jede andere Art von Transportmechanismus verbunden. Das Kommunikationssystem setzt auf dem jeweiligen Transportkanal auf und verbirgt so den Transportmechanismus vor den AOs. Für den Datentransport wird ein OSACA-spezifisches Protokoll verwendet, das einheitliche Datenformate und Nachrichtenformate sicherstellt. Die Portabilität der AOs wird garantiert durch die alleinige Nutzung des API für den Zugriff auf Plattformdienste, da das API auf allen Plattformen identisch ist.

Eine Numerische Steuerung auf Basis der OSACA-Architektur besteht aus der Zusammenschaltung einer Sammlung von anwendungsspezifischen AOs. Um die Steuerung an spezifische Anforderungen, z.B. neue Kinematiken (Hexapod), einfach anpassen zu können, genügt es entsprechenden AOs, bzw. deren Submodule, auszutauschen oder die Steuerung um neue AOs bzw. Submodule zu erweitern. Um dies zu erreichen, muß die Systemplattform über ein Konfigurationssystem verfügen. Dieses instanziiert zur Laufzeit die benötigten AOs aus vorhandenen Klassenbibliotheken und startet diese. Außerdem stellt es die Kommunikationsverbindungen für den Datenaustausch zwischen den AOs her. Die Informationen für den Konfigurationsvorgang entnimmt das Konfigurationssystem einer Konfigurationsdatei. Die Syntax wird dabei von einer OSACA-spezifischen Skriptsprache definiert, die von dem Konfigurationssystem interpretiert wird.

OSACA und die anderen weiter oben erwähnten Forschungsansätze beinhalten viele Gemeinsamkeiten. Sie sind jedoch nicht kompatibel. Ein Austausch von Softwaremodulen ist nicht möglich. Vergleichbar zum SPS-Markt ergibt sich damit für Offene Systeme im Bereich der Numerischen Steuerungen eine kontinentale Dominanz einzelner Lösungen. Nachdem die unterschiedlichen Ansätze realisiert und ihre Funktionsfähigkeit nachgewiesen wurde, sollten in Zukunft die Anstrengungen in Richtung einer gemeinsamen Plattform gehen. Nur damit kann den Anforderungen eines globalen Marktes Rechnung getragen werden und die Industrie zur Übernahme der Konzepte motiviert werden.

5 Komponenten einer konfigurierbaren Softwarearchitektur für eingebettete Hardwareplattformen

Auf der Grundlage der in Kapitel 4 herausgearbeiteten Konzepte, soll ein offenes, an spezifische Anwendungen anpaßbares Softwaresystem entwickelt werden. Der Einsatzzweck dieses Softwaresystems besteht darin, Steuerungsfunktionalitäten bereitzustellen, die die Grundfunktionalität einer Maschinensteuerung anwendungsspezifisch erweitert. Um die Integrationsfähigkeit und Maschinenunabhängigkeit sicherzustellen, verfügt das Softwaresystem über eine eigene Hardwareplattform. Diese wird in die Maschinensteuerung eingebettet. Man spricht deshalb von einem eingebetteten System (engl. Embedded System).

Die zugrundeliegende Softwarearchitektur muß die Forderung nach einer funktionalen Anpassung an die jeweiligen Anwendungsanforderungen optimal unterstützen. Dies wird durch einen Komponentenansatz erreicht, der um die Möglichkeit zur dynamischen Konfigurierbarkeit erweitert wurde. Die Komponenten einer derartigen Softwarearchitektur werden in den folgenden Kapiteln näher vorgestellt.

5.1 Grundlegendes Software-Modell

Ein Software-Modell wird durch die Aufgaben und Eigenschaften seiner Komponenten sowie durch ihre gegenseitige Abhängigkeit charakterisiert. Ein Software-Modell für dynamisch konfigurierbare Softwaresysteme sollte über folgende Komponenten verfügen [36]:

Funktionale Einheiten (Softwaremodule)

Diese Komponenten realisieren eine benötigte Teilfunktionalität, im Sinne einer funktionalen Dekomposition. Im Rahmen des Softwaresystems werden sie als Softwaremodule bezeichnet. Sie werden vorwiegend als eigenständige Rechenprozesse, meist in Form von Tasks oder Threads, realisiert.

Modulbibliothek

In der Modulbibliothek werden die bereits übersetzten Objektfiles der Softwaremodule in binärer Form abgelegt. In Abhängigkeit von der anwendungsspezifisch benötigten Funktionalität werden die benötigten Softwaremodule aus der Modulbibliothek entnommen und auf die Hardwareplattform übertragen.

Kommunikationsmechanismus

Die resultierende Softwarestruktur wird neben den funktionalen Einheiten maßgeblich von dem Informationsfluß zwischen diesen Einheiten bestimmt. Diese Aufgabe ist einem Kommunikationsmechanismus zugeordnet.

Konfigurator

Die Hauptaufgabe des Konfigurators besteht im dynamischen Bilden der Softwarestruktur. Für den Anwender fungiert er als Ausführungsinstanz, die aufgrund eines Konfigurierungsauftrages die Struktur der Software korrekt aufbaut. Dazu müssen die Softwaremodule auf die Hardwareplattform geladen und ausgeführt werden. Die Struktur ist vollständig aufgebaut, wenn die Kommunikationsverbindungen zwischen den Softwaremodulen etabliert sind.

Ablaufsteuerung

Bei dynamisch gebildeten Systemen ist es notwendig, daß die beteiligten Einheiten in ihrem Ablauf koordiniert und verwaltet werden. Ein technischer Prozeß erfordert meist die Einhaltung einer festgelegten Reihenfolge bei der Ausführung von Teilfunktionalitäten. Dies ist darauf zurückzuführen, daß eine funktionale Einheit zur Erfüllung ihrer eigenen Funktionalität die Ergebnisse einer vorher abgearbeiteten Funktionalität benötigt. Das Modul Ablaufsteuerung kontrolliert in Abhängigkeit von Ablaufanweisungen (Ablaufprogramm) die Abarbeitungsreihenfolge der Softwaremodule.

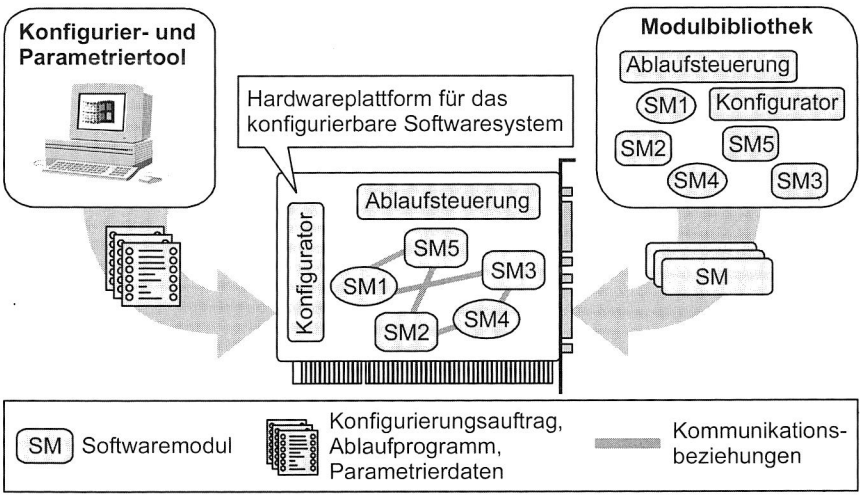


Bild 24: Darstellung der Komponenten des Software-Modells für dynamisch konfigurierbare Softwarestrukturen

Konfigurier- und Parametriertool

Für die benutzerfreundliche Erstellung des Konfigurierungsauftrages, der Ablaufanweisungen und der Parameterdateien zur Parametrierung der Softwarekomponenten wird

ein Konfigurier- und Parametriertool benötigt. Dieses wird auf einem handelsüblichen Personal Computer unter einer grafischen Benutzeroberfläche ausgeführt. Der Konfigurierungsauftrag und die Parameterdateien werden anschließend auf die Hardwareplattform übertragen.

Einen Überblick über die einzelnen Komponenten des Software-Modells liefert Bild 24.

5.2 Aufbau des Kommunikationsmechanismus

In Kapitel 4 wurden die Komponententechnologien CORBA und DCOM vorgestellt. Sie sind die Basis vieler Middleware-Realisierungen bei dispositiven Systemen. Als Middleware in einem Steuerungssystem sind sie jedoch ungeeignet, da sie die dort benötigten Echtzeitanforderungen nicht erfüllen können. Es wurde deshalb ein eigener Kommunikationsmechanismus entwickelt, der sich auf die benötigte Funktionalität beschränkt und die Resourcen- und Echtzeitanforderungen eines Embedded Systems berücksichtigt. Es soll deshalb auch nicht der Begriff Middleware verwendet werden.

5.2.1 Auswahl eines Dienstablaufmodells

Die Aufgabe des Kommunikationsmechanismus besteht im wesentlichen darin, Dienstanforderungen und Dienstergebnisse zwischen den Softwaremodulen zu vermitteln. Grundlage hierfür ist die Festlegung des verwendeten Dienstablaufmodells. Man unterscheidet zwischen dem Client/Server-Modell und dem Producer/Consumer-Modell (Bild 25).

Client/Server-Modell

Beim Client/Server-Modell nimmt ein Server eine Dienstanforderung von einem Client entgegen, führt den Dienst durch und gibt das Dienstergebnis an den Client zurück. Die Initiative für die Dienstdurchführung liegt beim Client. Die Sende- bzw. Empfangsereignisse, die ein Anwendungsprozeß bezüglich eines Dienstes erhalten bzw. verursachen kann, tragen die Bezeichnung Dienstprimitive, da sie die einfachsten, nicht weiter aufteilbaren Vorgänge in Bezug auf einen Dienst sind. Der Client-Prozeß löst dabei durch eine Dienstanforderung (Request) beim Server-Prozeß die Anzeige (Indication) einer Dienstanforderung aus. Der Server führt den geforderten Dienst aus und sendet eine entsprechende Antwort (Response) an den Client zurück. Mit dem Empfang der Dienstbestätigung (Confirmation) ist die Anforderung für den Client abgeschlossen.

Producer/Consumer-Modell

Im Gegensatz zu der engen Zweierbeziehung zwischen Client und Server wird beim Producer/Consumer-Modell von der Existenz eines systemweiten Datenbestandes ausgegangen, auf den sich alle Dienste beziehen. Producer ist derjenige Teilnehmer, der eine Information erzeugt, Consumer ist derjenige Teilnehmer, der eine Information für seine Aufgabe im technischen Prozeß benötigt. Es darf stets nur ein Producer für

eine Information gleichzeitig im System vorhanden sein, jedoch können beliebig viele Consumer für die Information vorhanden sein. Der Producer erzeugt die Information und fügt sie danach in den systemweiten Datenbestand ein. Der Consumer entnimmt die Information dann, wenn er sie benötigt. Deshalb sind im Rahmen eines Producer/Consumer-Modells nur die beiden Dienstprimitive Request und Confirm vorhanden.

Darüberhinaus bietet ein Producer/Consumer-Modell die Möglichkeit einen ereignisgetriebenen Transfer der Daten an die Consumer zu realisieren. Dabei werden die benötigten Daten automatisch an die anfordernden Consumer verschickt, sobald sich die Daten ändern. Dadurch wird der Datenverkehr auf dem Kommunikationssystem reduziert, da keine pollenden Anfragenachrichten versendet werden müssen.

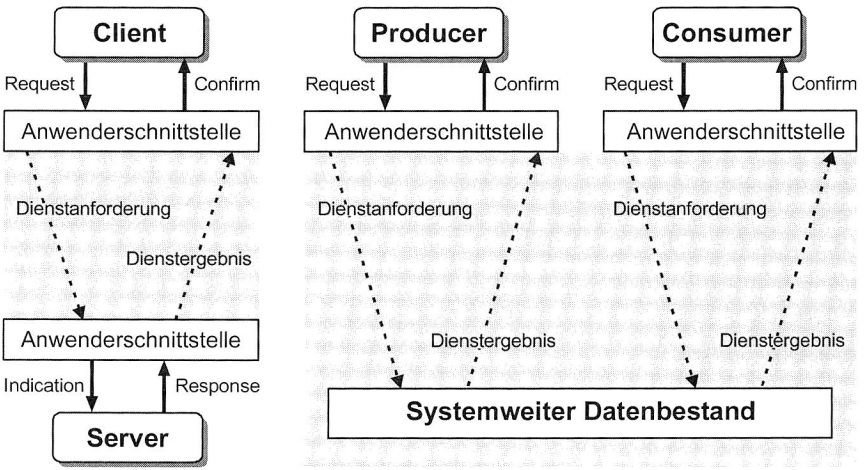


Bild 25: Unterschiedliche Dienstablaufmodelle bei Client/Server-Kommunikation und Producer/Consumer-Kommunikation

Bewertung der Dienstablaufmodelle

Das Producer/Consumer-Modell bietet in einem starren Softwaresystem einige Vorteile, da sich der Datenzugriff auf schnelle Speicherzugriffe reduzieren läßt. Eine weitere Performancesteigerung bietet der ereignisgesteuerte Datentransfer. In einem dynamisch konfigurierbaren Softwaresystem wechseln die Producer und Consumer in Abhängigkeit von den jeweiligen Anwendungsanforderungen. Dies erfordert ein Verwaltungssystem für den systemweiten Datenbestand. Konkret bedeutet dies, daß das Verwaltungssystem die Anfragen der Consumer einzeln verarbeiten muß. Dies bietet keinen Vorteil gegenüber der dezentralen Verarbeitung bei dem Client/Server-Modell. Ein weiterer Aspekt ist der vorteilhafte sequentielle Ablauf der Teilfunktionali-

täten in einem Steuerungssystem, der keinen ereignisgesteuerten Datentransfer benötigt.

Ein Producer/Consumer-Modell bietet im Bereich ereignisgetriebener, dispositver Anwendungen einige Vorteile. Im Steuerungsbereich, und ganz speziell bei dynamisch konfigurierbaren Softwaresystemen, sollte allerdings das Client/Server-Modell bevorzugt werden.

5.2.2 Auswahl des Schnittstellenkonzepts

Aus Sicht eines Softwaremoduls stellt sich der Kommunikationsmechanismus als eine Schnittstelle für den Datenaustausch mit anderen Softwaremodulen dar. Der Kommunikationsmechanismus basiert deshalb auf dem umzusetzenden Schnittstellenkonzept. Es existieren unterschiedliche Schnittstellenkonzepte, die im folgenden beschrieben und bewertet werden.

Nachrichtenorientierte Schnittstelle

Bei einer nachrichtenorientierten Schnittstelle erfolgt der Datenaustausch direkt zwischen den Prozessen der beteiligten Softwaremodule. Die zu transferierenden Daten werden zwischen den disjunkten Speichern der Prozesse in Form von Nachrichten (Messages) übertragen. Das dafür konzipierte Transportsystem wird als Kanal bezeichnet.

Nachrichten sind nach [28] eine Summe von Zeichen, die aufgrund von bekannten oder unterstellten Abmachungen Informationen darstellen. Zum Zwecke der Weitergabe werden diese als zusammenhängend angesehen und deshalb als Einheit betrachtet.

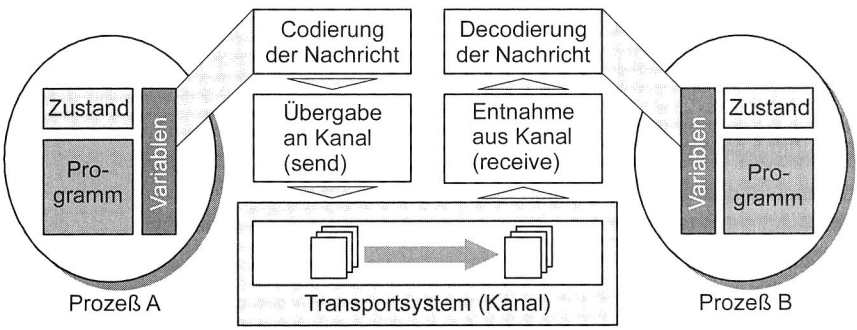


Bild 26: Datenaustauschmechanismus über eine nachrichtenorientierte Schnittstelle

Die Nachrichten müssen von der sendenden Anwendung als Datenstrukturen aufgebaut (codiert) werden. Der Nachrichtentransport erfolgt über zwei Transportope-

rationen, mit denen Nachrichten vom Sender an das Transportsystem übergeben (send) bzw. vom Empfänger aus dem Transportsystem übernommen werden (receive) (Bild 26). Beim Empfang werden die Nachrichten decodiert und in den Speicher des Prozesses übertragen. Der Prozeß löst daraufhin, in Abhängigkeit von der übertragenen Nachricht, resultierende Aktionen aus [43].

Die auf diesem Konzept basierenden Realisierungsformen lassen sich einteilen in:

- Speicherlose Transportsysteme
- Speichernde Transportsysteme
- Remote Procedure Calls.

Bei speicherlosen Transportsystemen werden die Nachrichten auf dem Übertragungsweg nicht zwischengespeichert. Die Nachricht wird aus dem lokalen Speicher des Senders direkt in den lokalen Speicher des Empfängers kopiert. Im Gegensatz dazu ist bei einem speichernden Transportsystem ein sog. Transporteur-Prozeß erforderlich. Der Sender übergibt die Nachricht an den Transporteur-Prozeß des Transportsystems. Im Transporteur-Prozeß wird die Nachricht solange zwischengespeichert, bis der Empfänger für die Übergabe der Nachricht bereit ist.

Der Remote-Procedure-Call (RPC), auch als "Remote Invocation" bezeichnet, stellt eine weitere Möglichkeit der Nachrichtenkopplung dar. Die Kommunikation zwischen Prozessen erfolgt durch den Aufruf von Prozeduren im jeweiligen Kommunikationspartner. Der Prozeduraufruf und das Aufrufergebnis stellen dabei die zu übertragenen Nachrichten dar. Die Wirkung eines Remote-Procedure-Calls ist dabei vergleichbar mit einem gewöhnlichen Prozeduraufruf, da der aufrufende Prozeß erst dann weiterarbeiten kann, wenn die externe Prozedur beendet ist.

Datenaustausch über gemeinsame Speicherbereiche

Die indirekte Form der Prozeßkopplung erfolgt über einen gemeinsam zugänglichen Datenbereich im Rechnersystem (engl. Shared memory). Diese Kopplungsart existiert bei nahezu allen Multitasking-Betriebssystemen. Das Synchronisationsproblem beim Datenzugriff wird dabei in der Regel mit dem in [25] entwickelten Semaphorenkonzept gelöst. Vor dem Zugriff auf den gemeinsamen Datenbereich, muß erst eine sog. Semaphore belegt werden. Falls die Semaphore schon belegt ist, wartet der Prozeß solange, bis sie wieder freigegeben wird.

Eine Weiterentwicklung stellt das Konzept der monitor-gekoppelten Prozeßsysteme [16,43,55] dar (Bild 27). Die gemeinsamen Daten und die darauf definierten Zugriffsfunktionen werden in einer abgeschlossenen Einheit zusammengefaßt, die als Monitor bezeichnet wird. Da der Zugriff auf die Daten nur über die entsprechenden Zugriffsfunktionen des Monitors erfolgen kann, ist durch die Unteilbarkeit der Funktionen ein gegenseitiger Ausschluß konkurrierender Zugriffe gewährleistet.

Bewertung der Schnittstellenkonzepte

Die indirekte Kopplung über einen gemeinsamen Hauptspeicherbereich bietet den Vorteil eines geringen Speicherbedarfs und einer kurzen Übertragungszeit. Außerdem sind die notwendigen Semaphorenmechanismen bei den meisten Programmiersprachen bzw. Betriebssystemen vorhanden. Trotzdem eignet sich dieser Mechanismus nur bedingt für den Datenaustausch. Ein wesentliches Problem ist, daß durch Adressierungsfehler die Stabilität des gesamten Softwaresystems beeinträchtigt werden kann. Ein weiterer Nachteil ist, daß die Zugriffsfunktionen und die zugehörigen Daten getrennt sind, und somit die Verständlichkeit der Software bei umfangreichen Anwendungen sehr schnell abnimmt. Das Monitorkonzept verhindert falsche Zugriffe auf die Daten und bewirkt eine Verbesserung der Übersichtlichkeit. Ein Grundproblem bei der Nutzung gemeinsamer Speicherbereiche bleibt jedoch bestehen. Ändert sich die Datenstruktur so zieht dies umfangreiche Änderungen in den Zugriffsfunktionen mit sich, die auf den gemeinsamen Speicher zugreifen.

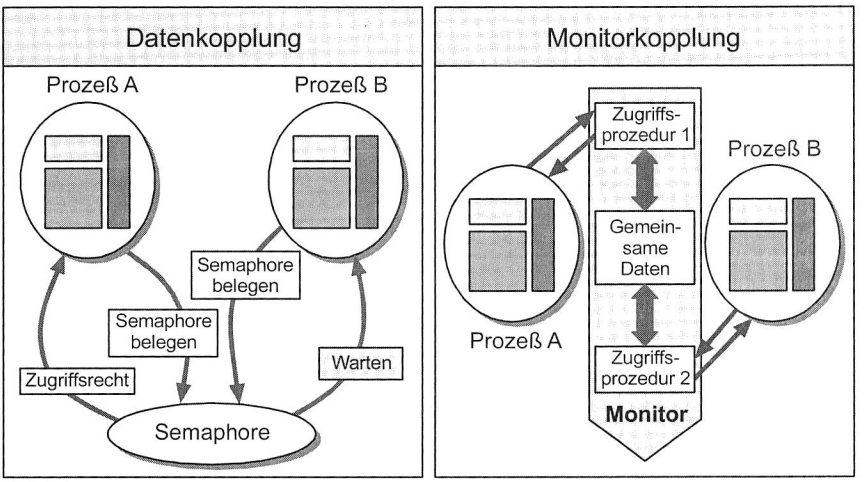


Bild 27: Mechanismen bei der Datenkopplung und der Monitorkopplung (nach [43])

Somit bleibt nur die Möglichkeit des direkten Datentransfers. Durch die vollständige Entkopplung der Daten ergibt sich eine geringere Fehleranfälligkeit und somit auch eine höhere Verfügbarkeit. Die Entkopplung der Daten bewirkt auch, daß sich Änderungen der Datenstrukturen in den meisten Fällen nur auf sehr wenige Prozesse auswirken, so daß offene, leicht anpaßbare Anwendungen aufgebaut werden können. Sowohl beim speicherlosen Transport als auch bei Remote-Procedure-Calls besteht aufgrund des gegenseitigen Wartens eine Deadlock-Problematik. Bei komplexen

Anwendungen steigt die Wahrscheinlichkeit, daß ein solcher Fall eintritt. Bei speichernden Transportsystemen sind hingegen Deadlock-Situationen kaum möglich, weil Senden und Empfangen von Daten voneinander entkoppelt sind. Hier spielt jedoch die Verfügbarkeit der Transporteurprozesse eine entscheidende Rolle. Da aber solche Transportprozesse als ausgetestete Standardfunktionen mit entsprechender Qualität in den meisten Betriebssystemen verfügbar sind, kann dieses Problem in der Regel vernachlässigt werden.

Beschreibung des Transportsystems Message-Queue

Aus der Bewertung der Schnittstellenkonzepte geht hervor, daß ein speicherndes Transportsystem für die Kopplung der Softwaremodule am geeignetsten ist. Die meisten Echtzeitbetriebssysteme stellen als speicherndes Transportsystem sog. Message-Queues zur Verfügung [117]. Eine Message-Queue stellt einen Mechanismus zum Speichern und Verwalten von Nachrichten dar. Über Betriebssystemaufrufe können Tasks Nachrichten einschreiben und auslesen [37]. Message-Queues erlauben darüber hinaus eine indirekte Synchronisierung und kommunikationstechnische Verknüpfung von Tasks. Da sie beliebige Nachrichten übertragen können, unterstützen sie die freie Kombinierbarkeit von Softwaremodulen. Die Adressierung einer Message-Queue erfolgt über einen systemweit eindeutigen Message-Queue Identifier (Message-Queue-ID).

Eine Message-Queue kann von beliebigen Tasks beschrieben und ausgelesen werden. Deshalb unterstützt dieser Mechanismus auch eine Client/Server-Kommunikation. Diese ist nach Bild 28 realisierbar.

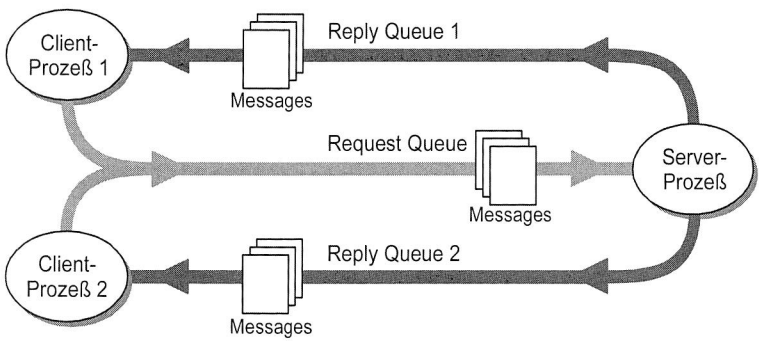


Bild 28: Realisierung einer Client/Server-Kommunikation mit Message-Queues

Bei einer Client/Server-Kommunikation erzeugt die Servertask eine Message-Queue, um Dienstanforderungen von allen Clients entgegennehmen zu können. Jede Dienstanforderung enthält den Identifier derjenigen Message-Queue, die der jeweilige Client

als Transportkanal für die Antwort vom Server vorgesehen hat. Im Vergleich zu Punkt-zu-Punkt-Verbindungen ist dieses Vorgehen wesentlich effektiver und resourcenschonender.

In einer dynamisch konfigurierten Umgebung ist die Adressierung der diensterbringenden Einheit problematisch. Theoretisch kann eine bestimmte Funktionalität, die im Softwaresystem benötigt wird, von einem beliebigen Softwaremodul bereitgestellt werden. Etabliert ein Softwaremodul eine Message-Queue, so ist deren Identifier prinzipiell beliebig und den anderen Modulen von vornherein nicht bekannt. Der Prozeßname eines Softwaremoduls muß jedoch eindeutig sein. Zur Lösung dieses Problems wird vom Konfigurator eine systemweit gültige Verbindungsliste angelegt.

5.3 Konzeption des Konfigurationsmechanismus

Konfigurierung bezeichnet im allgemeinen das Zusammenstellen eines Ganzen aus mehreren Einzelteilen [23]. Im Bezug auf Softwaresysteme wird darunter das Bilden der Struktur eines Softwaresystems aus mehreren Softwaremodulen verstanden.

5.3.1 Analyse von Konfigurierungsmechanismen

Steuerungssoftware kann prinzipiell zu unterschiedlichen Zeitpunkten im Entwicklungs- bzw. Nutzungsprozeß von Software konfiguriert werden [23].

Mechanismen auf Quellprogrammebene

Die erste Konfigurationsmöglichkeit besteht in der Bildung einer Softwarestruktur während der Codierphase. Die benötigte Funktionalität wird im Quellcode, in Form von Unterprogrammen, implementiert. Der Informationsfluß zwischen den funktionalen Einheiten erfolgt meist durch Übergabeparameter oder gemeinsame Speicherbereiche. Damit ist der Informationsfluß im Quellprogramm statisch festgelegt. Die Anpassung an anwendungsspezifische Anforderungen kann nur durch Parameter erfolgen. Bei erforderlichen Änderungen an der Softwarestruktur ist eine Neucompilierung erforderlich. Da der Quellcode in der Regel nicht frei verfügbar ist und darüberhinaus ein Compiliervorgang von einem Endbenutzer nicht verlangt werden kann, wird dieser Konfigurationsmechanismus verworfen.

Mechanismen auf Bindemodulebene

Betrachtet man die nächste relevante Phase der Softwareentwicklung, so kommen Mechanismen auf Bindemodulebene zum Einsatz (Bild 29). Aufgabenspezifische funktionale Einheiten werden dabei als Quellprogrammodule separat codiert, übersetzt und in einer Modulbibliothek abgelegt. Die benötigte Softwarestruktur wird in einem autonomen Initialisierungsprogramm festgelegt, welches sich auf die Elemente dieser Bibliothek bezieht. In einem Binde- und Ladevorgang wird anschließend das gesamte ausführbare Programm erzeugt und im Laufzeitsystem ausgeführt. Dieser Vorgang

kann durch Einsatz sogenannter Generatoren als unterstützende Werkzeuge automatisiert werden. Den Bindevorgang bezeichnet man als frühes Binden bzw. statisches Binden, da der Compiler schon zum Zeitpunkt der Übersetzung des Sourcecodes die tatsächlichen physikalischen Zieladressen von Funktionsaufrufen und Variablen festlegt. Grundsätzlicher Vorteil ist, daß schon zur Zeit der Kompilierung fehlerhafte Angaben vom Compiler gefunden werden können. Außerdem bieten solch früh gebundene Programme Laufzeitvorteile, da die Zieladressen feststehen und nicht bei jedem Aufruf neu berechnet werden müssen.

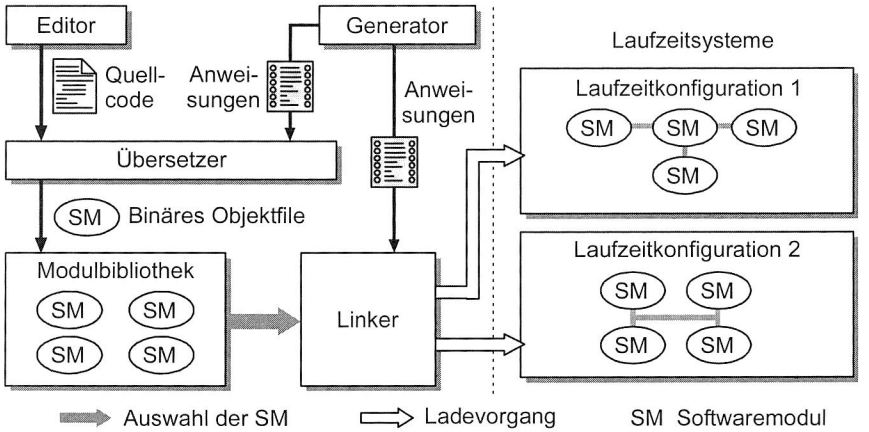


Bild 29: Konfigurationsmechanismus auf Bindemodulebene (nach [23])

Ein Nachteil dieses Konzeptes ist, daß hierfür zielsystemspezifische Entwicklungswerkzeuge wie Compiler, Linker und Lader erforderlich sind. Dies ist mit höheren Investitionen und vor allem spezifischen Qualifikationen des Bedienpersonals verbunden.

Dynamisches Konfigurieren zur Laufzeit

Setzt man diese Konzepte logisch fort, so stellt der letzte Schritt das Adaptieren von Funktion und Struktur des Softwaresystems erst zur Laufzeit dar. Voraussetzung ist die Existenz entsprechender Mechanismen zum dynamischen Binden von vorcompilierten Softwaremodulen zur Laufzeit. Derartige Mechanismen werden von einem modernen Echtzeitbetriebssystem bereitgestellt. Die übersetzten Softwaremodule stehen in binärer Form in einer Modulbibliothek zur Verfügung. Sie stellen damit, gemäß den Ausführungen in Kapitel 4, eine Softwarekomponente dar.

Die ausführende Instanz, die für die Konfiguration des Softwaresystems benötigt wird, wird im folgenden als Konfigurator bezeichnet. Alle für die Konfiguration benötigten

Angaben werden in einem Konfigurierungsauftrag zusammengefaßt. Der Konfigurator liest den Konfigurierungsauftrag ein, interpretiert diesen und baut inhaltsabhängig die Struktur des Softwaresystems auf. Mit diesem Ansatz kann das Softwaresystem konfiguriert werden, ohne daß bei strukturellen Änderungen eine neue Laufzeitversion durch Einsatz zielsystemspezifischer Entwicklungswerkzeuge gebildet werden muß. Das Spektrum der realisierbaren Systemvarianten wird nur durch die Gesamtheit der Softwaremodule in der Modulbibliothek bestimmt.

5.3.2 Funktionalität des Konfigurators

Der Konfigurator wird nach dem Booten der Hardwareplattform als erstes Anwendermodul gestartet. Zu Beginn der Ausführung erzeugt der Konfigurator eine eigene Message-Queue zum Empfang von Bestätigungsdiensten anderer Softwaremodule. Im nächsten Schritt wird der Konfigurierungsauftrag eingelesen und interpretiert (Bild 30). Anhand der Angaben im Konfigurierungsauftrag lädt der Konfigurator die benötigten Softwaremodule aus der Modulbibliothek in den Hauptspeicher und startet deren jeweilige Initialisierungsfunktion.

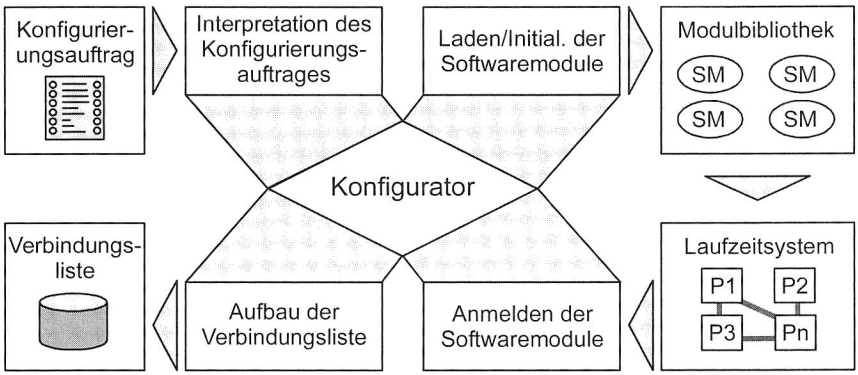


Bild 30: Funktionsübersicht zu dem Systemmodul Konfigurator

Die Konfiguration eines dynamischen Softwaresystems ist jedoch nicht auf die aufgabenspezifische Bereitstellung von Funktionalität in Form von Softwaremodulen beschränkt. Eine ebenso wichtige Rolle spielt der Aufbau von Kommunikationsverbindungen zum Datenaustausch zwischen den Softwaremodulen. Nach Kapitel 5.2 sind dazu Message-Queues, als speicherndes Transportsystem, besonders geeignet. Die Adressierung einer Message-Queue erfolgt durch einen Message-Queue-Identifier. Den modulspezifischen Identifier melden die Softwaremodule bei ihrer Initialisierung an den Konfigurator zurück. Mit dem systemweit eindeutigen Prozeßnamen eines Softwaremoduls aus dem Konfigurierungsauftrag und dem übertragenen Identifier

baut der Konfigurator eine Verbindungsliste auf. Auf Basis des Prozeßnamens kann ein Softwaremodul aus der Verbindungsliste den Identifier des gewünschten Prozesses ermitteln und mit diesem Daten austauschen. Die Verbindungsliste beschreibt damit die Kommunikationsstruktur des Softwaresystems.

Eine weitere Aufgabe des Konfigurators besteht darin, die Parametrierung der Softwaremodule zu initiieren. Mittels der Parametrierung werden sogenannte formale Parameter, die als Platzhalter in der Algorithmik der Softwaremodule vorgesehen sind, mit aktuellen Werten belegt.

Nach der Parametrierung der Softwaremodule sind diese ausführungsbereit und warten auf den Empfang des Aktivierungsdienstes. Der Konfigurationsvorgang ist damit beendet. Die letzte Aufgabe des Konfigurators besteht deshalb darin, das Modul Ablaufsteuerung zu starten. Danach beendet sich der Konfigurator selbst.

5.4 Realisierung einer Ablaufsteuerung

Die Notwendigkeit ein Modul zur Ablaufsteuerung vorzusehen ergibt sich aus dem Organisationsprinzip zur Zusammenarbeit von Prozessen. Die denkbaren Organisationsprinzipien und die daraus resultierende softwaretechnische Umsetzung der Ablaufsteuerung werden im folgenden beschrieben.

5.4.1 Organisationsprinzipien zur Ablaufsteuerung

Die Zusammenarbeit von Prozessen innerhalb eines Softwaresystems kann nach zwei unterschiedlichen Organisationsprinzipien erfolgen [70]: Dem Datenflußprinzip oder dem Client/Server-Prinzip (Bild 31).

Ist die Zusammenarbeit der Prozesse nach dem Datenflußprinzip organisiert, wird die Reihenfolge des gesamten Ablaufs durch einen fest definierten Fluß der Daten zwischen den Prozessen bestimmt. Das Abarbeiten des Berechnungsalgorithmus eines Prozesses hängt davon ab, ob alle von anderen Prozessen benötigten Daten in den entsprechenden Speicherplätzen bereitliegen. Der Prozeß erzeugt daraus neue Daten, die er wieder an andere Prozesse weitergibt. Dadurch ergibt sich beim Datenflußprinzip eine gute Übersicht über die Abläufe im Gesamtsystem. Allerdings werden durch die festgelegten Datenflüsse Erweiterungen bzw. Veränderungen des Prozeßsystems erschwert.

Die Funktionsweise des Client/Server-Prinzips wurde bereits erläutert. Mit dem Client/Server-Prinzip lassen sich offene Systemstrukturen einfacher realisieren, da bei der Integration neuer bzw. geänderter Funktionalitäten meist nur einzelne Serverprozesse ausgetauscht bzw. verändert werden müssen, während die meisten anderen Prozesse

unverändert bleiben. Daneben sind die Dienste bzw. Funktionen eines Servers wiederverwendbar, weil jeder beliebige Client die Dienste eines Servers benutzen kann.

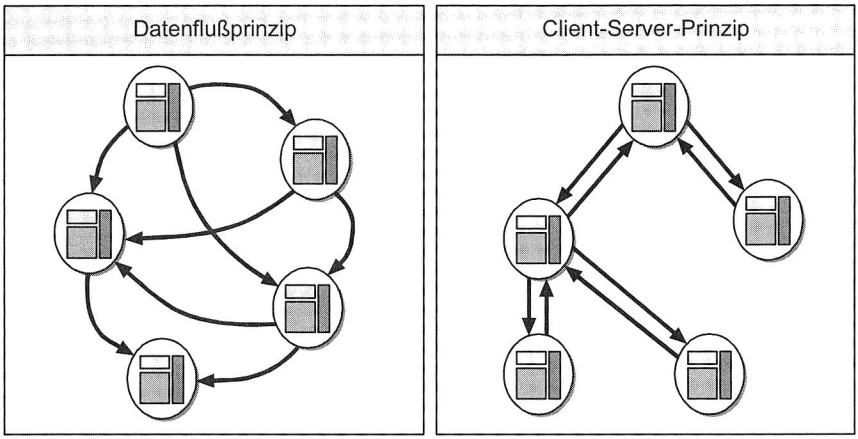


Bild 31: Organisationsprinzipien der Prozeßablaufmechanismen nach dem Datenflußprinzip und dem Client/Server-Prinzip

Im Gegensatz zum Datenflußprinzip ist die Reihenfolge des gesamten Ablaufs beim Client/Server-Prinzip a priori nicht vorgegeben. Ein Server kann beispielsweise von unterschiedlichen Clients zu beliebigen Zeitpunkten angesprochen werden. Um eine Abarbeitungsreihenfolge festzulegen, wird ein weiteres Modul, die sog. Ablaufsteuerung, benötigt. Darunter versteht man nach DIN 19237 [26] eine Steuerung mit einem zwangsläufigen schrittweisen Ablauf. Der Übergang von einem Schritt auf den oder die programmgemäß folgenden findet in Abhängigkeit von einer Übergangsbedingung statt.

Die Ablaufsteuerung aktiviert die Bearbeitungsroutinen der Prozesse in der erforderlichen Reihenfolge. Die Reihenfolge ist dadurch vorgegeben, daß ein Prozeß zur Durchführung seiner Bearbeitungsroutinen in der Regel auf Daten von anderen Prozessen angewiesen ist. Diese vorgelagerten Prozesse müssen zu einem früheren Zeitpunkt ausgeführt worden sein. Die Abarbeitungsreihenfolge ist vom Benutzer parametrierbar. Die erfolgreiche Beendigung der Bearbeitungsroutine eines Prozesses wird von diesem an die Ablaufsteuerung gemeldet. Dies stellt die Übergangsbedingung für die Aktivierung des folgenden Prozesses dar.

Der zusätzliche Aufwand, der mit der Realisierung einer Ablaufsteuerung verbunden ist, ist durch entscheidende Vorteile gerechtfertigt. Bei vielen Anwendungen ist die Abarbeitungsreihenfolge von einer wechselnden Betriebsart abhängig. Bei

Verwendung des Datenflußprinzips muß das betriebsartenabhängige Verhalten in jedem Prozeß implementiert werden. Änderungen sind äußerst aufwendig zu realisieren. Das Konzept der Ablaufsteuerung löst dieses Problem einfach durch die zentrale betriebsartenabhängige Aktivierung der Prozesse. Ein weiterer Aspekt ist die Berücksichtigung von sicherheitsrelevanten Funktionen. Tritt ein Fehler im Programmablauf auf, so muß die Abarbeitung der Softwaremodule sofort abgebrochen werden, um Maschinen- und Bedienergefahren zu vermeiden. Bei einem datenflußorientierten Programmablauf muß die Fehlerbehandlung ebenfalls in jedem einzelnen Modul implementiert werden. Bei einer zentralen Ablaufsteuerung kann dies zentral erfolgen. Änderungen und Erweiterungen sind dadurch leichter umzusetzen.

Eine wesentliche Eigenschaft von Softwaresystemen, speziell bei eingebetteten Hardwareplattformen, ist die Verarbeitung von Signalen aus der Geräteperipherie. Handelt es sich dabei um Maschinen, die über interne Regelkreise verfügen, so werden die Maschinendaten zyklisch bereitgestellt. Da die Funktionalität einiger Softwaremodule auf derartigen Daten beruhen, ist es erforderlich die Ablaufsteuerung mit der Zykluszeit der Führungsmaschine zu synchronisieren. Dies erfolgt am geeignetsten über Interrupts. Da die Kopplung einer eingebetteten Hardwareplattform mit der Führungsmaschine meist über einen gemeinsamen Speicherbereich (Dual Ported Memory, DPM) erfolgt, ist eine Interruptauslösung auf der eingebetteten Hardwareplattform durch die Steuerung der Führungsmaschine problemlos möglich. Es ist jedoch darauf zu achten, daß die Zykluszeit der Ablaufsteuerung geringer ist, als die Zykluszeit der Führungsmaschine. Die Skalierbarkeit der Hardwareplattform ist in diesem Zusammenhang ein entscheidendes Kriterium.

5.4.2 Realisierung des Moduls zur Ablaufsteuerung

Die konkrete Realisierung der Ablaufsteuerung im Rahmen des realisierten Softwaresystems führte zu zwei grundsätzlichen Teilbereichen. Der erste Teilbereich betrifft die Initialisierung des Moduls. Hierbei wird das Ablaufprogramm eingelesen und verarbeitet (Bild 32). Es enthält im wesentlichen die Prozeßnamen in ihrer Ausführungsreihenfolge und die Betriebsarten, in denen die jeweiligen Prozesse ausgeführt werden sollen.

Nach der Verarbeitung des Ablaufprogramms wird die Task zur Ablaufsteuerung gestartet. Diese bildet den zweiten funktionalen Teilbereich dieses Softwaremoduls. Die Ablaufsteuerung in der Task wird durch einen Zustandsautomaten [106] realisiert. Ein Zustand repräsentiert einen genau abgegrenzten Zeitabschnitt, in dem ein System ein bestimmtes Verhalten zeigt. Ein Zustandsübergang ist ein Zeitpunkt, an dem ein System einen gegebenen Zustand verläßt und in einen neuen Zustand übergeht. Die Aufgabe eines Zustandsautomaten besteht darin, in Abhängigkeit von internen bzw. externen Ereignissen einen Zustandsübergang zu veranlassen. Interne Ereignisse

sind z.B. der erfolgreiche Abschluß einer Teilfunktionalität. Externe Ereignisse sind z.B. Nachrichten, die über den moduleigenen Kommunikationskanal empfangen werden.

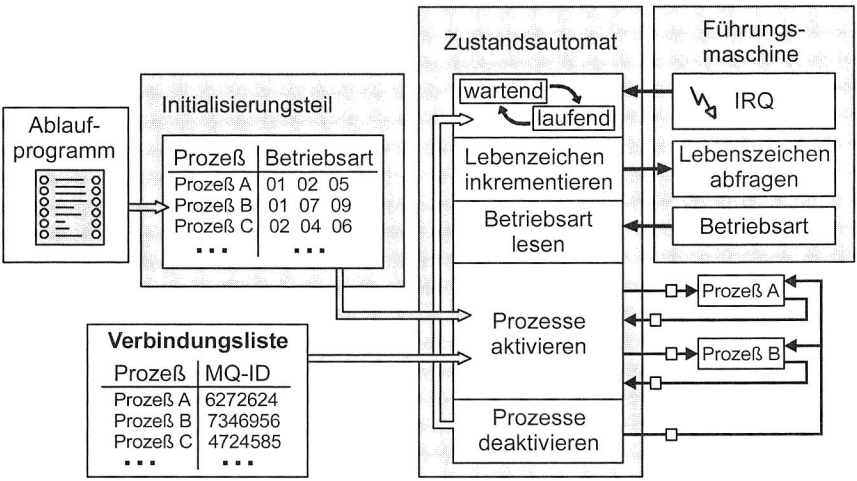


Bild 32: Struktureller Aufbau des Softwaremoduls zur Ablaufsteuerung

Im Grundzustand befindet sich die Task im Betriebszustand "wartend". Sie belegt dabei keine Rechenzeit. Durch den Interruptaufruf der Führungsmaschine geht die Task in den Zustand "laufend" über. Dies bedeutet, daß die Task Rechenzeit zugeordnet bekommt. Sie geht daraufhin in den nächsten Zustand des Softwaresystems über. In diesem Zustand wird eine Variable inkrementiert, die als Lebenszeichen bezeichnet wird. Diese Variable ist von der Führungsmaschine aus lesbar. Sie dient dazu die Funktionsfähigkeit des Softwaresystems von Seiten der Führungsmaschine zu überprüfen. Wird das Lebenszeichen durch das Modul Ablaufsteuerung nicht inkrementiert, so stoppt die Führungsmaschine den Bearbeitungsprozeß.

Im folgenden Zustand wird die aktuelle Betriebsart von der Führungsmaschine ausgelesen. In Abhängigkeit von dieser werden anschließend die einzelnen Prozesse aktiviert. Dazu sucht die Ablaufsteuerung in der Verbindungsliste die Message-Queue-ID zu den im Ablaufprogramm festgelegten Prozessen und sendet diesen den Dienst Aktivierungsaufforderung. Bei Empfang dieses Ereignisses gehen die Prozesse von dem Betriebszustand "wartend" in den Betriebszustand "laufend" über. Am Ende eines Aktivierungszykluses werden alle Prozesse wieder deaktiviert.

5.5 Struktur der anwendungsspezifischen Softwaremodule

Softwaremodule verfügen im wesentlichen über zwei Gruppen von Eigenschaften. Zum einen die invarianten Eigenschaften, die in jedem Modul realisiert sind. Sie sind die Grundlage einer einheitlichen Referenzarchitektur, welche die Integration neuer Softwaremodule und ihre Kombination mit bereits vorhandenen Einheiten ermöglicht. Beispielhaft können hier sowohl Mechanismen für die Instanzierung und Parametrierung als auch einheitliche Kommunikationsmechanismen für den Datenaustausch genannt werden. Diese Eigenschaften müssen nur einmal festgelegt werden. Sie werden dann in allen Modulen umgesetzt.

Die zweite Gruppe betrifft die funktionalen Eigenschaften eines Softwaremoduls. Diese Eigenschaften sind spezifisch für ein Modul und von der umzusetzenden Funktionalität abhängig. Der Zugriff auf die Funktionalitäten erfolgt über Dienste. Im folgenden wird die Umsetzung der ersten Gruppe von Eigenschaften für die anwendungsspezifischen Softwaremodule näher erläutert.

5.5.1 Grundkonzeption der Softwaremodule

Die Erstellung von Software läßt sich wesentlich vereinfachen, wenn die Realisierung der Programme systematisiert und eine Orientierung an bereits bestehende Referenzprogramme ermöglicht wird. Deshalb weisen alle anwendungsspezifischen Softwaremodule von ihrer Grundkonzeption her folgende Kennzeichen auf (Bild 33):

- Zustandsgesteuerte, zyklische Arbeitsweise
- Interprozeßkommunikation mit anderen Modulen
- Server- und Clienteigenschaften
- systemweit eindeutiger Prozeßname.

Im Gegensatz zu ereignisgesteuerten dispositiven Softwaresystemen hat sich in der Steuerungstechnik eine zyklische Abarbeitung der Steuerungsfunktionalitäten bewährt. Dies ist darauf zurückzuführen, daß ein steuerungstechnischer Prozeß kontinuierlich mit Steuerdaten versorgt werden muß.

Der Ablauf innerhalb eines komplexen Softwaremoduls ist selten rein sequentiell. In Abhängigkeit vom jeweiligen Prozeßzustand werden unterschiedliche Funktionalitäten ausgeführt. Eine zustandsabhängige Ablaufsteuerung innerhalb eines Softwaremoduls ist deshalb vorteilhaft. Diese Funktion übernimmt ein sog. Zustandsautomat, der bereits im Modul zur Ablaufsteuerung vorgestellt wurde.

Ein weiteres Merkmal der anwendungsspezifischen Softwaremodule ergibt sich aus der funktionalen Dekomposition der Steuerungsfunktionalität. Diese erfordert den Datenaustausch zwischen einzelnen Softwaremodulen, da die Durchführung der modulspezifischen Funktionalität in der Regel Daten aus anderen Modulen erfordert.

Der Datenaustausch basiert auf den vom Betriebssystem bereitgestellten Mechanismen zur Interprozeßkommunikation, wie sie in Kapitel 5.2 bereits vorgestellt wurden. In einer dynamischen Systemumgebung, in der die Art und Anzahl der verwendeten Softwaremodule nicht bereits im Quellcode festgelegt wird, ist deren Adressierung durch einen systemweit eindeutigen Prozeßnamen unbedingt erforderlich.

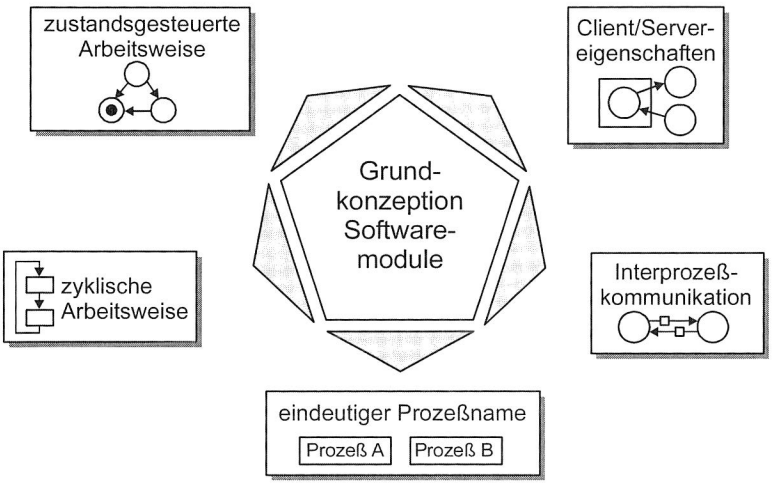


Bild 33: Bausteine der Grundkonzeption von Softwaremodulen

Da ein Softwaremodul sowohl Daten von anderen Modulen empfängt und verarbeitet und das Verarbeitungsergebnis wiederum anderen Modulen zur Verfügung stellt, muß ein Softwaremodul sowohl über Clienteigenschaften als auch über Servereigenschaften verfügen.

5.5.2 Programmiertechnische Umsetzung

Ein anwendungsspezifisches Softwaremodul besteht aus einer Initialisierungsfunktion und einer Hauptschleife. Die Hauptschleife wird dabei von einer eigenständigen Task gebildet.

Nach dem Laden des Objektfiles eines Softwaremoduls durch den Konfigurator, startet dieser die Initialisierungsfunktion des Softwaremoduls (Bild 34). Die Initialisierungsfunktion initialisiert zuerst die eigenen modulglobalen Variablen und richtet dann eine modulspezifische Message-Queue ein. Anschließend meldet sich das Modul über die globale Message-Queue des Konfigurators bei diesem an. Der Konfigurator sendet daraufhin die Dienstaufforderung zum Starten der Parametrierung. Bei der Parame-

trierung ließt das Softwaremodul die Parameterwerte aus dem modulspezifischen Parameterfile und weißt diese den entsprechenden Variablen zu. Nach Abschluß der Parametrierung sendet das Modul eine entsprechende Meldung an den Konfigurator. In einem letzten Schritt wird der eigentliche Modulprozeß als eigenständige Task gestartet. Die Initialisierungsfunktion beendet sich daraufhin.

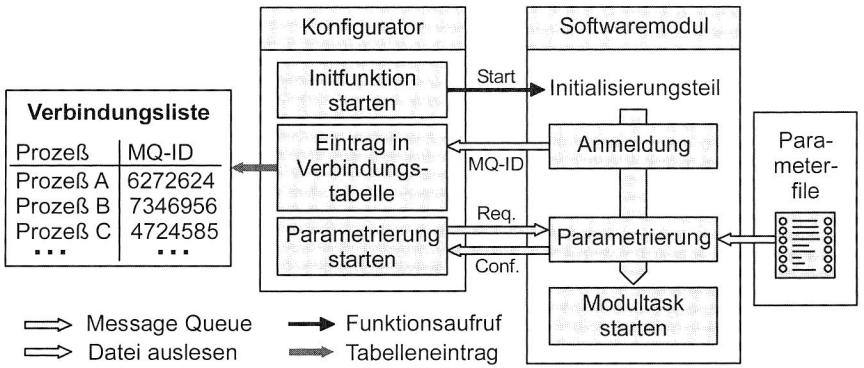


Bild 34: Struktureller Aufbau der anwendungsspezifischen Softwaremodule

Die Hauptschleife eines anwendungsspezifischen Softwaremoduls ist als Endlosschleife realisiert. Einmal gestartet, kann ein Softwaremodul nur durch Herunterfahren des gesamten Softwaresystems terminiert werden. Das Terminieren und Neustarten von Softwaremodulen während der Lebenszeit des Softwaresystems ist nicht vorgesehen, da es für Sensoranwendungen nicht benötigt wird.

Innerhalb der Hauptschleife übernimmt ein Zustandsautomat die Ablaufkontrolle. Dieser wird durch die Ablaufsteuerung zyklisch aktiviert. Während eines Ablaufzykluses werden verschiedene Teilbereiche der Modulfunktionalität abgearbeitet. Dazu werden meist Berechnungsergebnisse und spezifische Steuerdaten aus anderen Softwaremodulen benötigt. Im Verständnis der Client/Server-basierten Kommunikationsstruktur fungiert das datenanfordernde Softwaremodul als Client. Hat ein Softwaremodul seine Funktionalität abgearbeitet, so geht es in einen Servermodus über. Dies bedeutet, daß es auf Anfragen von anderen Modulen wartet, die ihrerseits modulspezifische Daten benötigen. Ist die Abarbeitung aller Softwaremodule, und damit ein Ablaufzyklus, beendet, so werden alle Softwaremodule, durch die Ablaufsteuerung, wieder in den Betriebszustand "wartend" versetzt.

Zur Unterstützung der Implementierungsphase neuer Softwaremodule wird die allgemeine Grundstruktur als Programmgerüst vorgegeben. Konkret realisiert das Programmgerüst bereits die Modulinitialisierungsroutinen, den Zustandsautomaten,

die Client/Server-Funktionalität und die Kommunikationsmechanismen. Der Programmierer kann dann die anwendungsspezifische Funktionalität in das Programmgerüst einfach integrieren. Damit wird der Softwareentwicklungsprozeß beschleunigt und die Fehlerwahrscheinlichkeit reduziert.

5.6 Entwicklung eines Konfigurier- und Parametriertools

Sowohl die Systemmodule als auch die anwendungsspezifischen Softwaremodule verfügen über eine stark abstrahierte Funktionalität, um den späteren Verwendungszweck nicht von vornherein einzuschränken. Die Anpassung an die konkrete Anwendung erfolgt über Parametrierfiles. Diese werden in der Initialisierungsphase jedes Moduls eingelesen und verarbeitet bzw. interpretiert. Um sie manuell editieren zu können, wurde das standardisierte ASCII-Format verwendet.

Der manuelle Editiervorgang ist jedoch äußerst fehleranfällig und unkomfortabel. Um die Fehlereinflüsse zu verringern und die Akzeptanz bei dem Bediener zu erhöhen, wurde ein Konfigurier- und Parametriertool, auf Basis der grafischen Bedienoberfläche von Microsoft Windows, entwickelt. Mit Hilfe dieses Tools werden der Konfigurierungsauftrag für den Konfigurator, das Ablaufprogramm für die Ablaufsteuerung und die Parameterfiles der restlichen Softwaremodule generiert.

Wie bereits mehrfach erwähnt, hängt bei einem dynamisch konfigurierten Softwaresystem die Art und Anzahl der verwendeten Softwaremodule von der jeweiligen Anwendung ab. Das Konfigurier- und Parametriertool muß deshalb so ausgelegt werden, daß es dynamische Anpassungen der Softwarestruktur abbilden kann. Die moderne ActiveX-Technologie von Microsoft ermöglicht es, die Anforderungen zu erfüllen.

Die Hauptkomponente des Konfigurier- und Parametriertools bildet der sog. Konfigurations-Server (Bild 35). Dabei handelt es sich um einen ActiveX-EXE-Server, der als Out-of-Process-Server realisiert wurde. Er besitzt eine öffentliche Klasse, von der Clients eine Instanz erzeugen können. Über dieses Objekt kann jeder Client benötigte Konfigurationsdaten vom Server beziehen. Ein Client ist dabei ein ausführbares Programm, das über eine grafische Bedienoberfläche verfügt und mit dem Server, über den vorgestellten Mechanismus, kommunizieren kann.

Für jedes anwendungsspezifische Softwaremodul existiert ein Konfigurations-Client. Dessen Aufgabe besteht darin, die Parametrierung des jeweiligen Softwaremoduls zu ermöglichen und aus dem Parameterdatensatz ein Parametrierfile automatisch zu generieren. Bei der Festlegung der Parameter wird weitgehend auf manuelle Eingaben verzichtet. Der Bediener muß lediglich Werte aus Listen auswählen oder mit Hilfe von Schiebereglern einstellen. So sind beispielsweise Prozesse, mit denen ein Softwaremodul kommunizieren will, nur aus einer Liste von Prozesse auswählbar, die bereits in

die Konfiguration aufgenommen wurden. Zusätzlich führt jeder Konfigurations-Client eine Plausibilitätsanalyse durch, bevor ein Parametrierfile erzeugt wird.

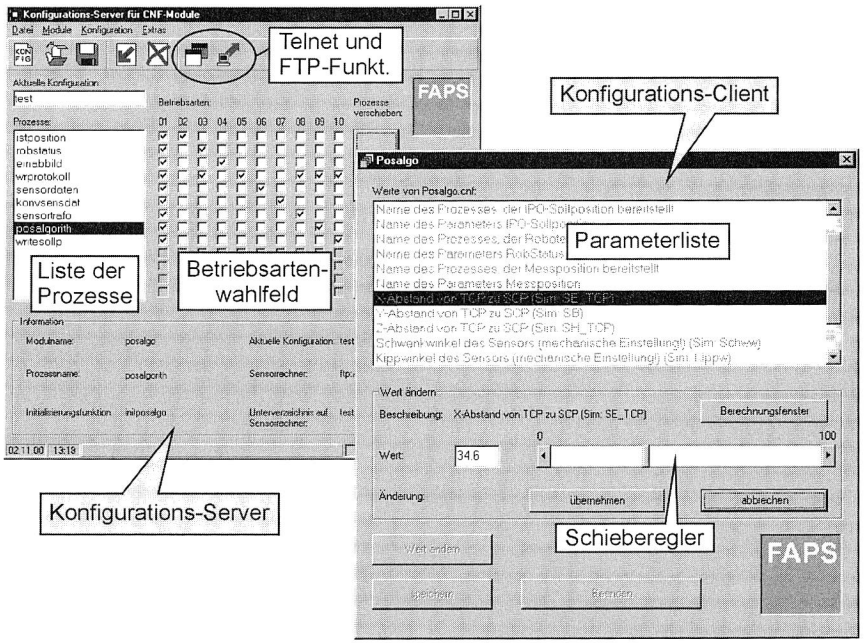


Bild 35: Bedienoberflächen des Konfigurations-Servers und eines beispielhaften Konfigurations-Clients

Der Konfigurations-Server verwaltet die gesamte Konfiguration und erstellt daraus automatisch den Konfigurierungsauftrag und das Ablaufprogramm. In der Oberfläche des Konfigurations-Servers können neue Softwaremodule zur Konfiguration hinzugefügt werden, bearbeitet werden und entfernt werden. Wurde eine Konfiguration fertiggestellt, so bietet der Konfigurations-Server die Möglichkeit die erzeugten ASCII-Dateien auf das Softwaresystem zu übertragen. Dazu wird eine Ethernetverbindung zur Hardwareplattform genutzt, über die ein FTP-Transfer abgewickelt wird.

Weiterhin kann der Konfigurator, nach einer Neukonfiguration des Softwaresystems, von der Bedienoberfläche geladen und gestartet werden. Dazu wird eine Telnet-Verbindung zur Hardwareplattform aufgebaut. Über die Telnet-Verbindung können sämtliche Ausgaben auf dem Konfigurationsrechner dargestellt werden. Dies ist vor allem in der Inbetriebnahmephase sehr hilfreich, da ein Embedded System meist über keine eigenen Ausgabemöglichkeiten verfügt.

6 Spezifikation der Softwaremodule zur sensorgestützten Roboterführung

Zum Aufbau eines konfigurierbaren Softwaresystems für eingebettete Systeme sind die Softwaremodule, die in Kapitel 5 ausführlich diskutiert wurden, unbedingt erforderlich. Diese Module sind anwendungsneutral. Die Umsetzung anwendungsspezifischer Anforderungen erfolgt durch eine weitere Klasse von Softwaremodulen. Dieses Kapitel beschreibt eine Reihe von Modulen, die für die Realisierung eines Softwaresystems zur sensorgestützten Roboterführung erforderlich sind.

6.1 Hardwareabhängige Softwaremodule

Eine wesentliche Eigenschaft offener Softwaresysteme ist die interne Verwendung standardisierter Schnittstellen zwischen den Softwaremodulen. Damit können einheitliche Kommunikationsmechanismen für den Datenaustausch zwischen den Softwaremodulen realisiert werden.

Jedes Softwaresystem muß jedoch auch Schnittstellen für den Datenaustausch mit seiner Systemumgebung bereitstellen. Im Großrechner- und Office-Bereich laufen häufig mehrere Softwaresysteme auf einer Hardwareplattform oder im Netzwerkverbund. Der Datenaustausch zwischen diesen Systemen erfolgt zunehmend über standardisierte Softwareschnittstellen, wie sie z.B. das Betriebssystem Microsoft Windows durch die COM/DCOM-Architektur bereitstellt [32].

Im Gegensatz dazu, steht bei den sog. eingebetteten Systemen vor allem der Datenaustausch mit anderen Hardwarekomponenten des Gesamtsystems im Mittelpunkt. Ein System zur Sensordatenintegration- und verarbeitung für Robotersteuerungen benötigt Schnittstellen zu den angeschlossenen Sensoren, zu der Geräteperipherie und zur Robotersteuerung. Eine Standardisierung dieser Schnittstellen konnte bisher nur hardwareseitig, durch die Einführung der Feldbustechnologie, erreicht werden. Klassische Softwaresysteme bedienen die nicht standardisierten Schnittstellen zur Systemperipherie softwaretechnisch deshalb über sogenannte Gerätetreiber. Ein derartiger Gerätetreiber ist darauf spezialisiert den Datenaustausch mit einem speziellen Gerät zu ermöglichen. Dies umfaßt sowohl die Steuerung der hardwaretechnischen, physikalischen Schnittstelle zum Sensor als auch der softwaretechnischen, logischen Datenschnittstelle. Dieser Lösungsweg ist dann sinnvoll, wenn bezüglich der Schnittstellen einer Gruppe funktional ähnlicher Peripheriegeräte keine relevanten Gemeinsamkeiten bestehen.

Verfügt eine Gruppe funktional ähnlicher Peripheriegeräte jedoch über vergleichbare Datenschnittstellen, so besteht die Möglichkeit den Datenaustausch mit diesen

Peripheriegeräten von einem parametrierbaren Softwaremodul durchführen zu lassen. Die gerätespezifischen Unterschiede werden durch Parametrierung berücksichtigt. Dieser Lösungsweg hat den wesentlichen Vorteil, daß für ein funktional ähnliches Gerät, für das bereits ein parametrierbares Schnittstellenmodul existiert, kein neuer Treiber entwickelt werden muß. Es ist lediglich eine neue Parametrierung erforderlich. Der Engineeringaufwand für die Realisierung des Datenaustausches mit einem neuen Peripheriegerät wird damit deutlich reduziert. Bei der Realisierung der folgenden hardwareabhängigen Softwaremodule wurde der jeweils günstigere Lösungsweg genutzt.

6.1.1 Einlesen der Sensorrohdaten

Sensoren für den Einsatz in Roboterzellen sind ein Beispiel für funktional ähnliche Peripheriegeräte. Trotz der Vielfalt an unterschiedlichen Meßprinzipien und Meßgrößen verfügen Sensoren im Hinblick auf die softwaretechnische Datenschnittstelle über wesentliche Gemeinsamkeiten. Die Struktur der sensorseitig bereitgestellten Daten ist für die überwiegende Anzahl der Sensoren identisch. Nahezu jeder der in Frage kommenden Sensoren liefert neben den eigentlichen Meßwerten auch Zustandsinformationen über die Gültigkeit der Meßwerte bzw. das Auftreten von Meßfehlern. Je nach Meßprinzip unterscheiden sich die Sensoren in der Anzahl der Meßwerte (Dimensionalität) und der Anzahl der Zustandsinformationen.

Die einzelnen Sensordaten werden sensorseitig häufig über analoge Schnittstellen ausgegeben. Das Eingabemodul eines Feldbusses liest diese Schnittstellen aus, digitalisiert die analogen Signale und überträgt die Sensordaten an das Feldbusmodul des Sensorrechners. Dort werden die Sensordaten im sog. Eingangsabbild abgelegt. Das Eingangsabbild stellt einen Speicherbereich auf der Feldbushardware des Sensorrechners dar, auf den vom Sensorrechner aus zugegriffen werden kann.

Zustandsinformationen werden meist über digitale Schnittstellen ausgegeben. Nach dem Einlesen und Übertragen der Daten über den Feldbus stehen die Zustandsinformationen ebenfalls im Eingangsabbild zur Verfügung. Bisher verfügen erst wenige Sensoren über eine eigene Feldbusschnittstelle. Der Trend geht jedoch eindeutig in diese Richtung. Derartige Sensoren übertragen ihre Meßwerte und Zustandsinformationen direkt an das Feldbusmodul des Sensorrechners. Das Einlesen der Sensordaten über spezielle Eingabemodule entfällt damit. Die Zuordnung zwischen Sensordatum und Adresse im Eingangsabbild wird bei der Parametrierung des Feldbussystems festgelegt.

Die Schnittstelle zu dem Feldbusmodul auf dem Sensorrechner wird durch ein Softwaremodul mit der Bezeichnung `leseEA` (lese Eingangsabbild) realisiert. Die Aufgaben diese Softwaremoduls bestehen darin, die Feldbushardware zu initialisieren, den Datenaustausch zu steuern und zu überwachen und die Daten aus dem Eingangs-

abbild auszulesen und anderen Softwaremodulen in einheitlicher Form zur Verfügung zu stellen (Bild 36).

Die Initialisierungs- und Zugriffsroutinen sind in Abhängigkeit von der eingesetzten Feldbushardware sehr unterschiedlich. Dies würde die Parametrierung eines Standardmoduls sehr aufwendig gestalten. Es ist deshalb sinnvoll für die Integration der Feldbushardware in das Softwaresystem spezialisierte Treibermodule bereit zu stellen. Je nach verwendeter Feldbushardware wird dann bei der Konfiguration des Softwaresystems das geeignete Treibermodul ausgewählt. Die Anpassung des Treibermoduls an die angeschlossene Sensorik erfolgt hingegen durch Parametrierung.

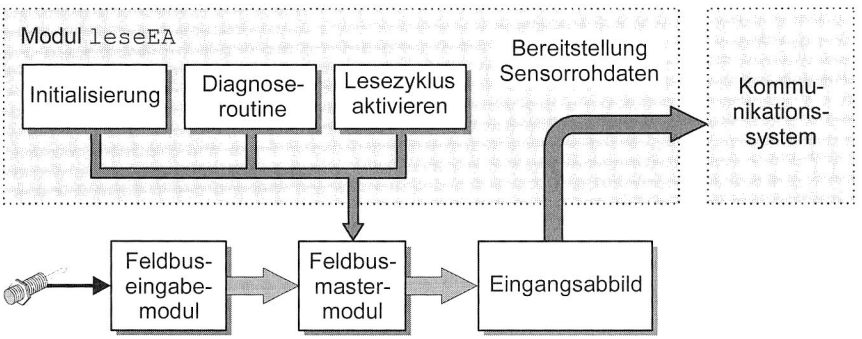


Bild 36: Struktureller Aufbau des Softwaremoduls zum Einlesen der Sensorrohdaten

Das Eingangsabbild enthält nur die von den Sensoren übertragenen Rohdaten. Das Softwaremodul `leseEA` stellt den Datenblock mit den Sensorrohdaten den anderen Softwaremodulen zur Weiterverarbeitung zur Verfügung.

Sensorrohdaten einlesen über serielle Schnittstelle

Viele intelligente Sensoren benutzen eine serielle Schnittstelle zur Datenübertragung. Eine serielle Schnittstelle stellt eine Standardschnittstelle dar, die von nahezu allen Steuerungen unterstützt wird. Die Feldbustechnik stellt auch für serielle Schnittstellen ein Eingabemodul zur Verfügung. Es ist jedoch zu berücksichtigen, daß bei derartigen Schnittstellen die Übertragungssicherheit durch Protokolle sichergestellt wird. Die Sensordaten werden in den oben beschriebenen Formaten in einen Protokollrahmen eingebettet. Dies bedeutet, daß im Eingangsabbild neben den Sensordaten auch die Protokolldaten abgelegt werden. Die Sensordaten müssen deshalb zuerst aus dem Protokollrahmen gelöst werden, bevor sie von anderen Softwaremodulen ausgewertet werden können. Diese Aufgabe übernimmt das Modul `serielleEinl` (serielle Daten einlesen).

Die physikalische Datenübertragung über serielle Schnittstellen ist in der DIN 66259 [31] festgelegt. Man unterscheidet zwei physikalische Schnittstellen: RS-232 und RS-485. Die Unterschiede liegen darin, daß bei RS-232 die Kommunikation auf zwei Teilnehmer pro Schnittstelle beschränkt ist, während bei Nutzung von RS-485 ein Bussystem aufgebaut werden kann. Auf Seiten der logischen Schnittstelle existiert allerdings keinerlei Normierung. Die Daten können mit oder auch ohne Nutzung eines Protokolls ausgetauscht werden. Basiert der Datenaustausch mit einem Sensor auf einem Protokoll, so ist dieses herstellerspezifisch und kann deshalb theoretisch beliebig aufgebaut sein. Aus diesem Grund ist es notwendig für jeden Sensor, der über ein serielles Protokoll kommuniziert, ein Treibermodul zu entwickeln und in das Softwaresystem zu integrieren.

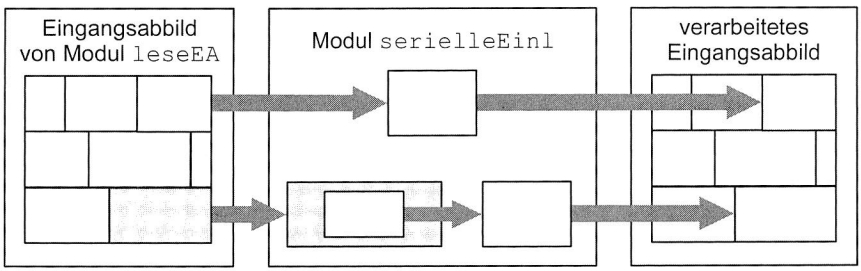


Bild 37: Verarbeitung des Eingangsabbildes durch das Modul `serielleEinl`

Das Modul `serielleEinl` liest den Datenblock mit den Sensorrohdaten aller Sensoren, von dem Modul `leseEA`. Aus dem gelesenen Datenblock müssen im nächsten Schritt alle von einer einzelnen seriellen Schnittstelle stammenden Daten entnommen werden (Bild 37). Die Nutzdaten sind jedoch noch in ein herstellerspezifisches Protokoll eingebettet. Ein auf dieses Protokoll spezialisierter Algorithmus des Softwaremoduls entnimmt die Nutzdaten aus den seriell übertragenen Daten. Diese werden anschließend zu einem neuen Datenblock zusammengefaßt, der für weiterverarbeitende Softwaremodule zur Verfügung steht. Die Struktur dieses Datenblockes ist identisch mit derjenigen, die das Modul `leseEA` an seiner Schnittstelle bereitstellt. Damit ist es für weiterverarbeitende Softwaremodule ohne Bedeutung, ob die Sensorrohdaten direkt aus dem Eingangsabbild stammen, oder einem seriell übertragenen Datenblock entnommen wurden.

6.1.2 Ausgabe der Aktorsteuerdaten

Die Kernaufgabe eines jeden Steuerungssystems besteht darin, einen technischen Prozeß gemäß einer vorgegebenen Aufgabenstellung zu steuern. Dazu bedient sich das Steuerungssystem sogenannter Aktoren, die es erlauben den Ablauf des techni-

schen Prozesses in gewünschter Weise zu beeinflussen. In einer roboterbasierten Produktionszelle sind der Roboter und das von diesem geführte Werkzeug die beiden wichtigsten Aktoren. Durch die exakte Positionierung des Werkzeugs wird die Prozeßqualität bei Montage- und Materialbearbeitungsaufgaben wesentlich beeinflußt. Steuergrößen von Werkzeugen sind z.B. die Leistung eines Materialbearbeitungslasers oder das Ausbringungsvolumen einer Dosierpistole. Häufig sind an dem Werkzeug noch zusätzliche Aktoren angebracht, die am Werkzeugeingriffspunkt Hilfsstoffe bereitstellen, ohne die der technische Prozeß nicht realisierbar wäre. Ein Beispiel für einen derartigen Aktor ist eine Drahtzuführeinrichtung beim Löten.

Vergleichbar den Sensoren verfügen auch Aktoren über unterschiedliche physikalische Schnittstellen. Auch hier bietet sich die Ansteuerung über die einheitliche Feldbusschnittstelle an. Moderne Aktoren sind bereits mit einer Feldbusschnittstelle ausgerüstet. Andere Aktoren verfügen über analoge, binäre, digitale oder serielle Schnittstellen. Diese können jedoch von geeigneten Ausgabemodulen des Feldbusses bedient werden. Die Übertragung der Steuerdaten an die Aktoren wird durch das Softwaremodul `schreibeAA` (Schreibe Ausgangsabbild) ermöglicht. Die Aufgaben dieses Softwaremoduls liegen darin, die von anderen Softwaremodulen übertragenen Aktorsteuerdaten zu analysieren und in das Ausgangsabbild zu übernehmen (Bild 38). Die Beschreibung der Aktordaten erfolgt in einer festgelegten, systemeinheitlichen Form. Außerdem wird von diesem Softwaremodul die Datenübertragung an die Aktoren gesteuert und überwacht.

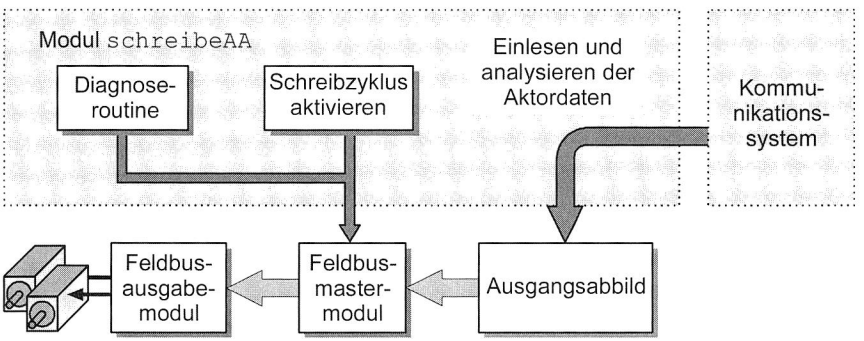


Bild 38: Struktureller Aufbau des Softwaremoduls zum Ansteuern der Aktoren

Liefert der Aktor selbst Meßwerte oder Zustandsgrößen, die für die Steuerung von Bedeutung sind, so werden diese Werte als Sensorwerte betrachtet und von dem Softwaremodul `leseEA` eingelesen.

Aktorsteuerdaten ausgeben über serielle Schnittstelle

Vergleichbar zu den Sensoren, gibt es auch Aktoren, die nur über eine serielle Schnittstelle zur Datenübertragung verfügen. Dies erfordert ein Softwaremodul, welches die Steuerdaten für den Aktor in dessen herstellerspezifisches Protokoll einbindet. Das Softwaremodul `serielleAusg` (serielle Ausgabe) übernimmt diese Aufgabe. Da ein herstellerspezifisches Protokoll beliebig strukturiert sein kann, muß dieses Softwaremodul als Treibermodul aufgebaut werden.

Die Steuerdaten für einen Aktor werden als Datenblock von einem entsprechenden Softwaremodul bereit gestellt. Ein auf das benötigte, herstellerspezifische Protokoll spezialisierter Algorithmus bindet die Steuerdaten in den Protokollrahmen ein. Das Ergebnis ist ein neuer Datenblock, der an das Modul `schreibeAA` übertragen werden soll. Für das Modul `schreibeAA` ist es dabei ohne Bedeutung, daß der Datenblock Protokollinformationen für eine serielle Schnittstelle enthält. Das Feldbus-Ausgabemodul für die serielle Schnittstelle interpretiert diese Daten und steuert damit die serielle Verbindung zum Sensorsystem.

6.1.3 Datenaustausch mit der Robotersteuerung

Eine Robotersteuerung kann sowohl als Sensor (z.B. Bereitstellung der Istpose) als auch als Aktor (z.B. Vorgabe der Sollpose) verstanden werden. Im Gegensatz zu Sensoren und Aktoren, bestehen zwischen Robotersteuerungen herstellerspezifische Unterschiede bezüglich des Umfangs der bereitgestellten Daten und der Manipulationsmöglichkeiten auf interne Steuerdaten des Roboters. Deshalb ist es heute noch sinnvoll Softwaremodule für den Datenaustausch mit der Robotersteuerung als Treiber und nicht in rein parametrierbarer Form zu realisieren. Ein Treiber hat den Vorteil, daß er an den jeweiligen Grad der Offenheit einer Robotersteuerung optimal angepaßt werden kann.

Für den Datenaustausch mit der Robotersteuerung wurden drei Softwaremodule implementiert. Diese sollen im folgenden vorgestellt werden (Bild 39).

Softwaremodul zum Lesen der aktuellen Position und Orientierung

Mechanische Drehgeber, an den Getrieben der Roboterachsen, bestimmen die Stellung jeder Achse. Mit Hilfe einer Vorwärtstransformation, die die Kinematik des Roboters abbildet, bestimmt die Robotersteuerung, aus den Achsstellungen, die kartesische Position und Orientierung des Werkzeugeingriffspunktes. Dies geschieht unmittelbar vor Ausführung der roboterinternen Softwarefunktionen, die den Datenaustausch mit dem Sensorrechner steuern. Auf Seiten des Sensorrechners übernimmt das Softwaremodul `leseIstpose` den Datenaustausch mit dem Roboter.

Nach dem erfolgreichen Auslesen der Istpose, stellt das Softwaremodul die Datenstruktur der Istpose den anderen Modulen zur Verfügung.

Softwaremodul zum Lesen von Systemdaten, Statusinformationen und Variablen

Ein weiteres Softwaremodul, mit der Bezeichnung `leseRobStatus`, liest von der Robotersteuerung bereitgestellte Systemdaten, Statusinformationen des Datenaustauschmoduls der Robotersteuerung und Variablen des Roboterprogramms. Unter Systemdaten versteht man interne Steuerungsdaten der Robotersteuerung, die für den Ablauf der Systemsoftware der Robotersteuerung benötigt werden. Es handelt sich dabei z.B. um die programmierte Bahngeschwindigkeit oder die festgelegte zeitliche Dauer eines Interpolationstaktes. Diese Daten sind vor allem für das Bahnkorrekturmodul von Bedeutung.

Eine weitere wichtige Information stellt der Status des Datenaustauschmoduls der Robotersteuerung dar. Dessen Status gibt an, ob der Datenaustausch von Seiten der Robotersteuerung im aktuellen Interpolationstakt aktiviert wurde, ob der Datenaustausch fehlerfrei abläuft oder ob der Datenaustausch nach dem aktuellen Interpolationstakt von der Robotersteuerung beendet wird. Auf Seiten des Softwaresystems zur Sensordatenverarbeitung haben diese Zustandsinformationen einen wesentlichen Einfluß auf die Verarbeitungsalgorithmen in einigen Softwaremodulen.

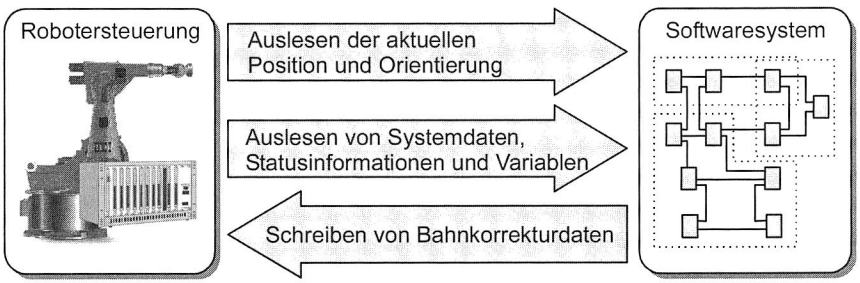


Bild 39: Datenaustausch mit der Robotersteuerung

Über die dritte Datengruppe besteht die Möglichkeit Variablen aus dem Roboterprogramm an das Softwaresystem zur Sensordatenverarbeitung zu übertragen. Diese Variablen werden vom Anwender im Roboterprogramm an geeigneter Stelle mit Werten belegt. In jedem Interpolationstakt werden diese Werte von der Robotersteuerung ausgelesen und in den Dual-Ported-Memory übertragen. Einer dieser Variablen kommt eine besondere Bedeutung zu. Es handelt sich dabei um die Festlegung einer Betriebsart. Diese Variable wird von dem Softwaremodul Ablauf-

steuerung benötigt. Die Betriebsart legt fest, welche Softwaremodule ausgeführt werden, solange diese Betriebsart gültig ist. Sie nimmt damit direkten Einfluß auf die Arbeitsweise der Ablaufsteuerung. Sinnvoll ist eine Betriebsartenvorgabe bei Prozessen, die schrittweise ablaufen. Ein typisches Beispiel hierfür ist eine Nahtfolgeanwendung, bei der man zwei Betriebsarten unterscheiden muß. Ist die erste Betriebsart angewählt, so werden die Softwaremodule aktiviert, die eine sensorgestützte Bestimmung des Nahtanfanges ermöglichen. Nach erfolgreicher Bestimmung des Nahtanfanges und geeigneter Positionierung des Roboters wird die zweite Betriebsart angewählt. In der zweiten Betriebsart werden die Softwaremodule aktiviert, die den Roboter entlang der sensorisch erfaßten Naht führen.

Darüber hinaus können noch weitere Variablen aus dem Roboterprogramm übertragen werden. Die Belegung dieser Variablen steht dem Anwender frei. Über diese Variablen kann vom Roboterprogramm aus Einfluß auf die Arbeitsweise einzelner Softwaremodule genommen werden. Die korrespondierenden Softwaremodule lesen die Variablenwerte aus und verarbeiten diese in ihrem Algorithmus. Eine derartige Variable kann beispielsweise die maximale Fügekraft bei kraftgeregelten Fügeprozessen in der Montage sein. In Abhängigkeit von dem zu montierenden Teil kann diese Variable unterschiedliche Werte annehmen. Dieses Vorgehen ist wesentlich flexibler, als die zur Laufzeit nicht mehr änderbare Parametrierung des entsprechenden Softwaremoduls.

Softwaremodul zum Schreiben von Bahnkorrekturdaten

Das dritte Softwaremodul, mit der Bezeichnung `schreibeKorrdat` (schreibe Korrekturdaten), dient der Datenübertragung vom Sensorrechner zur Robotersteuerung. Diese Daten stellen Korrekturwerte dar, die die Bahn des Roboters in gewünschter Weise beeinflussen. Neben den Korrekturwerten für Position und Orientierung, wird auch ein sog. Overridefaktor übertragen. Dieser nimmt Werte zwischen null und einhundert Prozent an und verändert damit die programmierte Bahngeschwindigkeit. Der Overridefaktor wird in Abhängigkeit der Sensorwerte bestimmt. Ein typischer Anwendungsfall ist z.B. das Entgraten von Gußteilen. Meldet der angeschlossene Kraftsensor bei massiven Gratausprägungen zu hohe Bearbeitungskräfte, so wird über den Overridefaktor die Bahngeschwindigkeit reduziert, um das Werkzeug und den Roboter nicht zu beschädigen.

6.2 Softwaremodule zur Vorverarbeitung von Sensordaten

Nach dem Einlesen der Sensordaten durch die hardwareabhängigen Softwaremodule, stehen die Sensordaten dem Softwaresystem zur weiteren Verarbeitung zur Verfügung. Diese liegen jedoch noch als Rohdaten vor, die die Meßwerte und Zustandsdaten in hardwarenaher Form enthalten. Um daraus die gewünschten

Meßwerte in einer SI-Einheit zu gewinnen, sind zwei Vorverarbeitungsschritte notwendig. Im ersten Verarbeitungsschritt werden die Zustandsdaten der einzelnen Sensoren ausgewertet. Im zweiten Verarbeitungsschritt erfolgt die Umwandlung der Sensordaten in Gleitkommazahlen, die die Meßwerte in ihren SI-Einheiten repräsentieren. Da es sich bei den Meßwerten eines Sensors meist um analoge Größen handelt, ist die einheitliche Darstellung als Gleitkommazahl naheliegend.

Aus Effektivitätsgründen, erfolgt die Verarbeitung der Sensordaten, im gesamten Softwaresystem, gruppenweise. Die Gruppenzuteilung der Sensordaten erfolgt im Hinblick auf deren weitere gemeinsame Verarbeitung durch das Softwaresystem. So bildet beispielsweise ein kartesisches Frame eine Gruppe von sechs Positions- bzw. Orientierungswerten. Diese können von einem mehrdimensional arbeitenden Sensorsystem oder von mehreren Einzelsensoren stammen. Für die weitere Verarbeitung werden sie zu einer Sensordatengruppe zusammengefaßt. Dies führt zu einer Effizienzsteigerung bei der Verarbeitung und bei der Übertragung über das Kommunikationssystem. Für jede Sensordatengruppe sind je zwei Softwaremodule notwendig, die die beiden Vorverarbeitungsschritte durchführen.

6.2.1 Auswertung der Sensor-Zustandsdaten

Das Softwaremodul zum Einlesen der Sensorrohdaten (`leseEA`) liefert das gesamte Eingangsabbild des Feldbussystems als ein zusammenhängender Datenblock. Dieser Datenblock enthält die Meßwerte und Zustandsdaten aller Sensoren. Das Softwaremodul `sensorstatus` liest die Zustandsdaten der Sensoren einer Sensordatengruppe aus diesem Datenblock, verarbeitet diese und generiert einen neuen Datenblock mit einer systemweit einheitlichen Zustandsinformation und den Meßwerten der Sensoren (Bild 40). Dabei liegen die Meßwerte jedoch noch in einem hardwarenahen Format vor.

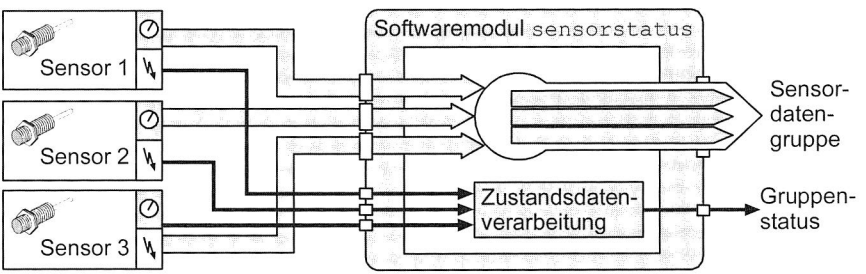


Bild 40: Struktur des Softwaremoduls zur Auswertung der Sensor-Zustandsdaten

Die Zustandsdaten können von einem Sensor als Zahl geliefert werden, die eine Fehlernummer repräsentiert. Eine Alternative besteht darin, die Zustandsdaten als

Bitfeld zu übergeben. Das Setzen oder auch Löschen eines Bits signalisiert das Auftreten eines bestimmten Fehlertyps.

Je nach verwendetem Datentyp belegen die Zustandsdaten eine unterschiedliche Anzahl von Bytes in dem vom Modul `leseEA` bereitgestellten Datenblock. Außerdem ist die Anzahl und Position der Zustandsdaten im Datenblock vom verwendeten Sensor und der Konfiguration des Feldbussystems abhängig. Über die Parametrierung dieses Softwaremoduls gibt der Anwender die benötigten Informationen vor, die die Extraktion und Auswertung der Zustandsdaten ermöglichen. Um eine maximale Flexibilität zu erreichen, geschieht die Auswertung bitorientiert. Das bedeutet, daß bei der Parametrierung angegeben werden muß, welche Bitpositionen im Datenblock zu einem Zustandsdatum gehören und welchen Wert diese Bits bei Fehlerfreiheit annehmen müssen. Der Auswertealgorithmus muß dann nur noch den Sollwert mit dem Istwert an den Bitpositionen vergleichen. Mit diesem Vorgehen können sowohl zahlenkodierte Zustandsdaten als auch bitorientierte Zustandsdaten ausgewertet werden.

Für die Konfiguration des Softwaresystems ist von Bedeutung, daß für jede Sensordatengruppe ein Softwaremodul vom Typ `sensorstatus` in die Konfiguration aufgenommen wird.

6.2.2 Verarbeitung der Sensorrohdaten

Das Softwaremodul `sensorkonv` (Sensordaten konvertieren) liest den Fehlerstatus der zuletzt durchgeführten Messung und die dabei ermittelten aktuellen Meßwerte eines Sensors von dem Softwaremodul `sensorstatus` als Datenblock. Die Meßwerte liegen noch als Rohdaten in dem gelesenen Datenblock vor. Die Aufgabe dieses Softwaremoduls besteht darin, die Sensorrohdaten einzeln aus dem Datenblock zu extrahieren und in Gleitkommazahlen, des C-Datentyps "float" nach IEEE-Standard 754-1985 [62], umzuwandeln. Diese Gleitkommazahl entspricht dem Meßwert in seiner SI-Einheit.

Bei Sensoren, die ihr Meßsignal als analoge Spannung ausgeben, erfolgt im Eingabemodul des Feldbusses eine Analog/Digital-Umsetzung. Das Ergebnis ist eine Ganzzahl mit einer Länge von 16 Bit. Da die Ausgangsspannung des Sensors auch negative Werte annehmen kann, wird die eingelesene Ganzzahl in der sog. Zweierkomplementdarstellung übertragen [106]. Mit der Zweierkomplementdarstellung ist es möglich negative Ganzzahlen darzustellen. Eine negative Zahl wird dabei durch Komplementierung aller Bits der betragsmäßig gleichen positiven Zahl und anschließender Addition von eins erzeugt. Der Wertebereich einer 16Bit-Ganzzahl liegt damit zwischen -32768 und 32767.

Sensoren, die über einen eigenen Feldbusanschluß verfügen, übertragen ihre Sensordaten ebenfalls als Ganzzahlen der Länge 16 Bit. Nehmen diese Ganzzahlen keine negativen Werte an, so ist die Darstellung als Zweierkomplement nicht notwendig.

Außerdem halbiert die Zweierkomplementdarstellung den Wertebereich positiver Zahlen. Die Darstellung positiver Ganzzahlen kann auch in der Betragsdarstellung erfolgen. Dabei werden sämtliche Bits für die Zahlendarstellung verwendet, da auf ein Vorzeichenbit verzichtet wird. Der Wertebereich einer immer positiven 16Bit-Ganzzahl liegt damit zwischen 0 und 65535. Bei der Parametrierung dieses Softwaremoduls muß der Anwender angeben, ob die Ganzzahl in der Zweierkomplementdarstellung oder in der Betragsdarstellung vorliegt.

Eine weitere wichtige Angabe, die der Anwender bei der Parametrierung angeben muß, stellt die verwendete Byteorder dar [106]. Die Repräsentation von Ganzzahlen im Speicher eines Rechnersystems erfordert eine unterschiedliche Anzahl von Bytes, die über verschiedene Wertigkeiten verfügen. Die Byteorder legt die Reihenfolge dieser Bytes fest. Werden die Bytes nach der Byteorder *Little Endian* im Speicher abgelegt, so werden die niederwertigen Bytes an niedrigeren Adressen und die höherwertigen Bytes an höheren Adressen abgelegt. Bei der Byteorder *Big Endian* verhält es sich genau umgekehrt. Werden analoge Sensoren eingesetzt, so wird die Byteorder von dem Feldbussystem festgelegt. Bei Konfiguration des Feldbussystems ist darauf zu achten, das dessen Byteorder mit dem des Softwaresystems übereinstimmt. Hardwareplattformen mit x86-Architektur (PCs) verwenden das Little Endian Format. Werden die Sensordaten von einem Sensor geliefert, der über eine eigene Feldbusschnittstelle verfügt, so kann die Byteorder der Sensordaten von der des Feldbussystems bzw. Softwaresystems abweichen. Für diesen Fall wurde der Parameter Byteorder vorgesehen, der es erlaubt die Byteorder der Sensordaten zu berücksichtigen und die Sensordaten bei Bedarf umzuwandeln.

Für die Umwandlung einer Ganzzahl in eine Gleitkommazahl, die dem Meßwert in seiner SI-Einheit entspricht, sind zwei weitere Angaben notwendig. Es handelt sich um die untere Meßbereichsgrenze und die obere Meßbereichsgrenze des Sensors, die als Parameter angegeben werden müssen. Die dem Meßwert entsprechende Gleitkommazahl kann dann nach Gl. 1 bis Gl. 3 berechnet werden.

Normierung der Ganzzahl bei Betragsdarstellung auf Intervall [0..1]:

$$\text{Ganzzahl\_norm} = \frac{\text{Ganzzahl}}{65535} \qquad \text{Gl. 1}$$

Normierung der Ganzzahl bei Zweierkomplementdarstellung auf Intervall [0..1]:

$$\text{Ganzzahl\_norm} = \left(\frac{\text{Ganzzahl}}{32768} + 1\right) \div 2 \qquad \text{Gl. 2}$$

Bestimmung des Meßwertes:

$$\text{Meßwert} = \text{Offset} + \text{MBU} + (\text{MBO} - \text{MBU}) \cdot \text{Ganzzahl\_norm} \qquad \text{Gl. 3}$$

mit MBU = untere Meßbereichsgrenze und MBO = obere Meßbereichsgrenze

Häufig tritt bei der Analog/Digital-Umsetzung des Feldbuseingabemoduls ein Offsetfehler des Umsetzers auf. Dies äußert sich darin, daß einer Eingangsspannung von Null Volt nicht die Zahl Null entspricht. Dies kann durch die Software kompensiert werden, wenn der Anwender für den Parameter *Offset* einen Wert, verschieden von Null, vorgibt. Das Modul `sensorkonv` stellt seine Verarbeitungsergebnisse anderen Softwaremodulen als Datenblock zur Verfügung.

6.3 Verarbeitung von Sensordaten zur Bahnkorrektur des Roboters

Äquivalent zur Vorverarbeitung der Sensordaten erfolgt die Weiterverarbeitung ebenfalls gruppenweise. In diesem Kapitel sollen die Softwaremodule vorgestellt werden, die die wichtigste Gruppe von Sensordaten weiterverarbeiten. Es handelt sich um die Gruppe von Sensordaten, die für die Bahnkorrektur des Roboters benötigt werden.

6.3.1 Transformation der Sensormeßwerte

Die Aufgabe einer sensorgestützten Bahnkorrektur für Roboter besteht darin, unter Nutzung von Sensormeßwerten eine Stellbewegung für die Robotermechanik zu bestimmen. Die Stellbewegung bezieht sich dabei auf das Referenzkoordinatensystem des Roboters. Die Sensormeßwerte hingegen beziehen sich auf das Koordinatensystem des Sensors. Sind beide Koordinatensysteme orthogonal zueinander ausgerichtet, so können die Meßwerte direkt in Stellbewegungen umgesetzt werden. Sind sie hingegen zueinander verdreht, so müssen die Meßwerte zuerst durch Rotationstransformationen in das Koordinatensystem der Stellbewegung überführt werden. Die Aufgabe des Softwaremoduls `transformation` besteht darin, die Sensormeßwerte in das Roboterkoordinatensystem zu transformieren (Bild 41).

Prinzipiell unterscheidet man in diesem Zusammenhang drei Koordinatensysteme. Der Roboter führt die Stellbewegungen im Roboterkoordinatensystem $\{R\}$ aus. Dieses Koordinatensystem ist orthogonal und ortsfest mit dem Sockel des Roboters verbunden. Es dient als Referenzkoordinatensystem für daraus abgeleitete Koordinatensysteme.

Das zweite Koordinatensystem bezeichnet man als Werkzeugkoordinatensystem $\{W\}$. Der Ursprung dieses ebenfalls orthogonalen Koordinatensystems befindet sich an dem Eingriffspunkt des Werkzeugs. Allgemein wird dieser Punkt als Tool Center Point (TCP) bezeichnet. Die Orientierung des Werkzeugkoordinatensystems zum Werkzeug kann durch den Anwender frei festgelegt werden. Es hat sich jedoch als sinnvoll erwiesen, eine Koordinatenachse in die vorgesehene Verfahrrichtung des

Werkzeugs zu orientieren. Eine zweite Koordinatenachse sollte so orientiert werden, das sie in Richtung des Normalenvektors aus Verfahrrichtung und Wirkrichtung des Werkzeugs orientiert ist. Die Orientierung der dritten Koordinatenachse entspricht dann dem Vektor des Vektorproduktes aus den beiden anderen Achsen.

Wurde das Werkzeugkoordinatensystem durch den Anwender noch nicht festgelegt, so verwendet die Robotersteuerung ein Standardkoordinatensystem, dessen Ursprung sich im Roboterflansch befindet. Eine Achse dieses Koordinatensystems verläuft in Richtung des Normalenvektors auf den Roboterflansch. Sieht die Stellbewegung der Robotersteuerung eine Orientierungsänderung vor, so bezieht sich diese auf das eingestellte Werkzeugkoordinatensystem. Eine Orientierungsänderung darf jedoch keinen Einfluß auf die Positionsänderung des Eingriffspunktes des Werkzeugs haben. Deshalb muß der Ursprung des Standardkoordinatensystems in den Eingriffspunkt des Werkzeugs verschoben werden. Außerdem muß die Verdrehung des realen Werkzeugkoordinatensystems zum Standardkoordinatensystem festgelegt werden.

Die Bestimmung der translatorischen und rotatorischen Beziehungen zwischen dem Standardkoordinatensystem und dem realen Werkzeugkoordinatensystem bezeichnet man als Vermessung des Werkzeugs. Die Werkzeugvermessung muß äußerst genau erfolgen, da sie großen Einfluß auf die Positioniergenauigkeit des Werkzeugeingriffpunktes hat. Eine moderne Robotersteuerung unterstützt den Anwender deshalb durch entsprechende Routinen bei der Werkzeugvermessung.

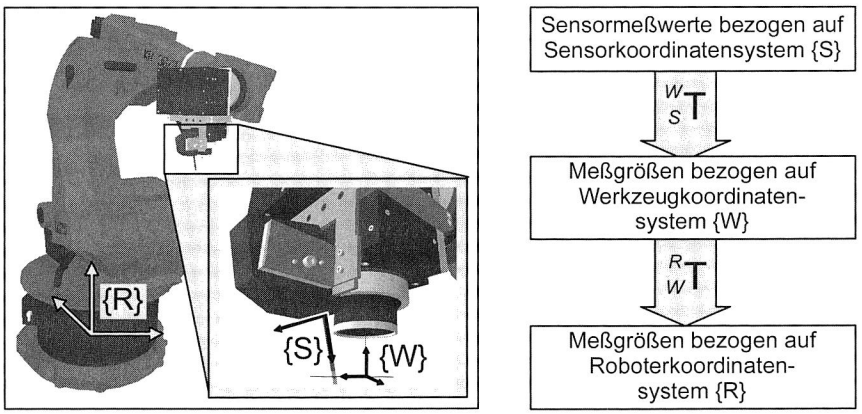


Bild 41: Darstellung der Koordinatensysteme von Roboter, Werkzeug und Sensor

Das dritte Koordinatensystem, das in diesem Zusammenhang von Bedeutung ist, ist das Sensorkoordinatensystem $\{S\}$. Es handelt sich hierbei um ein ein- bis dreidimen-

sionales Koordinatensystem. Die Koordinatenachsen sind durch die Meßrichtungen der verwendeten Sensoren festgelegt.

Es ist theoretisch möglich die Meßrichtungen beliebig, und damit auch nichtorthogonal, zu wählen. Aufgrund der einfacheren mathematischen Handhabung werden in der Praxis Sensoren bzw. Sensorsysteme aus mehreren Sensoren meist so aufgebaut, daß ihre Meßrichtungen orthogonal zueinander sind. Jeder Einzelmeßwert kann damit als Vielfaches des Einheitsvektors einer Koordinatenachse, im Sensorkoordinatensystem, dargestellt werden. Der Betrag dieses Vektors wird durch den Betrag des Meßwertes festgelegt. Sollte der Ausnahmefall auftreten, daß die Meßrichtungen der Sensoren nicht orthogonal zueinander angeordnet sind, so kann ein spezielles Softwaremodul entwickelt und in das Softwaresystem integriert werden, daß eine Transformation der Meßwerte in das orthogonale Sensorkoordinatensystem $\{S\}$ durchführt.

Die wesentliche Aufgabe des Softwaremoduls `transformation` besteht darin, die Vektoren der Meßwerte aus dem Sensorkoordinatensystem $\{S\}$ in das Roboterkoordinatensystem $\{R\}$ zu transformieren, da das Bahnkorrekturmodul im Roboterkoordinatensystem arbeitet. Diese Transformation kann durch eine einzige Transformationsmatrix beschrieben werden, wenn das Sensorkoordinatensystem mit dem Werkzeugkoordinatensystem $\{W\}$ übereinstimmt. Dies würde jedoch bedeuten, daß der Ursprung des Sensorkoordinatensystems im Eingriffspunkt des Werkzeugs liegt. Da dies bei den meisten Sensoren nicht der Fall ist, müssen die Sensormeßwerte zuerst in das Werkzeugkoordinatensystem transformiert werden.

Ein Sensor kann, unter Berücksichtigung technologischer Randbedingungen, in vielfältiger Weise an dem Werkzeug mechanisch befestigt werden. Durch die mechanische Befestigung bestehen zwischen dem Werkzeugkoordinatensystem und dem Sensorkoordinatensystem definierte mathematische Beziehungen. Die Bestimmung der translatorischen und rotatorischen Beziehungen zwischen dem Sensorkoordinatensystem und dem Werkzeugkoordinatensystem bezeichnet man als Kalibrierung des Sensors. Ebenso wie die Werkzeugvermessung muß auch die Kalibrierung des Sensors mit äußerster Sorgfalt durchgeführt werden, da sonst die Meßwerte fehlerhaft transformiert werden. Häufig werden zur Unterstützung der Kalibrierung mechanische Hilfsmittel eingesetzt. Diese sind an den spezifischen Sensor und an die mechanischen Gegebenheiten des Systemaufbaus anzupassen.

Transformation eines Positionsvektors von $\{S\}$ nach $\{W\}$

Das Sensorkoordinatensystem geht aus dem Werkzeugkoordinatensystem durch die Anwendung translatorischer und rotatorischer Operationen hervor. Dies wird mathematisch durch die (4,4)-Transformationsmatrix ${}^{W}_{S}\mathbf{T}$ beschrieben.

Diese hat den folgenden Aufbau:

$$ {}^{W}_{S}\mathbf{T} = \begin{bmatrix} {}^{W}_{S}\mathbf{R} & {}^{W}\mathbf{o}_{S} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \qquad \text{Gl. 4} $$

Diese Darstellung wird auch als Frame bezeichnet. Der Vektor ${}^{W}\mathbf{o}_{S} = (sx, sy, sz)^{\mathrm{T}}$ beschreibt die Translation des Ursprunges von $\{\mathrm{S}\}$ aus dem Ursprung von $\{\mathrm{W}\}$. Die Komponenten sx, sy, sz beziehen sich auf die Einheitsvektoren $\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W$ von $\{\mathrm{W}\}$.

Die (3,3)-Rotationsmatrix ${}^{W}_{S}\mathbf{R}$ kann zweifach interpretiert werden. Sie beschreibt die Orientierung von $\{\mathrm{S}\}$ bezüglich $\{\mathrm{W}\}$. Sie beschreibt aber auch die Rotationstransformation mit der $\{\mathrm{W}\}$ in $\{\mathrm{S}\}$ übergeht. Die Rotationstransformation erfolgt in drei Schritten. Dabei ist auf die Reihenfolge zu achten.

Die Reihenfolge wurde folgendermaßen festgelegt:

1. Drehung um die $\mathbf{x}_W$-Achse mit dem Winkel α
2. Drehung um die $\mathbf{y}_W$-Achse mit dem Winkel β
3. Drehung um die $\mathbf{z}_W$-Achse mit dem Winkel γ.

Die Drehung von $\{\mathrm{S}\}$ um die $\mathbf{x}_W$-Achse mit dem Winkel α beschreibt die Rotationsmatrix $Rot(\mathbf{x}_W, \alpha)$:

$$ Rot(\mathbf{x}_W, \alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \qquad \text{Gl. 5} $$

Die Drehung von $\{\mathrm{S}\}$ um die $\mathbf{y}_W$-Achse mit dem Winkel β beschreibt die Rotationsmatrix $Rot(\mathbf{y}_W, \beta)$:

$$ Rot(\mathbf{y}_W, \beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \qquad \text{Gl. 6} $$

Die Drehung von $\{\mathrm{S}\}$ um die $\mathbf{z}_W$-Achse mit dem Winkel γ beschreibt die Rotationsmatrix $Rot(\mathbf{z}_W, \gamma)$:

$$Rot(\mathbf{z}_W, \gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Gl. 7}$$

Die Gesamtrotationsmatrix ${}^{W}_{S}\mathbf{R}$ ergibt sich dann zu:

$${}^{W}_{S}\mathbf{R} = Rot(\mathbf{z}_W, \gamma) \cdot Rot(\mathbf{y}_W, \beta) \cdot Rot(\mathbf{x}_W, \alpha) \quad \text{Gl. 8}$$

Dabei ist zu beachten, daß die Multiplikationsreihenfolge umgekehrt zu der Durchführung der Drehungen sein muß.

Mit Einsetzen von Gl. 5, Gl. 6 und Gl. 7 in Gl. 8 erhält man für ${}^{W}_{S}\mathbf{R}$:

$${}^{W}_{S}\mathbf{R} = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\alpha\cos\gamma & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \quad \text{Gl. 9}$$

Die Transformationsmatrix ${}^{W}_{S}\mathbf{T}$ zur Beschreibung von Transformationen aus dem Sensorkoordinatensystem in das Werkzeugkoordinatensystem ergibt sich nach Gl. 4 zu:

$${}^{W}_{S}\mathbf{T} = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & sx \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\alpha\cos\gamma & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha & sy \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha & sz \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Gl. 10}$$

Die Sensormeßwerte m_x, m_y, m_z bilden durch Multiplikation mit den Einheitsvektoren $\mathbf{e}_{x,S}$, $\mathbf{e}_{y,S}$, $\mathbf{e}_{z,S}$ in $\{S\}$ den Vektor $\mathbf{p}_S = \mathbf{e}_{x,S}m_x + \mathbf{e}_{y,S}m_y + \mathbf{e}_{z,S}m_z$. Dieser beschreibt einen Punkt im Sensorkoordinatensystem.

Die Transformation ${}^{W}_{S}\mathbf{T}$ transformiert diesen Punkt in das Werkzeugkoordinatensystem. Er trägt dort die Bezeichnung $\mathbf{p}_W$. Da es sich bei der Transformationsmatrix um eine (4,4)-Matrix handelt, müssen die Vektoren $\mathbf{p}_S$ und $\mathbf{p}_W$ um eine Komponente erweitert werden, die jedoch immer den Wert eins annimmt. Gl. 11 verdeutlicht die Zusammenhänge:

$$\begin{bmatrix} \mathbf{p}_W \\ 1 \end{bmatrix} = {}^{W}_{S}\mathbf{T} \cdot \begin{bmatrix} \mathbf{p}_S \\ 1 \end{bmatrix} \quad \text{Gl. 11}$$

Der Vektor $\mathbf{p}_W$ beschreibt die Position des Punktes $\mathbf{p}_S$ im Werkzeugkoordinatensystem.

Transformation eines Positionsvektors von $\{W\}$ nach $\{R\}$

Aus der Stellung des Roboters und den Daten aus der Werkzeugvermessung sind die translatorischen und rotatorischen Beziehungen zwischen dem Werkzeugkoordinatensystem und dem Roboterkoordinatensystem bekannt. Die in das Werkzeugkoordinatensystem transformierten Sensormeßwerte können damit in einem zweiten Schritt in das Roboterkoordinatensystem transformiert werden.

Hierbei ist zu beachten, daß eine Robotersteuerung die Orientierungen des Werkzeugkoordinatensystems $\{W\}$ im Roboterkoordinatensystem $\{R\}$ des Roboters mit Z-Y-X-Euler-Winkeln beschreibt.

Dabei wird $\{W\}$ aus $\{R\}$ nach folgender Methode gewonnen:

1. Drehung um die $\mathbf{z}_W$-Achse (entspricht $\mathbf{z}_R$-Achse) mit dem Winkel γ
2. Drehung um die *neue* $\mathbf{y}_W$-Achse mit dem Winkel β
3. Drehung um die *neue* $\mathbf{z}_W$-Achse mit dem Winkel α.

Es ist zu beachten, daß alle Rotationen um die Achsen von $\{W\}$ ausgeführt werden.

Damit ergibt sich für die Rotationsmatrix ${}^{R}_{W}\mathbf{R}$:

$$ {}^{R}_{W}\mathbf{R} = Rot(\mathbf{z}_W, \gamma) \cdot Rot(\mathbf{y}_W, \beta) \cdot Rot(\mathbf{x}_W, \alpha) \qquad \text{Gl. 12} $$

Die Translation des Ursprungs des Werkzeugkoordinatensystems $\{W\}$ aus dem Roboterkoordinatensystem $\{R\}$ wird durch den Translationsvektor ${}^{R}\mathbf{o}_W = (wx, wy, wz)^T$ beschrieben.

Analog zu Gl. 10, ergibt sich damit für die Transformationsmatrix von $\{R\}$ nach $\{W\}$:

$$ {}^{R}_{W}\mathbf{T} = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & wx \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\alpha\cos\gamma & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha & wy \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha & wz \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Gl. 13

Die Transformation ${}^{R}_{S}\mathbf{T}$ von $\{R\}$ nach $\{S\}$ wird zusammenfassend beschrieben durch:

$$ {}^{R}_{S}\mathbf{T} = {}^{R}_{W}\mathbf{T} \cdot {}^{W}_{S}\mathbf{T} \qquad \text{Gl. 14} $$

Transformation eines Kraftvektors von $\{S\}$ nach $\{R\}$

Nicht alle Sensoren liefern Abstands bzw. Positionsangaben. Eine wichtige Klasse von Sensoren, die Kraft/Momentensensoren, liefern richtungsabhängige Kraft- bzw. Momentenmeßwerte. Ihre Transformation erfordert ein geringfügung modifiziertes Vorgehen. Die Kraft- bzw. Momentenkomponenten bilden im Sensorkoordinatensystem den Vektor $\mathbf{r}_S$. Um die Kraft- bzw. Momentenkomponenten im Werkzeugkoordinatensystem beschreiben zu können, kann ebenfalls die Transformationsmatrix ${}^{W}_{S}\mathbf{T}$, nach Gl. 10, angewendet werden. Es ist allerdings zu beachten, daß dabei die Komponenten sx, sy, sz des Translationsvektors ${}^{W}\mathbf{o}_S$ zu Null gesetzt werden müssen. Diese beschreiben nämlich eine räumliche Translation, die bei der Transformation von Kraft- bzw. Momentenkomponenten nicht benötigt wird.

Mit Hilfe der Gl. 15 werden die Kraftkomponenten auf das Werkzeugkoordinatensystem bezogen:

$$\begin{bmatrix} \mathbf{r}_W \\ 1 \end{bmatrix} = {}^{W}_{S}\mathbf{T} \cdot \left. \begin{bmatrix} \mathbf{r}_S \\ 1 \end{bmatrix} \right|_{sx = sy = sz = 0} \qquad \text{Gl. 15}$$

Der Vektor $\mathbf{r}_W$ gibt die Kraftkomponenten in Richtung der Koordinatenachsen des Werkzeugkoordinatensystems an. Bei der Weiterverarbeitung des Vektors $\mathbf{r}_W$ muß jedoch berücksichtigt werden, daß dieser sich nicht auf den Ursprung des Werkzeugkoordinatensystems bezieht, sondern auf den des Sensorkoordinatensystems. Der Vektor des Bezugspunktes für $\mathbf{r}_W$ wird damit durch den Vektor ${}^{W}\mathbf{o}_S$ beschrieben. Die Transformation in das Roboterkoordinatensystem muß für die Vektoren $\mathbf{r}_W$ und ${}^{W}\mathbf{o}_S$ getrennt erfolgen, da es sich bei $\mathbf{r}_W$ um die vektorielle Darstellung von Kraft- bzw. Momentenkomponenten und bei ${}^{W}\mathbf{o}_S$ um eine Positionsangabe handelt. Weiterhin ist zu beachten, daß bei der Transformation von $\mathbf{r}_W$ die Komponenten wx, wy, wz des Translationsvektors von $\{R\}$ nach $\{W\}$ wieder zu Null gesetzt werden müssen.

Die Kraft- bzw. Momentenkomponenten werden im Roboterkoordinatensystem $\{R\}$ beschrieben durch den Vektor $\mathbf{r}_R$:

$$\begin{bmatrix} \mathbf{r}_R \\ 1 \end{bmatrix} = {}^{R}_{W}\mathbf{T} \cdot \left. \begin{bmatrix} \mathbf{r}_W \\ 1 \end{bmatrix} \right|_{wx = wy = wz = 0} \qquad \text{Gl. 16}$$

Den Bezugspunkt für die Kräfte bzw. Momente stellt der Ursprung des Sensorkoordinatensystems $\{S\}$ dar. Die Beschreibung dieses Punktes erfolgt durch den Vektor ${}^{R}\mathbf{o}_S$:

$$\begin{bmatrix} {}^{R}\mathbf{o}_{S} \\ 1 \end{bmatrix} = {}^{R}_{W}\mathbf{T} \cdot \begin{bmatrix} {}^{W}\mathbf{o}_{S} \\ 1 \end{bmatrix} = {}^{R}_{W}\mathbf{T} \cdot {}^{W}_{S}\mathbf{T} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}} \qquad \text{Gl. 17}$$

Die Weiterverarbeitung von Kraft- bzw. Momentenmeßwerten erfolgt unterschiedlich. Kraftmeßwerte resultieren in Positionskorrekturen des TCP, während Momentenmeßwerte zu Korrekturen in der Orientierung des TCP führen. Beide Korrekturmechanismen dienen dazu die auftretenden Kräfte bzw. Momente die vorgegebenen Sollwerte anzugleichen.

Transformation von Orientierungen nach $\{R\}$

Einen häufigen Anwendungsfall stellt die Bestimmung der Orientierung eines sensorisch erfaßten Merkmals des Meßobjekts zum Werkzeugkoordinatensystem dar. Da die Orientierung keine physikalische Größe ist, kann sie nicht direkt gemessen werden. Vielmehr wird die Orientierung aus den Meßwerten abstandsmessender Sensoren mathematisch bestimmt. Die Verarbeitungsvorschrift zur Bestimmung der Orientierung aus den Meßwerten ist sensorabhängig. Dafür ist ein sensorspezifischer Algorithmus vorzusehen, der bei Bedarf in ein eigenes Softwaremodul integriert wird. In [122] wird beispielhaft ein Verfahren vorgestellt, mit dem aus drei Abstands-Meßwerten die Orientierung des Meßobjekts, im Werkzeugkoordinatensystem, bestimmt werden kann. Es handelt sich hierbei wieder um eine Transformationsvorschrift.

Die Orientierung gibt an, um welche Winkel das Werkzeugkoordinatensystem gedreht werden muß, damit die Orientierung des Werkzeugkoordinatensystems mit der sensorisch bestimmten Orientierung des Meßobjekts übereinstimmt. Die Drehung des Werkzeugkoordinatensystems geschieht dabei um dessen Koordinatenachsen $\mathbf{x}_W$, $\mathbf{y}_W$, $\mathbf{z}_W$. Die entsprechenden Drehwinkel sollen mit δ, λ, μ bezeichnet werden. Die Drehwinkel entsprechen den in der Schiffahrt verwendeten Faktoren für das Rollen, Gieren und Nicken (engl.: roll, pitch, yaw) [118].

Für die Reihenfolge der Drehungen wurde festgelegt:

1. Drehung um die $\mathbf{x}_W$-Achse mit dem Winkel δ
2. Drehung um die $\mathbf{y}_W$-Achse mit dem Winkel λ
3. Drehung um die $\mathbf{z}_W$-Achse mit dem Winkel μ.

Damit ergibt sich die Rotationsmatrix:

$${}^{W}_{S}\mathbf{R} = \begin{bmatrix} \cos\mu\cos\lambda & \cos\mu\sin\lambda\sin\delta - \sin\mu\cos\delta & \cos\delta\sin\lambda\cos\mu + \sin\delta\sin\mu \\ \sin\mu\cos\lambda & \sin\mu\sin\lambda\sin\delta + \cos\delta\cos\mu & \sin\mu\sin\lambda\cos\delta - \cos\mu\sin\delta \\ -\sin\lambda & \cos\lambda\sin\delta & \cos\lambda\cos\delta \end{bmatrix} \qquad \text{Gl. 18}$$

Die Beschreibung der Orientierung des Werkzeugkoordinatensystems $\{W\}$ zum Roboterkoordinatensystem $\{R\}$ kann durch die bereits hergeleitete Rotationsmatrix ${}^{R}_{W}\mathbf{R}$ erfolgen.

Die Transformation der Orientierung des Meßobjekts vom Sensorkoordinatensystem $\{S\}$ in das Roboterkoordinatensystem $\{R\}$ wird durch die Rotationsmatrix ${}^{R}_{S}\mathbf{R}$ beschrieben:

$${}^{R}_{S}\mathbf{R} = {}^{R}_{W}\mathbf{R} \cdot {}^{W}_{S}\mathbf{R} \qquad \text{Gl. 19}$$

Da es sich bei der Matrix ${}^{R}_{S}\mathbf{R}$ um eine Rotationsmatrix handelt, verfügt sie über die gleiche Struktur, wie die Rotationsmatrizen ${}^{R}_{W}\mathbf{R}$ und ${}^{W}_{S}\mathbf{R}$. Sie läßt sich damit in einer verallgemeinerten Form darstellen:

$${}^{R}_{S}\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad \text{Gl. 20}$$

$${}^{R}_{S}\mathbf{R} = \begin{bmatrix} \cos(c)\cos(b) & \cos(c)\sin(b)\sin(a) - \sin(c)\cos(a) & \cos(a)\sin(b)\cos(c) + \sin(a)\sin(c) \\ \sin(c)\cos(b) & \sin(c)\sin(b)\sin(a) + \cos(a)\cos(c) & \sin(c)\sin(b)\cos(a) - \cos(c)\sin(a) \\ -\sin(b) & \cos(b)\sin(a) & \cos(b)\cos(a) \end{bmatrix}$$

Gl. 21

Aus dieser Matrix müssen im folgenden Schritt die Orientierungswinkel a, b, c in Bezug auf das Roboterkoordinatensystem $\{R\}$ bestimmt werden. Dazu wählt man Elemente, die möglichst einfache Gleichungen ergeben und stellt diese in Form einer Arcustangensfunktion auf:

$$a = \operatorname{atan}\left(\frac{r_{32}}{r_{33}}\right) \qquad \text{Gl. 22}$$

$$b = \operatorname{atan}\left(\frac{-r_{31}}{\pm\sqrt{r_{11}^{\,2} + r_{21}^{\,2}}}\right) \qquad \text{Gl. 23}$$

$$c = \operatorname{atan}\left(\frac{r_{21}}{r_{11}}\right) \qquad \text{Gl. 24}$$

Die Quotienten entsprechen der Tangensfunktion des entsprechenden Orientierungswinkels. Bei der Anwendung dieser Gleichungen ist deshalb die Mehrdeutigkeit der

Tangensfunktion zu beachten! Die Gleichungen gelten nur, falls $\cos(b) \neq 0$ ist. Ist $\cos(b) = 0$, so sind die obigen Gleichungen nicht mehr gültig. Ein Rechneralgorithmus muß diese Bedingung zuerst prüfen. Dies kann mit Hilfe des Elementes r_{31} erfolgen. Im Falle der Ungültigkeit der Gleichungen besteht die Möglichkeit den jeweils letzten gültigen Wert der Orientierungswinkel zu verwenden, bis der Nulldurchgang der Kosinusfunktion des Orientierungswinkels *b* abgeschlossen ist.

Bestimmung des Meßzeitpunktes

Zum Abschluß der Ausführungen zum Softwaremodul `transformation` soll noch ein systemtechnischer Aspekt betrachtet werden. Es handelt sich um die Bedeutung des Meßzeitpunktes.

Die Transformationsmatrix ${}^{R}_{W}\mathbf{T}$ und die Rotationsmatrix ${}^{R}_{W}\mathbf{R}$ werden gebildet aus dem aktuellen Translationsvektor ${}^{R}\mathbf{o}_{W}$ und/oder aus den aktuellen Orientierungswinkeln α, β, γ des Werkzeugkoordinatensystems zu dem Roboterkoordinatensystem. Diese Größen werden von der Robotersteuerung zu Beginn eines jeden Interpolationszykluses aus den aktuellen Gelenkwinkeln der Roboterachsen bestimmt.

Die Aktualisierung der Sensordaten durch den Sensor erfolgt hingegen in einem festen Zeitraster, das von der Dauer der Meßwertaufnahme und der Dauer der sensorinternen Vorverarbeitung der Meßwerte abhängt. Dieses Zeitraster ist zwar fest, stimmt aber in der Regel nicht mit der Zeitdauer eines Interpolationszykluses überein. Dies bedeutet, daß die Meßwerte, die aktuell im Eingangsabbild vorliegen, nicht zu dem Zeitpunkt vom Sensor aufgenommen wurden, zu dem die Größen, die von der Robotersteuerung berechnet werden, gültig waren. Die in den Matrizen verwendeten Größen müssen sich auf den gleichen Zeitpunkt beziehen, da sich die Roboterstellung über die Zeit verändert. Als zeitlicher Bezugspunkt wird der Beginn des aktuellen Interpolationszykluses festgelegt. Zu diesem Zeitpunkt werden von der Robotersteuerung die Positions- und Orientierungsgrößen des Werkzeugkoordinatensystems aus der Roboterstellung bestimmt. Benötigt wird die Zeitspanne, die zwischen der letzten Meßwertaufnahme durch den Sensor und dem Beginn des Interpolationstaktes verstrichen ist.

Diese Zeitspanne wird aus vier unterschiedlichen Zeitabschnitten gebildet (Bild 42). Der erste Zeitabschnitt umfaßt die Zeitspanne zwischen der physikalischen Meßwertaufnahme und dem Ende der sensorinternen Vorverarbeitungsschritte. Diese Zeitspanne ist sensorspezifisch, aber konstant. Sie kann dem Datenblatt des Sensors entnommen werden und soll mit T_m bezeichnet werden.

Der zweite Zeitabschnitt umfaßt die Zeitspanne vom Ende des ersten Zeitabschnittes bis zu dem Zeitpunkt, zu dem die Sensordaten über den Feldbus zum Softwaresystem übertragen werden. Diese Zeitspanne soll mit T_w bezeichnet werden. Da die Zeitspanne für einen Meßzyklus von der des Interpolationszykluses abweicht, ist T_w

nicht konstant. Diese Zeitspanne wird als Zeitstempel bezeichnet. Intelligente Sensoren, übertragen den Zeitstempel zusammen mit den restlichen Sensordaten an das Softwaresystem.

Das Softwaresystem selbst wird mit dem Interpolationstakt der Robotersteuerung synchronisiert. Nach der Transformation der Achsstellungen in kartesische Positions- und Orientierungswerte und der Ausführung einiger weiterer Roboterfunktionen wird das Softwaresystem durch das Auslösen eines Interrupts aktiviert. Die Zeitspanne zwischen dem Beginn des Interpolationstaktes und der Synchronisierung des Softwaresystems wird mit T_{sync} bezeichnet. Dieser Zeitabschnitt ist konstant und hängt von der verwendeten Robotersteuerung ab.

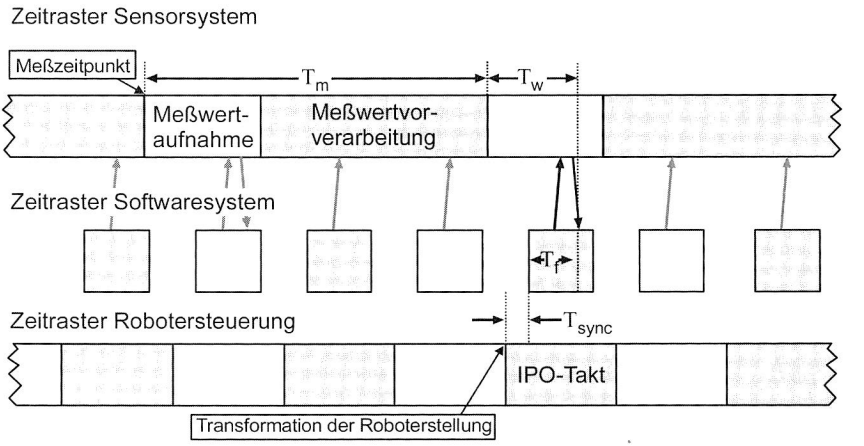


Bild 42: Darstellung der zeitlichen Abläufe zur Sensordatenerfassung

Nach dem Aktivieren des Softwaresystems und der Ausführung vorgelagerter Funktionen, werden die Sensordaten über das Feldbussystem eingelesen. Der Zeitabschnitt T_f repräsentiert die Zeitspanne bis zum Empfang der Sensordaten über den Feldbus.

Aus den eingeführten Zeitspannen kann nach Gl. 25 die Zeitspanne T_{verz} zwischen Meßzeitpunkt und Beginn des Interpolationstaktes bestimmt werden.

$$T_{verz} = T_m + T_w - (T_{sync} + T_f) \qquad \text{Gl. 25}$$

Mit Hilfe eines linearen Interpolationsverfahrens können damit der Translationsvektor ${}^{R}\mathbf{o}_{W}$ und die Orientierungswinkel α, β, γ zum Zeitpunkt der Meßwertaufnahme des Sensors bestimmt werden. Setzt man die interpolierten Werte in die Transformations-

matrix ${}^{R}_{W}\mathbf{T}$ ein, so werden Fehler bei der Transformation der Sensormeßwerte vermieden. Vor allem bei Anwendungen mit hohen Genauigkeitsanforderungen, wie sie z.B. die Lasermaterialbearbeitung erfordert, ist die Berücksichtigung des Zeitversatzes zwischen Meßwertaufnahme und Durchführung der Transformationen von qualitätsrelevanter Bedeutung.

6.3.2 Softwaremodul zur Berechnung der Bahnkorrekturwerte

Nach der Transformation der Sensormeßwerte in das Roboterkoordinatensystem werden diese durch das Bahnkorrekturmodul `bahnkorrektur` weiterverarbeitet. Dazu benötigt das Softwaremodul die aktuelle Istpose, die das Modul `leseIstpose` bereitstellt, und einige Statusinformationen der Robotersteuerung, die vom Modul `leseRobStatus` bereitgestellt werden. Die Aufgabe des Bahnkorrekturmoduls besteht darin, geeignete Korrekturwerte zur Anpassung der Bewegungsbahn des Roboters zu generieren. Die Korrekturwerte werden anschließend durch das Softwaremodul `schreibeKorrdat` an die Robotersteuerung übergeben.

Zur Berechnung der Bahnkorrekturwerte werden zwei unterschiedliche Verfahren eingesetzt (Bild 43). Die Entscheidung für eines der beiden Verfahren hängt stark von der Art der Meßwertaufnahme, durch den Sensor, ab. Erfolgt die Meßwertaufnahme am Werkzeugeingriffspunkt (TCP), so werden die Meßwerte durch einen Sensorregelkreis verarbeitet. Ist die Meßstelle zum TCP in Verfahrrichtung versetzt, so spricht man von einem vorlaufenden Sensor. Dessen Meßwerte werden durch einen Bahnplanungsalgorithmus verarbeitet.

Bahnkorrektur durch einen Sensorregelkreis

Die Meßwerterfassung in sensorbasierten Roboterzellen kann am Werkzeugeingriffspunkt stattfinden. Dabei muß die Messung nicht unbedingt durch einen Sensor erfolgen. Ein typischer Anwendungsfall ist beispielsweise das robotergestützte Entgraten von Gußwerkstücken [99]. Die Meßgröße stellt hier der Motorstrom des Entgratwerkzeugs dar. Es handelt sich also um einen Sensorwert, der von einem Aktor geliefert wird. Beim Auftreten großer Gratausprägungen ist eine größere Bearbeitungsleistung erforderlich. Eine Erhöhung des Motorstroms, durch den Antriebsregler, ist die Folge. Der Motorstrom kann bestimmt werden und dient als Eingangsgröße für einen Regelkreis zur Geschwindigkeitsregelung der programmierten Bewegungsbahn.

Einen anderen Anwendungsfall stellt die Messung des Fokusabstandes beim Laserstrahlschneiden dar [121]. Für die Schnittqualität ist es von entscheidender Bedeutung, daß der Laserstrahl optimal auf das Werkstück fokussiert wird. Dazu ist der Fokusabstand zu ermitteln und einem Sensorregelkreis zuzuführen. Die Messung des Fokus-

abstandes erfolgt durch einen zylinderförmigen, kapazitiven Sensor, der an der Austrittsöffnung der Strahldüse angebracht wird.

Eine weitere Gruppe von Sensoren, deren Meßwerte durch Sensorregelkreise verarbeitet werden, stellen die Kraft/Momentensensoren dar [79]. Ein derartiger Sensor wird zwischen Roboterflansch und Werkzeug angebracht. Die Messung erfolgt damit nicht direkt am Werkzeugeingriffspunkt. Durch die bekannte mechanische Anbringung des Werkzeugs am Sensor, können die gemessenen Kräfte und Momente jedoch auf den Werkzeugeingriffspunkt bezogen werden.

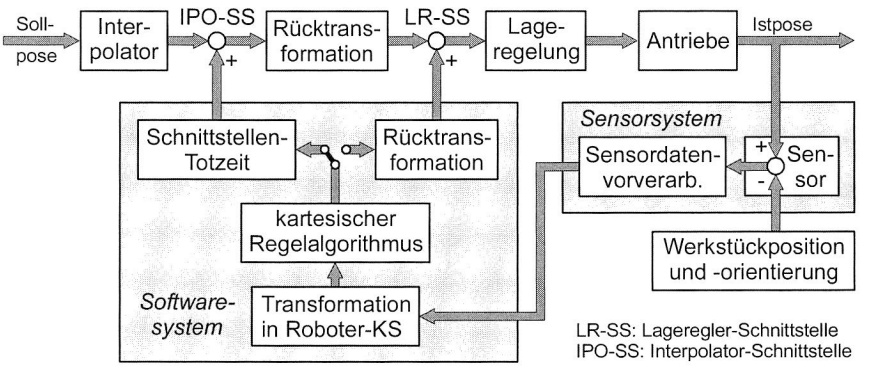


Bild 43: Blockschaltbild zu Realisierungsmöglichkeiten von Sensorregelkreisen

Die Verarbeitung der aufgenommenen Meßwerte verdeutlicht das Blockschaltbild nach Bild 43. Das Sensorsystem führt die Meßwertaufnahme und die anschließende Sensordatenvorverarbeitung durch. Die dafür erforderliche Zeitspanne wird als Sensortotzeit bezeichnet. Die Weiterverarbeitung der Sensordaten durch die Softwaremodule des Softwaresystems ist abhängig von der gewählten Integrationsschnittstelle der Robotersteuerung. Bei Nutzung der Lagereglerschnittstelle müssen die kartesisch vorliegenden Stellgrößen des Regelalgorithmus in axiale Korrekturwerte transformiert werden. Diese werden anschließend auf die Lagereglersollwerte aufgeschaltet. Dabei tritt eine Schnittstellentotzeit von einem Lageregeltakt auf. Diese ist jedoch im Vergleich zu der Sensortotzeit vernachlässigbar.

Bei vielen Robotersteuerungen ist die Lagereglerschnittstelle nicht offengelegt oder die benötigte Rücktransformation wird vom Steuerungshersteller nicht veröffentlicht. In diesen Fällen muß auf die Interpolatorschnittstelle zurückgegriffen werden. Nach der Transformation der Stellsignale des Regelalgorithmus in das Roboter-Koordinatensystem können die Korrektursignale erst im darauffolgenden Interpolationszyklus an die Robotersteuerung übertragen werden. Dies entspricht einem steuerungsbedingten

Totzeitbeitrag von der Zeitdauer eines Interpolationszykluses. Dieser ist um ein Vielfaches höher als die Schnittstellentotzeit bei der Lagereglerschnittstelle und darf deshalb nicht vernachlässigt werden.

Das Auftreten von Totzeiten während der Sensordatenverarbeitung und die Begrenztheit der Roboterdynamik führt zu einer relevanten Verzögerung zwischen der Meßwertaufnahme und den resultierenden Bahnkorrekturen. Aus diesem Grund werden Sensorregelkreise in der Regel nur bei Anwendungen eingesetzt, die geringe bis mittlere Bahngeschwindigkeiten erfordern. Außerdem ist zu beachten, daß die Korrekturwerte des Sensorreglers direkt auf die Sollwerte des Interpolators bzw. eines Lagereglers aufaddiert werden. Das Aufschalten großer Korrekturwerte führt zu einer Veränderung der programmierten Bahngeschwindigkeit. Bei Bearbeitungstechnologien, bei denen sich dies negativ auf das Bearbeitungsergebnis auswirkt, sollte deshalb darauf geachtet werden, daß die programmierte Bahn nicht zu stark von der idealen Sollbahn abweicht.

Steigen die Anforderungen an die Bahngeschwindigkeit, so kann das Werkzeug vom Roboter nicht mehr mit der geforderten Dynamik geführt werden. Stattdessen wird das Werkzeug an einer hochdynamischen Zusatzachse befestigt, die am Roboterflansch angebracht wurde. Ein typisches Beispiel für eine derartige Anwendung stellt die Regelung des Fokusabstandes beim Laserstrahlschneiden dar. Hier treten Bahngeschwindigkeiten bis 15 m/min auf. Die Korrekturwerte aus dem Sensorregelkreis dienen zur Ansteuerung der Zusatzachse. Der Roboter hingegen führt die Zusatzachse, und das daran angebrachte Werkzeug, entlang der programmierten Bahn. Betrachtet man den Vorgang im Frequenzbereich, so führt der Roboter die niederfrequenten Bahnänderungen aus und die Zusatzachse, aufgrund ihrer höheren Dynamik, die höherfrequenten Bahnänderungen [99].

Bahnplanung mit vorlaufendem Sensor

Die Problematik auftretender Sensor- und Steuerungstotzeiten und die eingeschränkte Dynamik herkömmlicher Industrieroboter begrenzen die Einsatzmöglichkeiten von Sensorregelkreisen. Diese Schwierigkeiten werden durch den Einsatz von vorlaufenden Sensoren vermieden. Die begrenzte Zugänglichkeit am Bearbeitungsort und prozeßspezifische Randbedingungen, wie z.B. das Prozeßleuchten bei Schweißanwendungen, sind weitere Gründe für die vorlaufende Anbringung von Sensoren.

Bei vorlaufenden Sensoren erfolgt die Meßwerterfassung nicht im Werkzeugeingriffspunkt. Vielmehr wird der Meßort in Verfahrrichtung des Werkzeugs verschoben. Die Sensormeßwerte beschreiben damit den zukünftigen Verlauf der Werkstückkontur. Durch die vorlaufende Erfassung der Sensormeßwerte können diese nicht sofort durch einen Regelalgorithmus verarbeitet werden. Vielmehr müssen diese für eine geeignete Zeitspanne gespeichert werden. Nach der Transformation der Sensormeßwerte in das

Roboterkoordinatensystem werden diese in einem FIFO (First-in-first-out)-Speicher zwischengespeichert (Bild 44). Die theoretisch erforderliche Zeitspanne für die Zwischenspeicherung der transformierten Sensormeßwerte hängt von der Länge des Vorlaufs und der Bahngeschwindigkeit ab. Verkürzt man diese Zeitspanne, so können die auftretenden Sensor- und Steuerungstotzeiten elegant kompensiert werden. Außerdem können die antriebsbedingten dynamischen Einschränkungen, durch das frühzeitige Aufschalten der Korrekturgrößen, vermindert werden.

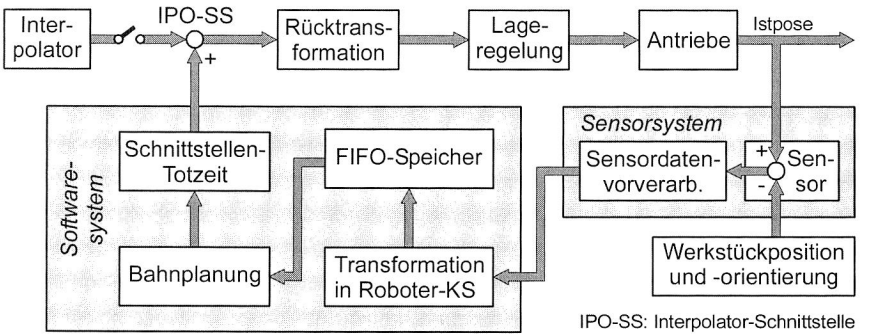


Bild 44: Blockschaltbild zur Bahnplanung mit vorlaufendem Sensor

Im Gegensatz zu einer Sensorregelung ist eine vorprogrammierte Bewegungsbahn nicht zwingend erforderlich. Mit Hilfe der vorlaufend aufgenommenen Sollposen, kann ein Bahnplanungsalgorithmus Bahnstützpunkte, im Zeitraster des Interpolationszykluses der Robotersteuerung, interpolieren. Nach der softwaretechnischen Deaktivierung des Bahninterpolators der Robotersteuerung, gibt der Bahnplanungsalgorithmus des Softwaresystems die Bahnstützpunkte vor. Die Programmierung der Bewegungsbahn beschränkt sich damit auf das Teachen des Bahnanfangspunktes. Der Sensor führt den Roboter dann selbstständig entlang der berechneten Bahnstützpunkte. Eine detaillierte Darstellung eines Bahnplanungsalgorithmus wird in Kapitel 7, am Beispiel einer Nahtfolgeanwendung, vorgestellt.

6.3.3 Softwaremodul zur Funktionssteuerung

Im Hinblick auf den Montage- bzw. Bearbeitungsprozeß, der von einer sensorgestützten Roboterzelle durchgeführt werden soll, sind zwei unterschiedliche Aufgabenstellungen denkbar. Die primäre Aufgabe besteht darin, die Bahn des Roboters derart zu steuern, daß das Werkzeug zum Werkstück so positioniert wird, wie es der umzusetzende Prozeß erfordert. Dies wird durch das Softwaremodul `bahnkorrektur` sichergestellt.

Neben der optimalen Positionierung des Werkzeugs ist auch die Steuerung der Werkzeugfunktionalität eine wesentliche Aufgabe des Softwaresystems. Dies umfaßt sowohl binäre Steuerdaten (z.B. Aktivieren bzw. Deaktivieren des Werkzeugs) als auch analoge Steuerdaten (z.B. Steuern der Leistung eines Bearbeitungslasers). Viele Prozesse erfordern zusätzlich zum eigentlichen Werkzeug noch weitere Peripheriegeräte, deren Funktionalität ebenso angesteuert werden muß. Beispiele hierfür sind eine Vorrichtung zur Drahtzufuhr bei Lötprozessen oder eine Vorrichtung zur automatisierten Zuführung von Schrauben für einen Schraubprozeß.

Die Steuerung der Funktionalität von Werkzeug und Peripheriegeräten muß mit der Bewegungsbahn des Roboters synchronisiert werden. Aufgrund der Bahnabhängigkeit der Steuerdaten könnten die Algorithmen zur Funktionssteuerung in das Softwaremodul `bahnkorrektur` integriert werden. Um die Komplexität dieses Softwaremoduls zu begrenzen und andererseits die Flexibilität des Softwaresystems zu steigern, wird die Funktionssteuerung durch ein eigenständiges Softwaremodul realisiert. Dieses trägt die allgemeine Bezeichnung `funktionssteuerung` (Bild 45).

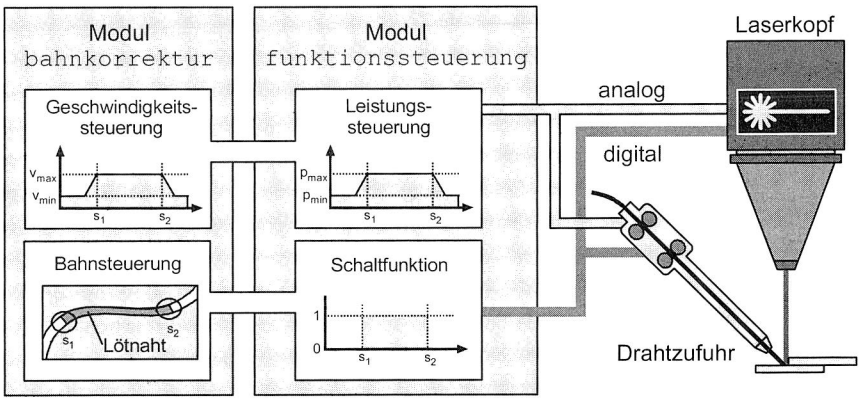


Bild 45: Steuerung von Bearbeitungswerkzeug und Peripheriegeräten durch das Modul `funktionssteuerung`

Die Synchronisation mit der Bahnkorrektur erfolgt über mehrere Variablen, die vom Modul `bahnkorrektur` übergeben werden. Die übergebenen Variablen können Werte zwischen null und einhundert Prozent annehmen. Binäre Schaltanweisungen, wie z.B. Aktivierung bzw. Deaktivierung eines Peripheriegerätes, werden durch die Variablenwerte null bzw. einhundert Prozent umgesetzt. Analoge Bahngrößen nutzen den gesamten Wertebereich. Häufig benötigte analoge Bahngrößen sind z.B. die aktuelle Bahngeschwindigkeit, Sensorwerte oder der prozentuale Anteil der bisher verfahrenen Bahnlänge an der insgesamt zu verfahrenden Bahnlänge. Die Belegung

der Variablen mit binären bzw. analogen Bahngrößen ist anwendungsabhängig und steht dem Modul `bahnkorrektur` prinzipiell frei. Durch Parametrierung legt der Anwender fest, welcher Variablenwert auf welche modulinterne Variable gemappt werden soll. Die Funktionssteuerung des Werkzeugs bzw. der Peripheriegeräte erfolgt jedoch nicht nur auf Basis von übergebenen Bahngrößen aus dem Modul zur Bahnkorrektur. Ebenso können interne Sensorwerte der angesteuerten Geräte oder weitere externe Sensormeßwerte verarbeitet werden. Diese werden von den bereits diskutierten Softwaremodulen zur Vorverarbeitung von Sensordaten bereit gestellt. Mit Hilfe dieser Sensorwerte können modulinterne Sensorregelkreise aufgebaut werden. Die vom Modul `funktionssteuerung` generierten Steuerdaten werden an das Softwaremodul `schreibeAA`, zum Beschreiben des Ausgangsabbildes des Feldbusses, übergeben. Der Feldbusmaster überträgt die Steuerdaten, über die Feldbusmodule, an die zu steuernden Einheiten.

Zum Abschluß dieses Kapitels zeigt Bild 46 einen Überblick über die anwendungsspezifischen Softwaremodule des eingebetteten Softwaresystems für die flexible Sensorführung von Robotern. Weiterhin ist der Informationsfluß zwischen den Softwaremodulen dargestellt. Dies verdeutlicht den sequentiellen Ablauf bei der Abarbeitung der Softwaremodule.

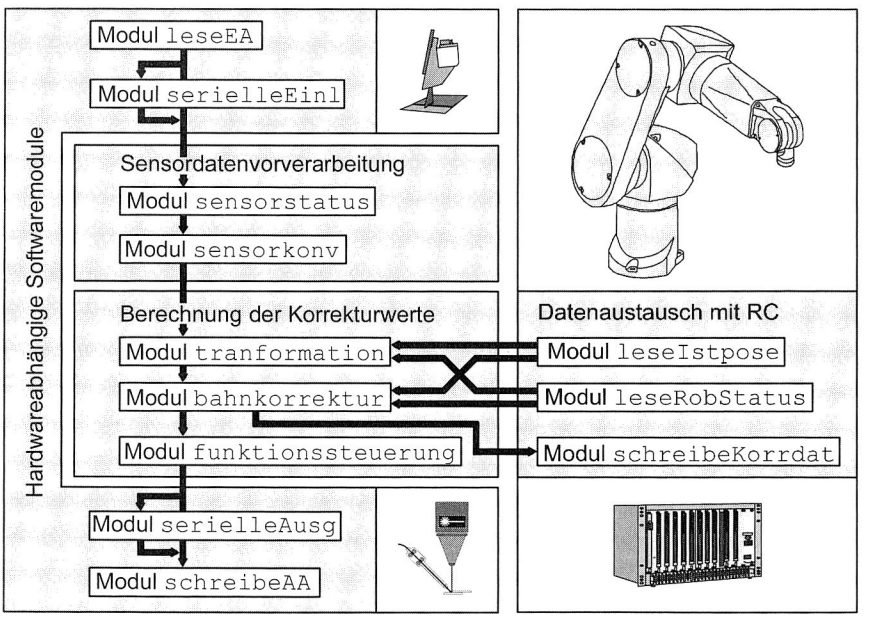


Bild 46: Übersicht über alle anwendungsspezifischen Softwaremodule

7 Entwicklung und simulationsgestützte Optimierung eines Nahtfolgealgorithmus

In dem vorangegangenen Kapitel wurden die anwendungsspezifischen Softwaremodule des konfigurierbaren Softwaresystems zur Sensorführung eines Roboters diskutiert. Die Anwendungsspezifik bezieht sich auf die Anpassung der Softwaremodule an die Anforderungen einer spezifischen Anwendung durch Parametrierung.

Ein Modul nimmt jedoch eine Sonderstellung ein. Es handelt sich um das Modul zur Berechnung der Bahnkorrekturwerte (Kapitel 6.3.2). Der Algorithmus zur Berechnung der Bahnkorrekturwerte hängt in einem wesentlich stärkeren Maße, als die anderen Module, von der jeweiligen Sensoranwendung ab. Beispielsweise unterscheidet sich eine Kraftregelung für Montageaufgaben strukturell wesentlich von einer Nahtfolgeregelung für die Materialbearbeitung. Eine Anpassung durch Parametrierung ist in diesen Fällen nicht möglich.

In diesem Kapitel wird die Realisierung eines Softwaremoduls zur Bahnkorrektur, am Beispiel einer Nahtfolgeanwendung, diskutiert. Neben den Entwicklungskonzepten wird auch auf Optimierung freier Parameter eingegangen. Sowohl bei der Entwicklung wie bei der Optimierung des Nahtfolgealgorithmus werden die Potentiale der Simulationstechnologie genutzt. Dieses Vorgehen ist auf andere Sensoranwendungen übertragbar und deshalb von allgemeiner Bedeutung.

7.1 Entwicklung eines Nahtfolgealgorithmus

Eine Nahtfolgeanwendung ist dadurch gekennzeichnet, daß ein Sensorsystem die relevante Fügekante auf dem Werkstück erfaßt. Ein Nahtfolgealgorithmus bestimmt aus den Meßwerten Korrekturwerte, die in die Robotersteuerung integriert werden. Die Korrektur der Bewegungsbahn führt dazu, daß der Werkzeugeingriffspunkt, während der Bearbeitung, eine geometrisch festgelegte relative Position und Orientierung zu der Fügekante einnimmt.

Für die durchgeführten Untersuchungen wurde die Interpolatorschnittstelle zur Integration der Korrekturwerte verwendet. Die Nutzung der dynamischeren Lagereglerschnittstelle war nicht möglich, da diese bei der verwendeten Robotersteuerung vom Hersteller nicht freigegeben wurde. Die Untersuchungen zeigten jedoch, daß die Interpolatorschnittstelle, unter dynamischen Gesichtspunkten, ausreichend ist.

Für die Entwicklung eines Nahtfolgealgorithmus ist von besonderer Bedeutung, daß der Sensor in Verfahrrichtung vorlaufend an das Bearbeitungswerkzeug montiert wird. Dieser Aspekt hat einen wesentlichen Einfluß auf die Entwicklung des Nahtfolgealgorithmus.

7.1.1 Regelungstechnische Bestimmung der Streckendynamik

Die Realisierung des Nahtfolgealgorithmus erfordert eine genaue Kenntnis über die dynamischen Eigenschaften der Regelstrecke. Den Eingang der Regelstrecke bildet die Integrationsschnittstelle für Korrekturdaten in der Robotersteuerung. Die von dem Sensorsystem erfaßte Position der zu folgenden Werkstückkante stellt das Ausgangssignal der Regelstrecke dar. Damit umfaßt die Regelstrecke das zeitliche Verhalten der Integrationsschnittstelle, die Dynamik der roboterinternen Regelkreise und das zeitliche Verhalten der Sensordatenerfassung. Zur Bestimmung der dynamischen Eigenschaften wurde die Sprungantwort aufgenommen. Die Aufnahme der Sprungantwort stellt ein regelungstechnisches Standardverfahren zur Bestimmung dynamischer Kenngrößen dar. Um eine bessere zeitliche Auflösung zu erhalten, wurde die Zykluszeit des Interpolationstaktes auf $T_{IPO} = 6\ \mathrm{ms}$ reduziert.

Zur experimentellen Bestimmung der Sprungantwort der Regelstrecke wurde eine Verfahrbahn programmiert, die den Werkzeugeingriffspunkt TCP in der Y-Z-Ebene des Roboterkoordinatensystems bewegt. An der Integrationsschnittstelle der Robotersteuerung wurde zu einem festgelegten Zeitpunkt die X-Komponente der Korrekturwerte sprungförmig auf x_{e0}=1 mm geändert. Bild 47 stellt die resultierende Sprungantwort und wesentliche Kenngrößen grafisch dar.

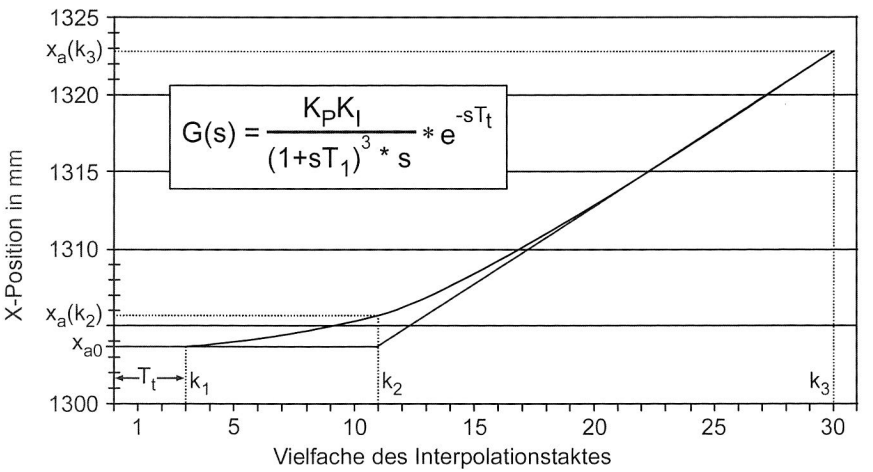


Bild 47: Verlauf der Sprungantwort der Regelstrecke G(s)

Aus der Sprungantwort sind die relevanten dynamischen Eigenschaften bestimmbar. Deutlich erkennbar ist die Totzeit T_t von drei Interpolationstakten und das integrierende Verhalten der Sprungantwort.

Mit Hilfe der in Bild 47 eingetragenen Größen läßt sich das Produkt aus Proportionalverstärkung K_P und Integrierverstärkung K_I nach Gl. 26 bestimmen.

$$K_P K_I = \frac{x_a(k_3) - x_{a0}}{x_{e0} \cdot (k_3 - k_2)} \qquad \text{Gl. 26}$$

Im Rahmen der Ablesegenauigkeit ergibt sich ein Wert von $K_P K_I = 1$. Dies bedeutet, daß die Robotersteuerung keine weitere Verstärkung der Korrekturwerte durchführt. Der Wert für $K_P K_I$ ist dimensionslos, da hier und im weiteren in Vielfachen des Interpolationstaktes gerechnet wird.

Zur Ermittlung der Anzahl der Verzögerungselemente wird die Hilfsgröße h gemäß der Gl. 27 bestimmt.

$$h = \frac{x_a(k_2) - x_{a0}}{x_{e0} \cdot K_P K_I \cdot (k_2 - k_1)} = \frac{(x_a(k_2) - x_{a0}) \cdot (k_3 - k_2)}{(x_a(k_3) - x_{a0}) \cdot (k_2 - k_1)} \qquad \text{Gl. 27}$$

Aus der gemessenen Sprungantwort erhält man für die Hilfsgröße h den Wert 0,24. Anhand von Tabelle 1 kann damit die Anzahl der Verzögerungselemente bestimmt werden.

n	1	2	3	4	5	6
h_{opt}	0,3679	0,2702	0,2240	0,1945	0,1755	0,1606

Tabelle 1: Bestimmung der Anzahl der Verzögerungselemente (nach [76])

Nach Tabelle 1 und [76] enthält die Übertragungsfunktion der Regelstrecke $n = 3$ identische Verzögerungszeitkonstanten. Dabei wurde derjenige Ordnungsgrad gewählt, für den die Hilfsgröße h dem optimalen Wert h_{opt} am nächsten kommt.

Die Bestimmung der Größe der Verzögerungszeitkonstanten T_1 erfolgt nach Gl. 28:

$$T_1 = \frac{(k_2 - k_1) \cdot T_{IPO}}{n} \qquad \text{Gl. 28}$$

Mit einer Interpolationszykluszeit von $T_{IPO} = 6\ \text{ms}$ erhält man eine Verzögerungszeitkonstante von $T_1 = 16\ \text{ms}$. Für die Bearbeitungsversuche wird eine Interpolationszykluszeit von $T_{IPO} = 12\ \text{ms}$ verwendet, da diese den Standardwert darstellt, mit dem die Robotersteuerung ausgeliefert wird. Durch die Verdoppelung der Interpolationszykluszeit verändert sich die Zeitkonstante T_1 nicht, da die Differenz $k_2 - k_1$ in diesem Fall halbiert wird.

Bei der Regelstrecke handelt es sich damit regelungstechnisch um ein totzeitbehaftetes I-T_3-System. Dafür soll in dieser Arbeit die Notation $\text{I-T}_3\text{-T}_t$-System eingeführt

werden. Dies bedeutet, daß ein Korrektursignal in der Steuerung aufsummiert wird und erst zeitlich verzögert zu einer Positionsänderung am TCP führt. In diesem Zusammenhang ist zu beachten, daß es sich, aufgrund des integralen Charakters der Regelstrecke, bei den aufzuschaltenden Korrektursignalen um Geschwindigkeitsänderungen und nicht um Positionsänderungen handeln muß.

Aus den gewonnen Parametern kann die Übertragungsfunktion gebildet werden. Im kontinuierlichen Zeitbereich wird sie durch Gl. 29 repräsentiert.

$$G(s) = K_P K_I \cdot \frac{1}{(1+sT_1)^3 \cdot s} \cdot e^{-sT_t} \qquad \text{Gl. 29}$$

Zur Verwendung in dem diskontinuierlich arbeitenden Softwaresystem muß diese, mit Hilfe der z-Transformation, in eine Differenzengleichung umgewandelt werden.

Den Vergleich zwischen der realen Sprungantwort und der theoretischen Sprungantwort eines modellierten $I\text{-}T_3\text{-}T_t$-Systems zeigt Bild 48.

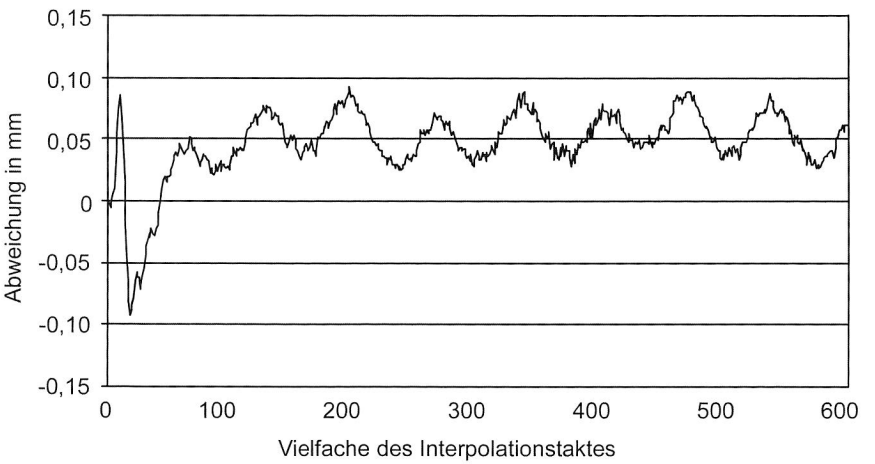


Bild 48: Abweichungen zwischen realer und modellierter Sprungantwort

In den ersten Zehntelsekunden nach der Sprungaufschaltung tritt eine Abweichung zwischen real gemessener und modellierter Sprungantwort von maximal ±0,10 mm auf. Diese Abweichungen sind auf Reversierbewegungen zurückzuführen, die immer beobachtet werden, wenn sich die Bewegungsbahn unstetig ändert. Im Bereich der Angleichung an die Asymptote tritt eine mittlere Abweichung von 0,06 mm mit einer Schwankungsbreite von ca. 0,03 mm auf. Dies zeigt, daß die reale Regelstrecke mit

guter Näherung modelliert wurde und das Modell für weitere Untersuchungen verwendet werden kann.

7.1.2 Lokale Bahnplanungsstrategie

Die online Sensorführung des Industrieroboters kann in zwei Betriebsarten realisiert werden. Bei der ersten Betriebsart werden die programmierten Bewegungssätze des Roboterprogramms, in Abhängigkeit von den Sensormeßwerten, korrigiert. Diese Betriebsart wird im nächsten Kapitel diskutiert. Die zweite Betriebsart ist dadurch charakterisiert, daß der Interpolator softwaretechnisch deaktiviert wird und die anzufahrenden Stützstellen ausschließlich durch das Softwaresystem generiert werden. Diese Betriebsart bezeichnet man als lokale Bahnplanung.

Der wesentliche Vorteil einer lokalen Bahnplanung durch das Softwaresystem besteht darin, daß nur der Anfangspunkt der sensorgeführten Bearbeitungsbahn durch den Bediener programmiert werden muß. Dies ist vor allem bei kleinen Losgrößen vorteilhaft. Die einzige Bedingung bei der Programmierung des Anfangspunktes besteht darin, daß sich die relevante Werkstückkante, welcher der Sensor folgen soll, im Erfassungsbereich des Sensors befindet. Treten Fehler bei der Aufspannung des Werkstücks auf, so kann diese Bedingung verletzt werden. Für diesen Fall kann eine Funktion zur Nahtfindung implementiert werden, die definierte Drehungen des Bearbeitungskopfes um den TCP durchführt, bis die Werkstückkante detektiert wird.

Die Aufgabe der lokalen Bahnplanung besteht im wesentlichen darin, die von dem Sensorsystem erfaßten geometrischen Bahninformationen nach einer Steuerstrategie in Korrekturframes umzusetzen und zum richtigen Zeitpunkt an die Steuerung zu übergeben.

Für die lokale Bahnplanung mit vorlaufendem Sensor wird ein Verfahren zur linearen Interpolation angewendet (Bild 49 links). Der Anfangspunkt der Interpolationsstrecke ist durch den aktuellen Werkzeugbezugspunkt festgelegt (Istposition). Die wesentliche Aufgabe der Steuerstrategie ist es, einen geeigneten Endpunkt festzulegen. Die dynamischen Untersuchungen der Regelstrecke haben gezeigt, daß Korrektursignale nicht ohne zeitlichen Verzug umgesetzt werden können. Es ist deshalb sinnvoll einen Endpunkt zu wählen, der wesentlich weiter entfernt auf der vom vorlaufenden Sensor bestimmten Sollbahn liegt, als in einem Interpolationstakt verfahren werden kann. Der Abstand des jeweiligen Endpunktes vom Werkzeugbezugspunkt wird als Vorlauflänge bezeichnet.

Ist die optimale Vorlauflänge bekannt, so kann damit als Endpunkt der Interpolation der entsprechende Bahnpunkt auf der Sollbahn bestimmt werden. Je nach parametrierter Bahngeschwindigkeit wird der nächste anzufahrende Stützpunkt auf der Interpolationsgeraden bestimmt und an die Robotersteuerung übergeben. Der Abstand zwischen

Werkzeugbezugspunkt und nächstem Stützwert wird als Schrittweite bezeichnet. Diese stellt das Produkt aus Bahngeschwindigkeit und Interpolatorzykluszeit dar.

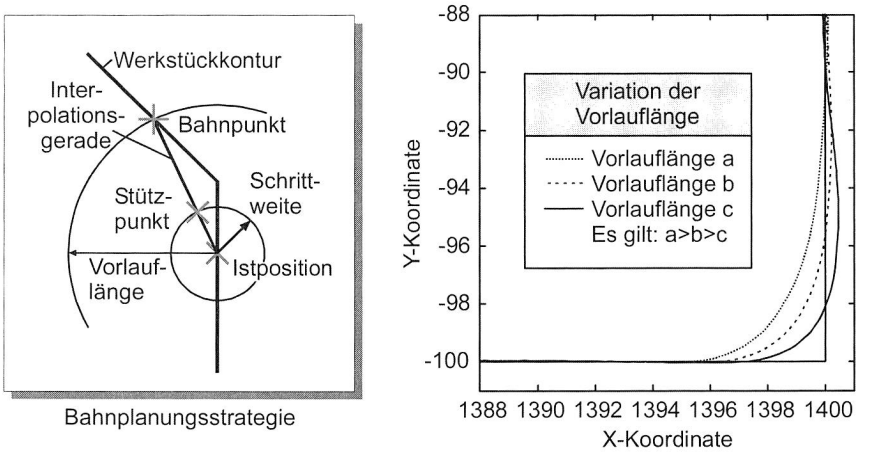


Bild 49: Darstellung der Bahnplanungsstrategie und Einfluß der Vorlauflänge

Der steuerungstechnische Vorteil einer Bahnplanungsstrategie liegt darin, daß die Informationen über den zukünftigen Verlauf der Sollbahn in Bezug auf die Dynamik der Regelstrecke optimal genutzt werden. Damit lassen sich die dynamisch bedingten Abweichungen zwischen Soll- und Istbahn deutlich reduzieren. Zur Verdeutlichung dieser Aussage wurde beispielhaft eine rechtwinklige, zweidimensionale Kontur mit der Bahnplanungsstrategie verfolgt. Eine derartig unstetige Sollbahn kann von einem Sensorsystem nicht erfaßt werden und wurde deshalb softwaretechnisch erzeugt. Bei einer rechtwinkligen Kontur treten die dynamischen Effekte jedoch besonders deutlich hervor. Das rechte Teilbild von Bild 49 zeigt den Einfluß der Vorlauflänge. Je größer die Vorlauflänge eingestellt wird, desto früher werden Orientierungsänderungen der Sollbahn berücksichtigt. Für die Minimierung der Bahnfehler ist die Optimierung der Vorlauflänge erforderlich. In Kapitel 7.3.1 wird dazu ein simulationsbasiertes Verfahren vorgestellt.

7.1.3 Korrektur interpolierter Stützwerte bei programmierten Bahnen

Ein Nachteil der lokalen Bahnplanung durch das Softwaresystem besteht darin, daß bei Ausfall des Sensorssystems die Bearbeitung nicht fortgesetzt werden kann. In diesem Fall steht die Produktion, bis das Sensorsystem ausgetauscht wurde. Bei verketteten Produktionslinien, wie sie häufig im Karosserierohbau zu finden sind, steht

im Fehlerfall die gesamte Produktionslinie. Das damit verbundene erhebliche wirtschaftliche Risiko ist bei derartigen Anwendungen nicht akzeptabel.

Aus diesem Grund wurde eine zweite Betriebsart zur Sensorführung realisiert. Dabei wird die Bewegungsbahn durch ein Roboterprogramm vorgegeben. Das Roboterprogramm kann manuell geteacht werden oder in einem Offline-Programmiersystem erzeugt werden [4]. Die vom Interpolator bestimmten Stützwerte werden gemäß der Sensormeßwerte korrigiert. Fällt das Sensorsystem aus, so kann immer noch die programmierte Bahn verfahren werden. Dies ist zwar meist mit einer Qualitätsminderung verbunden, doch die Produktion kann aufrecht erhalten werden. Das erforderliche Qualitätsniveau kann gegebenenfalls durch Nacharbeit wieder hergestellt werden.

Bei dieser Betriebsart wird derselbe Algorithmus, wie bei der lokalen Bahnplanung, zur Erzeugung der Korrekturwerte verwendet. Da bei dieser Betriebsart der Interpolator aktiviert ist, werden die Korrekturwerte jedoch auf die vom Interpolator gebildeten Stützwerte aufaddiert. Bei der Berechnung der Korrekturwerte muß deshalb der interpolierte Stützwert berücksichtigt werden (Bild 50). Konkret muß die jeweilige Stützwertkomponente von den entsprechenden Korrekturwerten abgezogen werden.

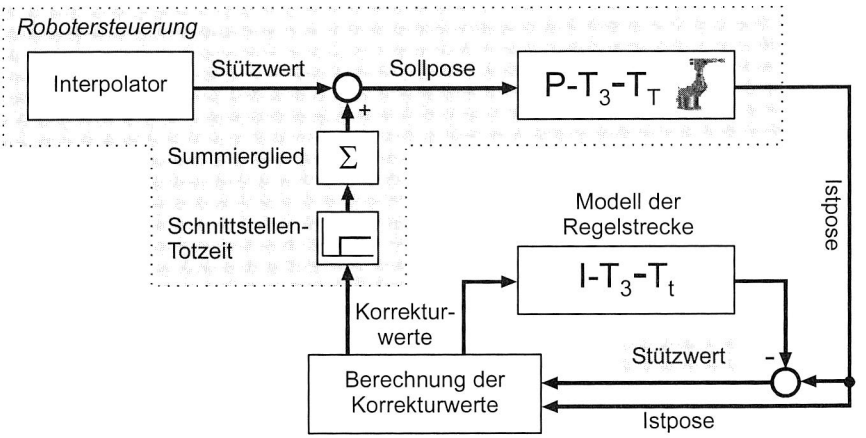


Bild 50: Berücksichtigung des Interpolatoranteils an der Sollpose

Eine Schwierigkeit bei dieser Betriebsart besteht darin, daß die Robotersteuerung den interpolierten Stützwert an der Sensorschnittstelle nicht zur Verfügung stellt. Mit Hilfe des dynamischen Modells der Regelstrecke kann nach Bild 50 der vom Interpolator berechnete Stützwert bestimmt werden. Dazu werden die an die Steuerung überge-

benen Korrekturwerte dem Modell der Regelstrecke zugeführt und das Ausgangssignal anschließend von der aktuellen Istpose subtrahiert.

Im Grunde werden bei diesem Verfahren die interpolierten Stützwerte komplett kompensiert und der Regelstrecke das gleiche Verhalten, wie bei der lokalen Bahnplanung aufgeprägt. Der Vorteil ist jedoch, daß bei Ausfall des Sensors kein Korrektursignal aufgeschalten wird und damit auch die Stützwerte nicht kompensiert werden. Ein weiterer Vorteil besteht darin, daß man das Verfahren nur auf die Positionskomponenten der Sollpose anwenden kann. Die interpolierten Orientierungskomponenten bleiben dadurch erhalten und werden nicht verändert. Dies ist vor allem dann sinnvoll, wenn die Gefahr von Kollisionen des Bearbeitungskopfes mit Werkstückteilen oder Spannmitteln besteht, die vom Sensorsystem nicht detektiert werden können. Durch das Roboterprogramm kann der Bearbeitungskopf geeignet orientiert werden, um Kollisionen zu vermeiden. Bei der Programmierung der Orientierungen ist jedoch darauf zu achten, daß sich die Werkstückkontur immer im Erfassungsbereich des Sensors befindet.

7.2 Aufbau einer Simulationsumgebung

Die Simulationstechnik ist eine Anwendung, die viel Rechenleistung erfordert. Früher wurden Hochleistungsworkstations benötigt, um z.B. eine Kinematiksimulation mit grafischer Animation in Realzeit berechnen zu können. Die rasante Entwicklung der Mikroelektronik ermöglicht es immer mehr Rechenleistung bei gleichzeitig sinkenden Preisen bereitzustellen. So ist es heute möglich, für viele Simulationsanwendungen herkömmliche Personal Computer einzusetzen. Aufgrund der stetig fallenden Investitionkosten setzen immer mehr Unternehmen Simulationstechnik ein. Mittlerweile entwickelt sich die Simulationstechnik vom Exoten zu einem Standardwerkzeug für Planungs-, Entwicklungs- und Optimierungsaufgaben [48].

7.2.1 Motivation für den Einsatz von Simulationstechnologie

Der Nutzen der Simulationstechnologie liegt in der schnellen und wirtschaftlichen Planung und Optimierung von Produktionsanlagen, Produkten und Prozessen. Die verschiedenen Betrachtungsebenen führen zu unterschiedlichen Simulationsmodellen, deren Abstraktionsgrad und Genauigkeit auf die jeweiligen Fragestellungen abgestimmt sind [50]. Die wichtigsten Simulationsarten in der Produktionstechnik sind in Bild 51 dargestellt. Die Ablauf- oder Systemsimulation dient der Abbildung von Produktionsanlagen, von der Montagezelle bis zur virtuellen Fabrik [85]. Bewegungsstudien werden mit der 3D-Kinematik-Simulation durchgeführt [103]. Die Funktionalität von Maschinen- und Zellensteuerungen läßt sich mit einer Steuerungssimulation überprüfen [22]. Für die Simulation der dynamischen Eigenschaften einer Maschine wird meist die Mehrkörpersimulation (MKS) eingesetzt [13]. Zur Simulation und

Optimierung von Bearbeitungsprozessen oder Maschinenbauteilen verwendet man numerische Simulationsmethoden, wie die Finite Elemente Methode [8].

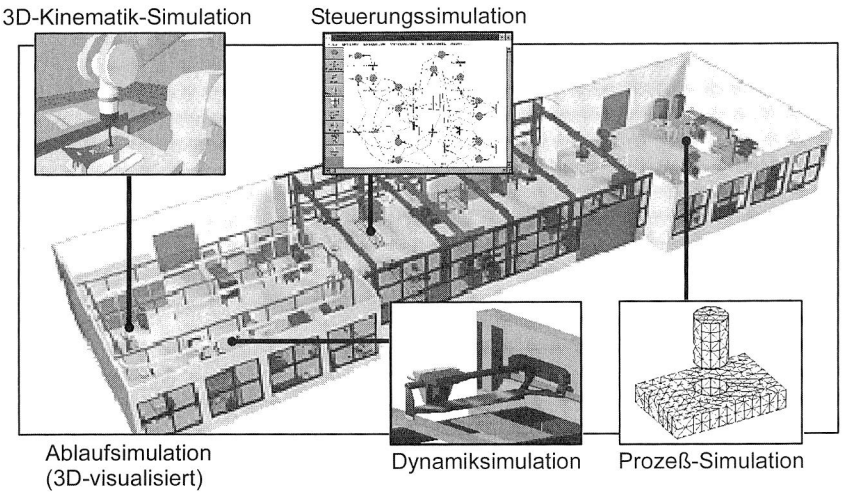


Bild 51: Unterschiedliche Simulationsarten in der Produktionstechnik [103]

Für die Entwicklung und Optimierung der Nahtfolgeregelung wurde in dieser Arbeit ein 3D-Kinematik-Simulationssystem eingesetzt. Wesentliche Hilfestellung leistete dieses Simulationssystem bei der Überprüfung der Funktionsfähigkeit der Transformationsalgorithmen für die Transformation der Sensormeßergebnisse in das Roboterkoordinatensystem. Vor allem die algorithmische Berücksichtigung der Mehrdeutigkeit der Tangensfunktion konnte damit, für dreidimensionale Anwendungsfälle, überprüft werden. Besondere Bedeutung hatte die Simulationsunterstützung jedoch für die Entwicklung und Optimierung des Nahtfolgealgorithmus. Die Entwicklung fand vollständig unter Nutzung des Simulationssystems statt. Dies hat den entscheidenden Vorteil, daß die Algorithmen in der Simulationsumgebung getestet werden konnten, bevor sie auf der realen Anlage zum Einsatz kamen. Eine Gefährdung der Anlage infolge von Kollisionen und die damit verbundene Beschädigung von Roboter oder Sensorsystem, infolge von algorithmischen Fehlern, kann damit weitgehend ausgeschlossen werden. Der größte Vorteil ergab sich jedoch für die Optimierung der freien Parameter. Beispielsweise hätte die Bestimmung der optimalen Vorlauflänge die Fertigung einer Vielzahl von Testkonturen erfordert. In der Simulationsumgebung beschränkt sich dies auf die Erzeugung einfacher CAD-Modelle. Die Bestimmung der

Einsatzgrenzen, in Bezug auf maximale Bahngeschwindigkeiten oder minimale Krümmungsradien, kann in der Simulation wesentlich schneller erfolgen.

Voraussetzung für die Durchführung ist jedoch die Modellierung der beteiligten Komponenten in dem Simulationssystem. Zum besseren Verständnis soll im folgenden kurz der Aufbau eines Kinematik-Simulationssystems vorgestellt werden.

7.2.2 Aufbau eines Kinematik-Simulationssystems

3D-Kinematik-Simulationsysteme werden für die Planung und Überprüfung des Zusammenbaus, der Kinematik und der Installation von technischen Anlagen eingesetzt. Sie ermöglichen die Simulation komplexer Bewegungen von Körpern und kinematischen Ketten und dient damit der Visualisierung kinematischer Abläufe.

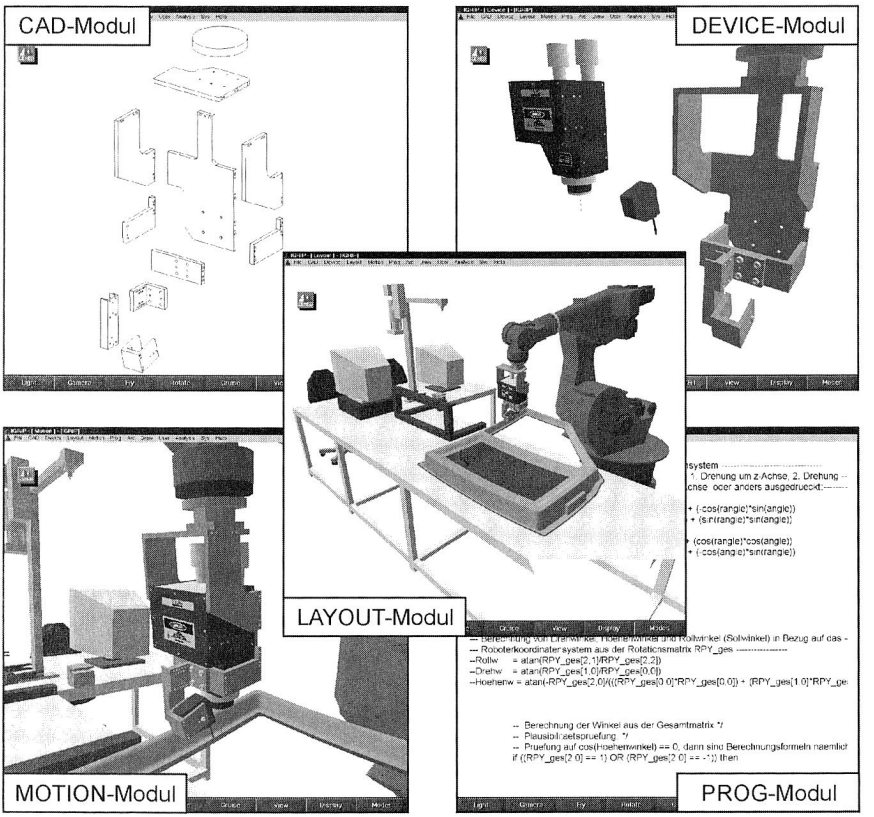


Bild 52: Funktionsmodule des 3D-Kinematik-Simulationssystems IGRIP

Daneben sind Kollisionsüberprüfungen und Abstandsuntersuchungen möglich. Da in der Simulation auch das Geschwindigkeits- und Beschleunigungsverhalten von Körpern berücksicht wird, können Aussagen über Takt- bzw. Durchlaufzeiten generiert werden [103]. Ein weiterer wichtiger Anwendungsbereich der Kinematik-Simulation ist die Offline-Programmierung von bewegungsgesteuerten Maschinen [4].

Das verwendete Kinematik-Simulationssystem IGRIP der Firma Deneb, ist modular aufgebaut. Bild 52 zeigt die fünf zur Verfügung stehenden Module.

Das CAD-Modul dient zur Generierung der Geometrie aller in der Simulation benötigten Bauteil-Modelle. Diese Modelle müssen in der Gestalt und den Abmessungen den realen Bauteilen exakt nachgebildet werden, um eine Übertragung der Simulationsergebnisse auf die reale Anlage zu gewährleisten. Zur Generierung der benötigten Geometrien stellt das CAD-Modul eine Reihe geometrischer Grundkörper wie Zylinder, Quader, Sphäre oder entlang einer Linie extrudierte Polygone zur Verfügung. Aus diesen können dann unterschiedlich komplexe Geometrien hergestellt werden. Das CAD-Modul bietet außerdem eine Reihe von Schnittstellen wie z.B. STL, IGES, STEP und VDA, über die Geometrien aus reinen CAD-Systemen importiert werden können. Für die aufzubauende Simulationsumgebung wurden Halterungen und Werkstücke aus dem CAD-Programm ProEngineer importiert.

Ein funktional abgeschlossener Körper wird als sog. Device bezeichnet. Ein Device besteht aus verschiedenen Teilkörpern, sog. Parts. Die verschiedenen Parts sind gegeneinander beweglich und bilden damit eine Kinematik. Im DEVICE-Modul werden die Bewegungsabhängigkeiten zwischen den Parts modelliert. Besonders komplexe Kinematiken, wie der verwendete Industrieroboter, werden in dem Kinematik-Simulationssystem bereitgestellt. Das Robotermodell verfügt ebenfalls über die Definition einer Inversen Kinematik. Damit ist es möglich, Bewegungen des TCP in die entsprechenden Achswinkel umzurechnen.

Im Modul LAYOUT werden die modellierten Geräte und Vorrichtungen zu einer Arbeitszelle zusammengesetzt. In diesem Modul können sechsdimensionale Frames, sog. Tag-Points, definiert werden. Diese werden zu einem Pfad zusammengefaßt und vom TCP des Robotermodells nacheinander abgefahren. Ein Tag-Point muß immer einem Part bzw. Device zugeordnet werden, da er sich auf dessen Koordinatensystem bezieht.

Nach der Erstellung der Arbeitszelle kann im Modul MOTION die Bewegung der modellierten Kinematiken erfolgen. Dazu werden zwei Möglichkeiten angeboten. Zum einen kann eine Kinematiksimulation innerhalb des Moduls MOTION durch das Abfahren vorher gesetzter Tag-Points erfolgen. Da die Bewegungsgeschwindigkeit eingestellt werden kann, können damit Laufzeituntersuchungen durchgeführt werden.

Eine andere Variante der Bewegungssimulation ist durch das Laden eines im Modul PROG erstellten Steuerungsprogrammes gegeben.

Das Modul PROG stellt einen Editor zur Erstellung eines Steuerungsprogrammes bereit. Das Steuerungsprogramm wird in der Syntax der GSL-Programmiersprache erstellt. GSL bietet neben den Befehlen einer klassischen Programmiersprache spezielle Befehle zur Kinematiksteuerung. Der GSL-Programmcode wird im Modul MOTION interpretierend verarbeitet.

7.2.3 Modellierung des Sensors

In einem Kinematik-Simulationssystem werden Bewegungssequenzen durch das Positionieren von Tag-Points im Arbeitsraum der entsprechenden Kinematik programmiert. Eine sensorgeführte Bewegungssteuerung ist nicht vorgesehen. Um eine derartige Bewegungssteuerung zu realisieren muß der Sensorkopf eines Sensorsystems nicht nur geometrisch, sondern auch funktional modelliert werden. Das verwendete Kinematik-Simulationssystem bietet eine Funktion, mit der jedoch nur abstandsmessende Sensoren modelliert werden können. Das Funktionsprinzip beruht darauf, daß von einem beliebigen Punkt im Raum, in eine definierte Richtung, ein imaginärer Meßstrahl ausgesendet wird. Trifft dieser Meßstrahl auf die Oberfläche eines Körpers, so wird die Kollision durch das Kinematik-Simulationssystem detektiert und der Abstand zum Ursprungspunkt des Meßstrahles als Funktionsergebnis zurückgegeben.

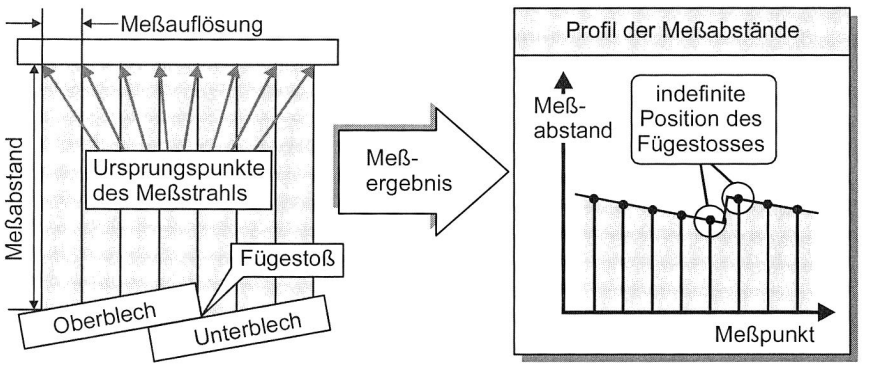


Bild 53: Funktionsprinzip des virtuellen Lichtschnittsensors

Diese Funktion kann genutzt werden, um einen Lichtschnittsensor funktional zu modellieren. Dazu wird der Ursprungspunkt des Meßstrahles auf der Schnittgeraden zwischen Meßebene des Sensors und Sensoroberfläche im Abstand der gewünschten lateralen Meßauflösung verschoben. Man erhält ein Abstandsprofil in der Meßebene

des Sensors (Bild 53). Um aus dem Abstandsprofil die Position des Fügestoßes zu ermitteln, wird das Verfahren der linearen Regression eingesetzt [5]. Die Achsen des orthogonalen, zweidimensionalen Sensorkoordinatensystems werden mit X und Y bezeichnet. Die Ursprungspunkte der Meßstrahlen liegen auf der X-Achse an den Positionen x_i. Zu jedem x_i gehört ein Abstandsmeßwert y_i. Die arithmetischen Mittel aus allen x_i bzw. y_i bezeichnet man mit $\bar{x}$ bzw. $\bar{y}$. Als Regressionsgerade bezeichnet man die Gerade, bei der die Summe der Abstände zwischen der Regressionsgeraden und den Abstandsmeßwerten y_i an den Stellen x_i minimal ist. Die Geradengleichung der Regressionsgeraden hat die allgemeine Form: $y = a_R + b_R \cdot x$.

Die Steigung b_R wird als Regressionskoeffizient bezeichnet und nach Gl. 30 berechnet.

$$b_R = \frac{\sum_i (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \qquad \text{Gl. 30}$$

Der Achsenabschnitt a_R wird gemäß Gl. 31 bestimmt.

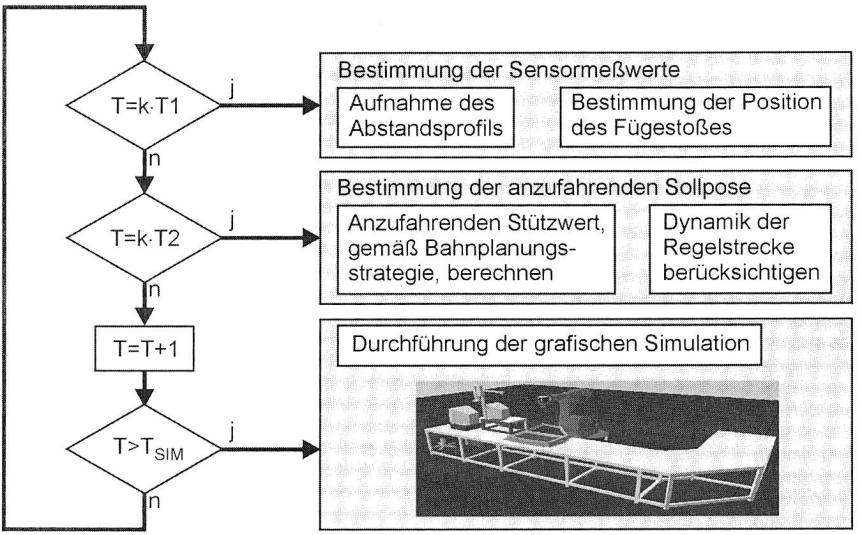
$$a_R = \bar{y} - b_R \cdot \bar{x} \qquad \text{Gl. 31}$$

Zur Bestimmung der Position des Fügestoßes wird die Regressionsgerade über die Meßpunkte x_0 bis x_n bestimmt. Anschließend wird die Summe der Abstandsbeträge zwischen Regressionsgerade und Abstandsmeßwerten bestimmt. Liegt die Summe unter einem vorgegebenen Maximalwert, so wird das Verfahren über die Meßpunkte x_1 bis x_{n+1} bestimmt. Das Verfahren wird solange wiederholt, bis der Maximalwert für die Summenbildung überschritten wird. In diesem Fall weicht der neu hinzugekommene Meßwert zu stark von der Regressionsgeraden ab. Dies ist immer am Fügestoß der Fall. Die Position des Fügestoßes ist damit bestimmt. Die x- und y-Komponenten des Fügestoßes werden als Sensormeßwerte an den Nahtfolgealgorithmus übergeben. Konnte der Fügestoß nicht ermittelt werden, so hat der Sensor die Naht verloren. Die Information über die erfolgreiche bzw. nicht erfolgreiche Bestimmung des Fügestoßes aus dem Abstandsprofil wird als Statusinformation des Sensors an weiterverarbeitende Module übergeben.

7.2.4 Implementierung der Softwaremodule

Die Softwaremodule des Softwaresystems zur flexiblen Sensorführung werden in dem Kinematik-Simulationssystem in der C-ähnlichen Programmiersprache GSL (engl. Graphic Simulation Language) implementiert. Auf Funktionalitäten, wie das dynami-

sches Laden der Softwaremodule, Konfigurationsmechanismen und Message-Queue-Mechanismen für die Datenübertragung wurde allerdings verzichtet.



T1: Zykluszeit des Sensorsystems T2: Zykluszeit des Interpolators

Bild 54: Grobstruktur des GSL-Programms zur Simulationssteuerung

In der realen Versuchsanlage werden die Meßergebnisse mit der Meßfrequenz des Sensorsystems geliefert, die Ablaufsteuerung wird mit dem Interpolationstakt der Robotersteuerung synchronisiert und die Zykluszeit des Interpolationstaktes wird in der Robotersteuerung über Timer generiert. In der Simulationsumgebung muß die zeitliche Abfolge zur Aktivierung der unterschiedlichen Funktionalitäten explizit programmiert werden. Dazu wird eine Schleifenstruktur implementiert (Bild 54). Nach jedem Schleifendurchlauf wird eine Zeitvariable, mit der Auflösung von einer Millisekunde, inkrementiert. Innerhalb der Schleifenstruktur wird überprüft, ob die Zeitvariable ein ganzzahliges Vielfaches der Periodendauer eines Funktionsmoduls annimmt. Ist dies der Fall, so wird dieses Funktionsmodul aktiviert. Ein Funktionsmodul implementiert die Sensorfunktionalität. Ein weiteres Funktionsmodul implementiert die lokale Bahnplanungsstrategie. Da in einem Kinematik-Simulationssystem dynamische Eigenschaften nicht berücksichtigt werden, wurde zusätzlich das Dynamikmodell der Regelstrecke implementiert.

Wurden die Berechnungen für die festgelegte Zeitdauer der Simulation abgeschlossen, so wird der Nahtfolgevorgang grafisch visualisiert.

7.3 Simulationsgestützte Optimierung des Nahtfolgealgorithmus

Mit der maßstabsgetreuen Modellierung der Versuchsanlage, der Implementierung des Nahtfolgealgorithmus und der Berücksichtigung der dynamischen Eigenschaften der Regelstrecke innerhalb des Kinematik-Simulationssystems sind die Voraussetzungen für eine wirklichkeitsnahe Simulation geschaffen worden. Die realisierte Simulationsumgebung soll dazu verwendet werden den Nahtfolgealgorithmus zu testen und zu optimieren und dessen Einsatzgrenzen zu bestimmen.

7.3.1 Bestimmung des optimalen Vorlaufs

In Kapitel 7.1.2 wurde im Rahmen der Bahnplanungsstrategie die Notwendigkeit eines Vorlaufes bei der Bestimmung der Bahnstützpunkte diskutiert. Die Bestimmung der optimalen Vorauflänge soll innerhalb des Kinematik-Simulationssystems erfolgen. Dazu wird dem Bahnplanungsalgorithmus eine rechtwinklige Sollkontur softwaretechnisch vorgegeben. Der Bahnplanungsalgorithmus bestimmt dann in Abhängigkeit von der eingestellten Vorauflänge und der vorgegebenen Bahngeschwindigkeit die an den virtuellen Roboter zu übergebenden Stützwerte. Die resultierende TCP-Bahn stellt sich dann in Abhängigkeit von der modellierten Dynamik der Regelstrecke ein.

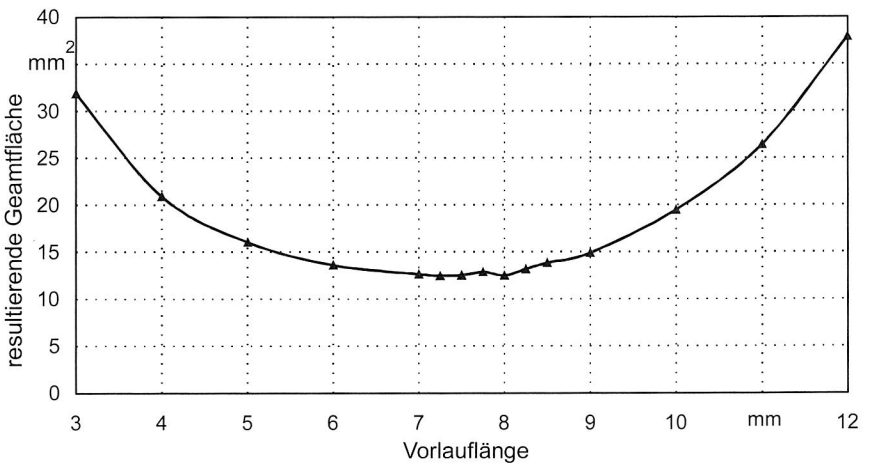


Bild 55: Abhängigkeit der resultierenden Gesamtfläche von der Vorauflänge bei einer Bahngeschwindigkeit von 3 m/min

Angestrebt wird eine möglichst gute Annäherung der Istbahn an die Sollbahn. Zur Bewertung der Ergebnisse wird ein Flächenkriterium verwendet. Das Optimierungskri-

terium besteht darin, die Flächenanteile zwischen der idealen rechtwinkligen Kontur und dem TCP-Bahnverlauf zu minimieren. Von den möglichen numerischen Integrationsformeln wird hier die Trapezregel verwendet [78]. Bei diesem Verfahren wird die Fläche unter einer Kurve durch eine Summe von Trapezflächen approximiert. Aus den in der Simulation berechneten und in einer Datei protokollierten Istpositionen berechnet ein speziell für diesen Anwendungsfall entwickeltes C-Programm die relevanten Flächenanteile.

In Bild 55 ist die Abhängigkeit der resultierenden Gesamtfläche von der Vorlauflänge, bei einer Bahngeschwindigkeit von 3 m/min, dargestellt. Für eine Vorlauflänge im Bereich von ca. 7,5 mm ist für die Gesamtfläche ein Minimum erkennbar. Diese Messung wird für unterschiedliche Bahngeschwindigkeiten durchgeführt. Es ergibt sich eine Abhängigkeit der optimalen Vorlauflänge von der Bahngeschwindigkeit. Nach Bild 56 ist der Zusammenhang nahezu linear.

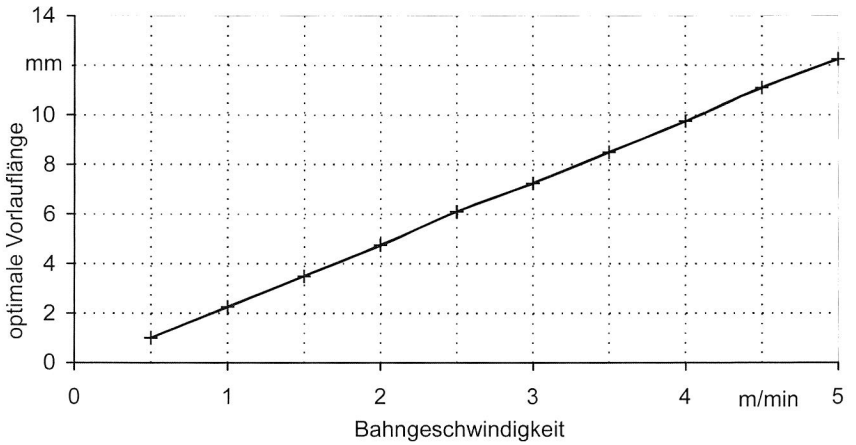


Bild 56: Abhängigkeit der optimalen Vorlauflänge von der Bahngeschwindigkeit

Aus den Ergebnissen läßt sich ein formelmäßiger Zusammenhang zwischen der optimalen Vorlauflänge RV_{opt} und der Bahngeschwindigkeit v_B, gemäß Gl. 32, herleiten.

$$RV_{opt} = 2{,}5 \cdot v_B - 0{,}25 \qquad \text{Gl. 32}$$

Die in der Simulation gewonnenen Ergebnisse wurden durch äquivalente Versuche in der realen Roboterzelle bestätigt.

7.3.2 Bestimmung des Dämpfungsfaktors

Beim Einsatz eines sensorgestützten Nahtfolgesystems ist von entscheidender Bedeutung, daß das Sensorsystem die zu verfolgende Werkstückkontur nicht verliert. Treten starke Orientierungsänderungen im Verlauf der Werkstückkontur auf, so muß der vorlaufend befestigte Sensorkopf in Richtung der Konturkrümmung gedreht werden. Die Drehung erfolgt dabei im Werkzeugkoordinatensystem um den TCP. Eine exakte Vermessung des TCP ist deshalb entscheidend für die Bearbeitungsqualität.

Die Drehwinkel α und β werden in Abhängigkeit von den Sensormeßwerten m_x und m_z bestimmt (Bild 57). Anschließend werden sie einer Transformation in das Roboterkoordinatensystem unterworfen. Die Übergabe der transformierten Drehwinkel an die Sensorschnittstelle der Robotersteuerung bewirkt eine Orientierungsänderung des Werkzeugeingriffspunktes TCP. Diese soll darin resultieren, daß der Mittelpunkt des Sensormeßfeldes mit dem Meßpunkt zur Deckung gebracht wird. Aufgrund der dynamischen Eigenschaften der Regelstrecke führt das direkte Aufschalten der transformierten Drehwinkel zum Auftreten von Schwingungen in der TCP-Orientierung.

Die berechneten Soll-Drehwinkel müssen deshalb über einen Dämpfungsfaktor reduziert werden. Die Einstellung des Dämpfungsfaktors ist kritisch. Wählt man diesen zu groß, so werden die Korrekturwerte für die TCP-Orientierung zu stark reduziert und der Sensor verliert die Werkstückkante bei starken Orientierungsänderungen der Werkstückkontur. Andererseits führt ein zu geringer Dämpfungsfaktor zu dem bereits erwähnten Aufschwingen der TCP-Orientierung.

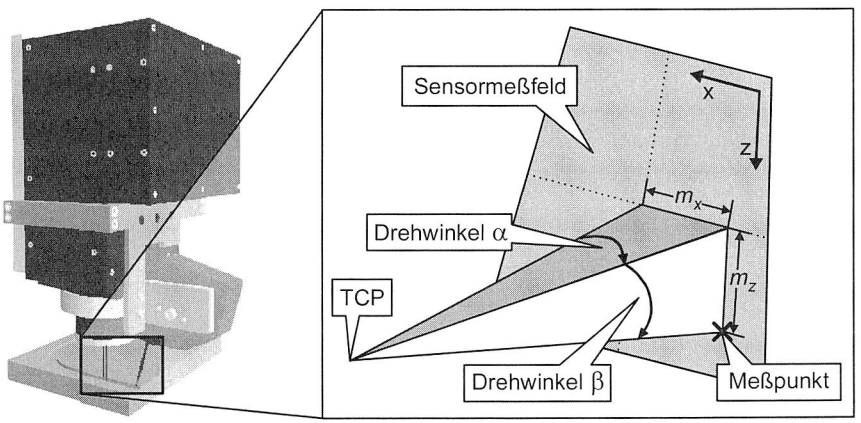


Bild 57: Darstellung der Drehwinkel zur Korrektur der TCP-Orientierung

Um diese Problemstellung zu lösen, wurde jeweils ein Schwellwert für die Sensormeßwerte m_x und m_z definiert. Sind die Beträge der Sensormeßwerte kleiner als der Schwellwert, so wird ein hoher Dämpfungsfaktor verwendet. Das bedeutet, daß kleine Orientierungsänderungen der Werkstückkontur zu keinen Korrekturen der TCP-Orientierung führen. Entsprechende Bahnabschnitte werden deutlich ruhiger verfahren.

Übersteigt der Betrag eines Sensormeßwerts den Schwellwert, so wird ein kleiner Dämpfungsfaktor verwendet. Damit ist ein schnelles Reagieren auf starke Orientierungsänderungen der Werkstückkontur möglich. Die Größe der jeweiligen Dämpfungsfaktoren wird experimentell bestimmt. Bild 58 zeigt beispielhaft den Verlauf des lateralen Sensormeßwertes m_x beim Folgen einer Werkstückkontur aus einem Viertelkreis mit einem Radius von 50 mm.

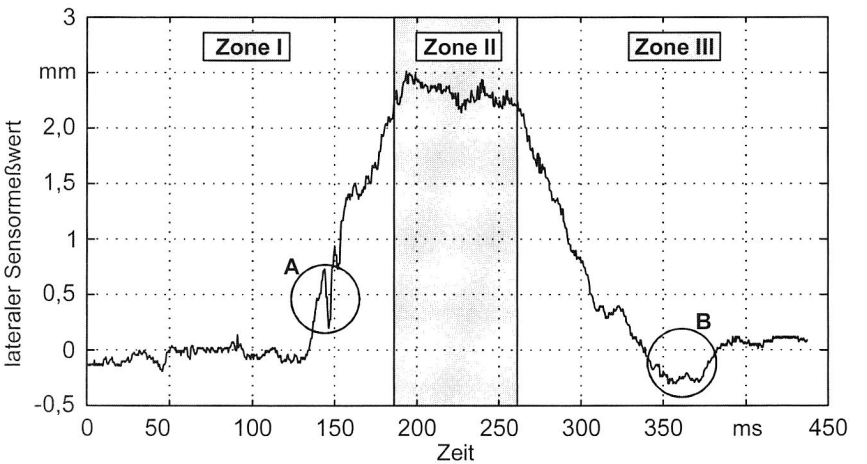


Bild 58: Verlauf des lateralen Sensormeßwertes beim Verfahren eines Viertelkreises mit Radius 50 mm

Im Verlauf des lateralen Sensormeßwertes sind drei Zonen zu unterscheiden. Zone I beschreibt das Einfahren in den Kreisbogen. Der laterale Sensormeßwert nimmt stetig zu, bis die Winkelgeschwindigkeit, mit der die TCP-Orientierung korrigiert wird, den theoretisch erforderlichen Wert erreicht hat. Der theoretisch erforderliche Wert hängt ab von dem Radius und der Bahngeschwindigkeit. Der Verlauf des lateralen Sensormeßwertes bewegt sich dann in Zone II. Verläßt der Bahnverlauf den Kreisbogen und geht wieder in eine Gerade über, so beginnt Zone III. Der laterale Sensormeßwert wird kleiner und damit wird auch die Winkelgeschwindigkeit der TCP-Orientierung geringer.

Besondere Beachtung erfordern die im Verlauf markierten Punkte A und B. Im Punkt A wird die Orientierungsdämpfung von einem hohen Dämpfungsfaktor sprungförmig auf einen geringen Dämpfungsfaktor umgeschalten. Dies führt zu einer sprungförmigen Änderung des lateralen Sensormeßwertes. Ein kontinuierlicher Übergang des Dämpfungsfaktor kann das sprungförmige Verhalten des Sensormeßwertes im Übergangsbereich etwas glätten. Untersuchungen haben jedoch gezeigt, das dann die Winkelgeschwindigkeit der TCP-Orientierung nicht schnell genug ansteigt und der Sensor die Werkstückkontur verliert. Aus diesem Grund wurde die sprungförmige Änderung des Dämpfungsfaktors beibehalten.

Das starke Unterschwingen des Meßwertverlaufs in Punkt B ist auf den hohen Dämpfungsfaktor zurückzuführen, der bei geringen Sensormeßwerten verwendet wird. Trotz des problematischen Verhaltens in den Punkten A und B, hat sich die gewählte Strategie für die Anpassung des Dämpfungsfaktors in vielen Versuchsreihen als die beste Lösung bewährt.

7.3.3 Bestimmung der Einsatzgrenzen

Für den Einsatz des Nahtfolgealgorithmus in realen Anwendungen ist es erforderlich die Einsatzgrenzen zu kennen. Bei industriellen Anwendungen ist vor allem die maximal erreichbare Bahngeschwindigkeit beim Auftreten von kreisförmigen und unstetigen Konturänderungen von Interesse, da diese die Bearbeitungszeit direkt beeinflußt.

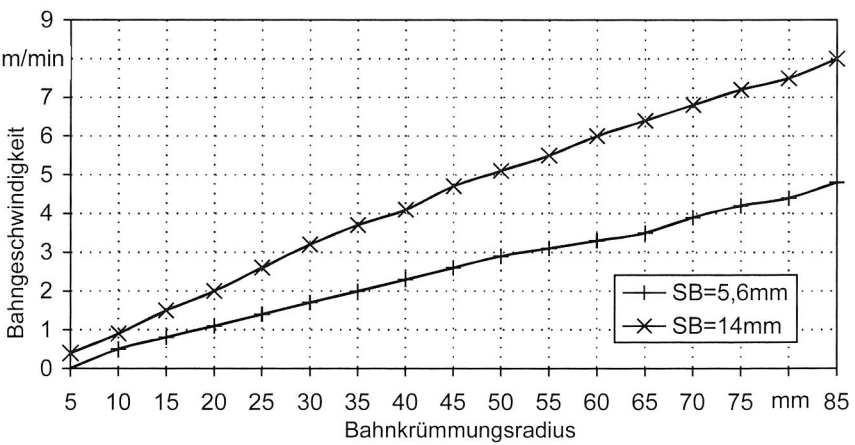


Bild 59: Abhängigkeit der maximal erzielbaren Bahngeschwindigkeit von dem Bahnkrümmungsradius bei unterschiedlichen Sensorfeldbreiten

Die Simulationsuntersuchungen wurden jeweils für die Sensorfeldbreite SB=5,6 mm des verwendeten realen Sensorsystems und alternativ für eine Sensorfeldbreite von SB=14 mm, wie sie von Konkurrenzprodukten verwendet wird, durchgeführt. Bild 59 zeigt die Abhängigkeit der maximal erzielbaren Bahngeschwindigkeit für unterschiedliche Bahnkrümmungsradien und Sensorfeldbreiten. Es ist ein näherungsweise linearer Zusammenhang erkennbar. Außerdem wird deutlich, das die Verbreiterung des Sensorfeldes eine deutliche Erhöhung der maximalen Bahngeschwindigkeit ermöglicht.

Die selben Untersuchungen wurden für unstetige Konturänderungen, in Form von zwei Geraden, die zueinander einen einstellbaren Bahnknickwinkel bilden, durchgeführt. In industriellen Anwendungen kommt diese Konturform sehr häufig vor. Bild 60 zeigt die Simulationsergebnisse.

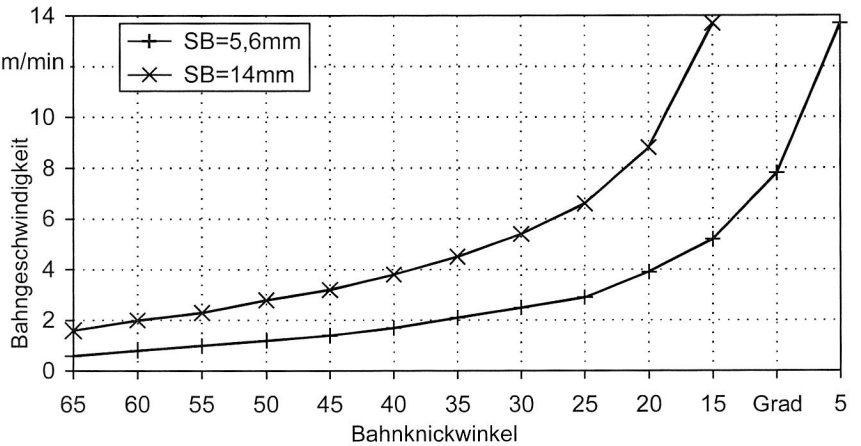


Bild 60: Abhängigkeit der maximal erzielbaren Bahngeschwindigkeit von dem Knickwinkel bei unterschiedlichen Sensorfeldbreiten

Auch hier wird wieder deutlich, das ein breiteres Sensorfeld höhere Bahngeschwindigkeiten ermöglicht. Besonders bei geringen Bahnknickwinkeln, wie sie näherungsweise beim Folgen von Dachnähten im Automobilrohbau auftreten, werden die Vorteile eines breiten Sensorfeldes deutlich.

8 Beispielhafte Umsetzung einer robotergestützten Nahtfolgeanwendung

Eine wesentliche Zielsetzung dieser Arbeit ist es, einen Beitrag zu leisten, um den zukünftig immer wichtiger werdenden Sensoreinsatz in roboterbasierten Bearbeitungszellen weiter voran zu treiben. Um dieses Ziel zu erreichen, ist es notwendig sich nicht nur auf konzeptionelle Arbeiten zu beschränken. Die angestrebte industrielle Umsetzung der konzeptionellen Inhalte dieser Arbeit erfordert den Nachweis der Industrietauglichkeit der entwickelten Systemarchitektur. Dazu ist es notwendig einen anspruchvollen industriellen Bearbeitungsprozeß auf Grundlage der Systemkonzepte zu realisieren. Um dies zu ermöglichen, wurde eine Versuchsanlage aufgebaut, die die Leistungsfähigkeit der vorgestellten Konzepte überzeugend verdeutlichen soll.

8.1 Komponenten der realisierten Versuchsanlage

Als eine der anspruchsvollsten robotergestützten Bearbeitungstechnologien gilt die Lasermaterialbearbeitung. Diese stellt im industriellen Umfeld derzeit die höchsten Anforderungen sowohl an die Robotertechnik, als auch an die Sensortechnik [114]. Die Bedeutung lasergestützter Fügeverfahren nimmt vor allem im Automobilbau stark zu. Die unvermeidbaren Fertigungstoleranzen werden häufig noch mit teurer Spanntechnik kompensiert, da sensorgestützte Verfahren bisher sehr aufwendig zu realisieren waren. Dies liegt im wesentlichen an dem hohen Engineeringaufwand, der mit der Integration eines Nahtfolgesystems in eine Roboterzelle verbunden ist. Ein konfigurierbares Softwaresystem bietet das Potential diese Problematik zu lösen [38]. Zur Qualifizierung des entwickelten Softwaresystems wurde deshalb eine Anwendung zum lasergestützten Hartlöten, unter Nutzung einer Nahtfolgesensorik, ausgewählt.

In Zusammenarbeit mit namhaften Unternehmen aus den Bereichen Robotik, Sensorik und Automobilbau wurde im Rahmen eines Teilprojektes des Bayerischen Forschungsverbundes FORLAS eine entsprechende Versuchsanlage aufgebaut [114]. Diese besteht aus den folgenden Komponenten:

- Standardindustrieroboter als Führungsmaschine
- Hochleistungsdiodenlaser als Strahlquelle
- Drahtzufuhrsystem für drahtförmiges Lot
- Nahtfolgesensorik auf Basis des Lichtschnittverfahrens
- Sensorrechner zur Verarbeitung der Sensordaten.

Bild 61 zeigt den realisierten Bearbeitungskopf mit Hochleistungsdiodenlaser, Nahtfolgesensor und Drahtzufuhrsystem.

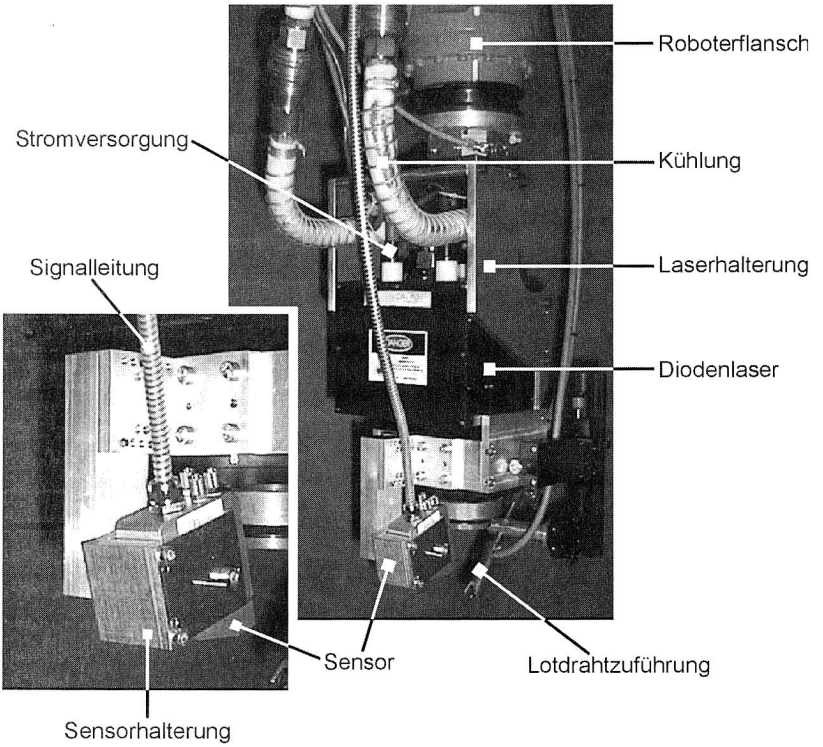


Bild 61: Aufbau des Bearbeitungskopfes mit adaptiertem Lichtschnittsensor

Im folgenden werden die einzelnen Komponenten kurz vorgestellt. Die zentralen Komponenten, wie Sensorsystem und Sensorrechner werden im Anschluß daran in separaten Kapiteln ausführlich diskutiert.

8.1.1 Standardindustrieroboter als Führungsmaschine

In Laseranlagen werden häufig Portalanlagen als Führungsmaschinen eingesetzt. Portalanlagen zeichnen sich durch eine hohe Steifigkeit im mechanischen Aufbau und einer damit verbundenen hohen erreichbaren Dynamik und Wiederholgenauigkeit aus. Diese Vorteile müssen jedoch mit hohen Investitionskosten erkauft werden. Ein wesentliches Teilziel dieses Projektes war es deshalb einen preisgünstigen Standard-

industrieroboter als Führungsmaschine zu qualifizieren. Dies stellt besonders hohe Anforderungen an die Qualität der Sensorik und des Sensorregelkreises.

Zur Durchführung dieses Forschungsprojektes stand ein Industrieroboter der KUKA Roboter GmbH, Augsburg, zur Verfügung. Dieser verfügt über eine Wiederholgenauigkeit von <±0,15 mm bei einer Traglast von 30 kg.

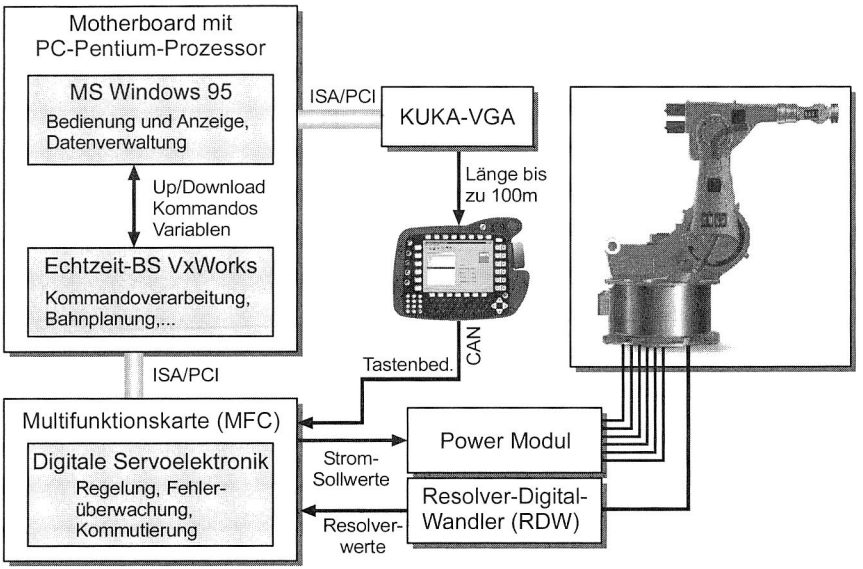


Bild 62: Steuerungsstruktur der PC-basierten Robotersteuerung (nach [105])

Die Steuerung ist PC-basiert und damit in Hinblick auf Hardware- sowie Softwareerweiterungen offen [105]. Den Kern der Steuerung bildet ein Standard-PC-Motherboard (Bild 62). Die Ansteuerung der Antriebe erfolgt über eine spezielle Einsteckkarte. Auf dieser Einsteckkarte befindet sich außerdem eine Elektronik, die es erlaubt auf der CPU zwei Betriebssysteme abwechselnd zu betreiben. Für die Programmierung und Visualisierung wird Windows 95 eingesetzt, für die zeitkritischen Funktionen Bahnplanung, Interpolation und Lageregelung wird das Echtzeitbetriebssystem VxWorks von Wind River genutzt. Dabei hat das Betriebssystem VxWorks absolute Priorität. Hat VxWorks seine Aufgaben, innerhalb eines Interpolationstaktes, abgeschlossen, so bleibt noch genügend Zeit für Windows zur Bedienung der Mensch-Maschine-Schnittstelle. Damit steht eine Steuerung zur Verfügung, die die Vorteile von Windows als Benutzerschnittstelle mit hartem Echtzeitverhalten verbindet.

8.1.2 Hochleistungsdiodenlaser als Strahlquelle

Beim Diodenlaser wird die Laserstrahlung direkt durch den Stromdurchgang durch einen Halbleiter erzeugt [111]. Die eingesetzten Halbleiterbauelemente sind Laserdioden [24]. Die Wellenlänge der emittierten Strahlung ist vom Halbleitermaterial abhängig. Mehrere dieser Laserdioden werden zu einem sogenannten Laserbarren kombiniert, der auf einen Kühlkörper aufgebracht wird (Bild 63). Die heute erreichbaren Ausgangsleistungen eines solchen Barrens liegen bei 30-50 Watt [58]. Durch eine geeignete Stapelung dieser Barren zu sogenannten Stacks addieren sich die Laserleistungen durch optische Überlagerung der einzelnen Strahlenbündel. Durch Kombination von mehreren Stacks lassen sich Ausgangsleistungen von bis zu 6000 Watt im Wellenlängenbereich von 808 nm bis 980 nm erzielen [58]. Eine Laserstrahlquelle auf Basis dieses Funktionsprinzips bezeichnet man als Hochleistungs-Diodenlaser (HDL).

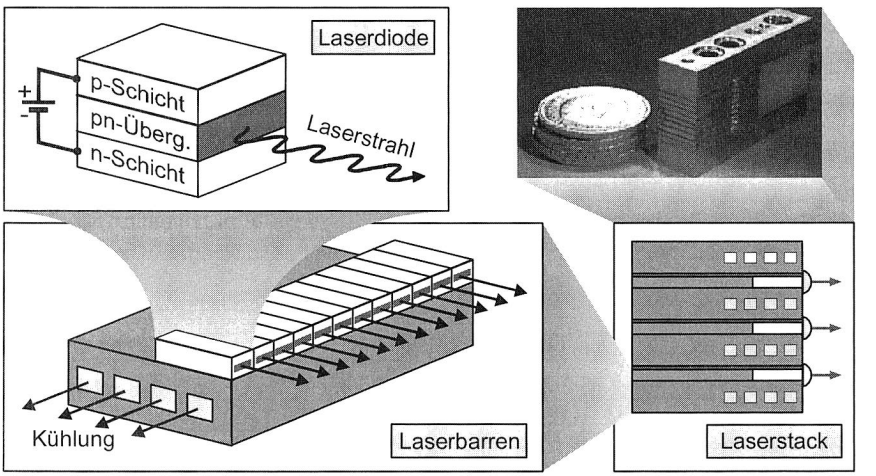


Bild 63: Schematischer Aufbau eines Laserstacks (Bildquelle [58])

Die Verwendung eines Diodenlasers als Strahlquelle erlaubt ein völlig neuartiges Systemkonzept, bei dem der Laser direkt an der Roboterhand angebracht ist. Da hier sowohl auf ein in die Führungsmaschine integriertes als auch auf ein externes Strahlführungssystem verzichtet werden kann, werden im Vergleich zu konventionellen Lasersystemen eine verbesserte Zugänglichkeit und Beweglichkeit, eine höhere Dynamik und ein robusteres Einsatzverhalten ermöglicht. Darüber hinaus weist der Diodenlaser, durch die kürzere Wellenlänge und dem damit verbundenen höheren Absorptionsvermögen auf metallischen Oberflächen, einen höheren Wirkungsgrad (30%-40%) gegenüber Nd:Yag- und CO_2-Lasern auf [72]. Aufgrund der gegenüber

anderen Materialbearbeitungslasern größeren Fokusabmessungen des Diodenlaserstrahls und der zu Projektbeginn verfügbaren Laserleistung von maximal 1,5 kW, wurde als Fügeverfahren das Laserstrahlhartlöten ausgewählt. Durch den Einsatz eines stoffschlüssigen Fügeverfahrens können aufwendige Schraubverbindungen, die ein Handling von zusätzlichen Verbindungselementen erfordern oder Klebeverbindungen, die ein Reinigen der Fügepartner erfordern, entfallen und eine höhere Steifigkeit und Festigkeit der Fügestelle erzielt werden.

8.1.3 Drahtzufuhrsystem für drahtförmiges Lot

Der Lötprozeß erfordert einen Zusatzwerkstoff, das Lot. Der Lotdraht muß direkt in die Fügezone zugeführt werden. Dazu wird ein Drahtzufuhrsystem eingesetzt. Dieses besteht aus einem Antriebsteil und einem Steuerteil. Der Antriebsteil richtet den Draht und fördert ihn zur Fügezone. Um ein Durchbiegen des Drahtes zu vermeiden, wird der Draht dabei in einem Metallröhrchen geführt. Der Steuerteil des Drahtzufuhrsystems erlaubt die Regelung der Drahtvorschubgeschwindigkeit. Über digitale Steuersignale vom Sensorrechner wird die Drahtzufuhr gestartet bzw. beendet.

8.2 Sensorsystem zur Nahtfolge

Im Karosserierohbau treten vor allem auf der Werkstückseite relevante Fertigungstoleranzen auf. Diese sind zu einem großen Teil auf die vorgelagerten Umformprozesse zurückzuführen. Zu nennen sind hier beispielsweise Rückfedereffekte beim Tiefziehprozeß, die in Maßtoleranzen des Werkstücks resultieren. Werkzeugverschleiß und Lagefehler führen bei den anschließenden Stanzprozessen zu Maßabweichungen von Werkstückkonturen. Außerdem handelt es sich bei den Karosseriebaugruppen um Leichtbauteile, die bei der Handhabung zusätzlichen Biegeeinflüssen unterliegen. Ein weiterer Fehlereinflußfaktor sind Lagefehler beim Aufspannen des Werkstücks vor der Bearbeitung.

Die auftretenden Werkstücktoleranzen sind unvereinbar mit den hohen Genauigkeitsanforderungen die ein Bearbeitungsprozeß, wie die Lasermaterialbearbeitung, erfordert. Aufgrund der lasertypischen punktförmigen Energieeinbringung, muß eine Positioniergenauigkeit zwischen Werkzeug und Werkstück im Bereich weniger Zehntelmillimeter sichergestellt werden. Diese Anforderung ist nur mit einem Sensorsystem realisierbar.

8.2.1 Aufbau und Integration des Sensorsystems

Für die Untersuchungen wurde von der Firma Jurca Optoelektronik GmbH, Rodgau, ein Nahtfolgesensor zur Verfügung gestellt. Dieser Sensor arbeitet nach dem Lichtschnittverfahren. Dabei wird eine Laserlinie binormal auf die Nahtfuge projiziert. Das Abbild der projizierten Laserlinie wird unter einem definierten Winkel von einem photo-

sensitiven CMOS-Chip erfaßt. Die Meßfrequenz liegt bei 60 Hz. Der Meßbereich liegt für den Abstandswert zwischen 45 mm und 55 mm und für den Seitenversatz bei ±3 mm. Das Videosignal des Sensorkopfes wird auf einer Videosignalprozessorkarte verarbeitet. Die Bildverarbeitung bestimmt aus dem Videosignal die zweidimensionale Position der Fügekante im Sensorkoordinatensystem. Außerdem wird die Steigung von Ober- und Unterblech im Bezug zum Sensorkoordinatensystem bestimmt. Damit stehen für jede detektierte Fügekante zwei Positionswerte und ein Orientierungswert zur Verfügung. Neben diesen Werten liefert die Bildverarbeitung noch wichtige Informationen zum Fehlerstatus des Sensorsystems.

Während der Inbetriebnahmephase ist es unbedingt erforderlich das Kamerabild und die Bildverarbeitungsergebnisse zu visualisieren, um die Bildverarbeitungsparameter optimal einstellen zu können. Um dies zu erreichen, wurde eine benutzerfreundliche Bedienoberfläche entwickelt. Diese Oberfläche stellt in verschiedenen Fenstern das von der Kamera gelieferte Videosignal und das daraus bestimmte Kantenprofil dar. Weiterhin werden die vorverarbeiteten Sensorwerte ausgegeben und eine Menüstruktur zur Parametrierung der Bildverarbeitungssoftware bereitgestellt.

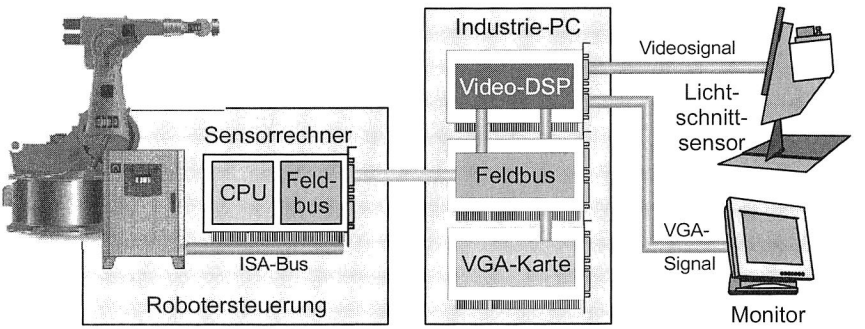


Bild 64: Anbindung des Lichtschnittsensors an den Sensorrechner

Die Video-DSP-Karte verfügt über einen ISA-Busanschluß, so daß sie in einen beliebigen PC bzw. in die verwendete PC-basierte Robotersteuerung eingebaut werden kann. Problematisch ist allerdings, daß die Karte über keinen eigenen VGA-Chip verfügt. Es muß deshalb auf die VGA-Karte des Host-PC zurückgegriffen werden. Um das Kamerabild in der Visualisierungsoberfläche darstellen zu können, muß der Bildspeicher der Video-DSP-Karte ausgelesen werden und in den VGA-Speicher übertragen werden. Dies ist über den ISA-Bus nicht in Echtzeit realisierbar. Zur Lösung dieses Problems wurde das sogenannte "Color-Keying"-Verfahren eingesetzt. Dabei wird auf der Visualisierungsoberfläche ein rechteckförmiger Bereich, der für die Darstellung des Kamerabildes vorgesehen ist, mit einer sonst auf der Oberfläche nicht

vorkommenden Farbe ausgefüllt. Das VGA-Signal der Grafikkarte wird über einen Feature-Connector an das DSP-Board übertragen. Der Video-DSP kopiert anschließend das Kamerabild in den farblich markierten Bereich. Über den VGA-anschluß des Video-DSP-Boards kann das vollständige Bild der Visualisierungsoberfläche auf einem Monitor ausgegeben werden (Bild 64).

Die Information über die Position der Kante wird über eine Feldbusschnittstelle an die Sensordatenverarbeitungseinheit (Sensorrechner) übertragen. Diese Positionsinformation ist zunächst ein zweidimensionaler Vektor im Sensorkoordinatensystem. Diese Information wird in das Hand-Werkzeug-Koordinatensystem (TCP-KS) transformiert. Die erforderlichen Transformationsparameter sind konstant und ausschließlich durch die Montage des Sensors am Werkzeug definiert. Diese Parameter können direkt aus Montagezeichnungen bzw. durch einen Kalibriervorgang ermittelt werden.

8.2.2 Kalibrierung des Lichtschnittsensors

Mit der Verfügbarkeit einer Visualisierungsoberfläche kann der Nahtfolgesensor kalibriert werden. Die Kalibrierung erfolgt in zwei Schritten. Im ersten Schritt werden die Meßwerte des Sensors kalibriert. In einem zweiten Schritt wird die Position und Orientierung des Sensorkoordinatensystems relativ zum Werkzeugkoordinatensystem des Roboters kalibriert. Die Ergebnisse der Kalibrierung dienen zur Bestimmung von Parametern für die Transformation von Sensormeßwerten aus dem Sensorkoordinatensystem in das Werkzeugkoordinatensystem.

Für die Kalibrierung der Sensormeßwerte wurde ein am Lehrstuhl verfügbares hochgenaues Werkzeugvoreinstellgerät eingesetzt. Der Sensor wird dabei auf dem verfahrbaren Tisch des Werkzeugvoreinstellgerätes derart fixiert, daß dessen Meßebene senkrecht zu einer mit dem Voreinstellgerät fest verbundenen Meßfläche und einer darauf befindlichen Kante orientiert ist. Der Tisch des Werkzeugvoreinstellgerätes wird nun derart verfahren, daß die Meßebene des Sensors durchfahren wird. Dabei werden die Sensormeßwerte und die jeweilige Position des Tisches aufgenommen. Die Meßergebnisse waren Grundlage für die Entwicklung eines Softwarealgorithmus, der die vom Sensorsystem gelieferten Positionsdaten korrigiert. Der resultierende Linearitätsfehler des Sensors beträgt nach der Kalibrierung ±10 µm.

Im zweiten Kalibrierschritt wird die translatorische und rotatorische Verschiebung des Werkzeugkoordinatensystems in das Sensorkoordinatensystem bestimmt. Dazu muß die Position und Orientierung des Ursprungs des Sensorkoordinatensystems (SCP) relativ zum Werkzeugkoordinatensystem vermessen werden.

Aufgrund von Montagetoleranzen und auftretenden mechanischen Einflüssen während des Betriebes, können diese Daten nicht einfach aus den technischen Zeichnungen der Halterungen für Laser und Sensor entnommen werden. Vielmehr müssen diese in regelmäßigen zeitlichen Abständen vermessen werden. Dazu wurde ein

Kalibrierhilfsmittel entwickelt, mit dem die Kalibrierung zunächst manuell durchgeführt werden kann. Eine spätere Automatisierung ist möglich.

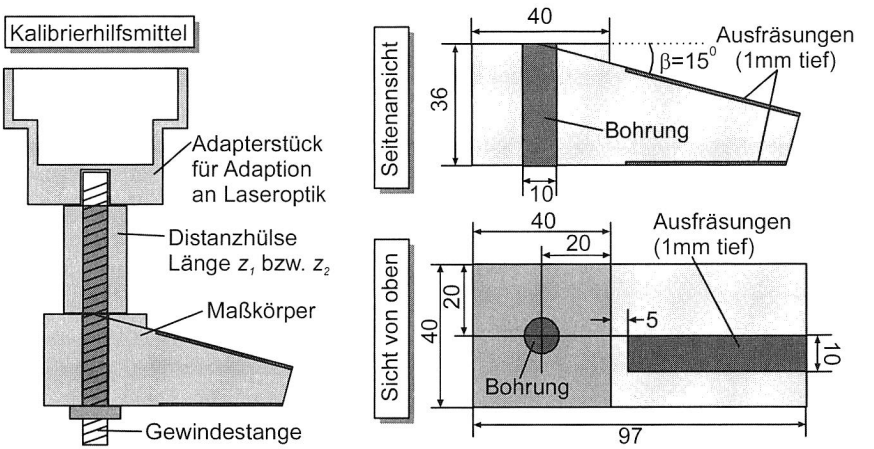


Bild 65: Kalibrierhilfsmittel (links) und Maßkörper (rechts) zur Kalibrierung der Beziehungen von SCP zu TCP

Das Kalibrierhilfsmittel besteht aus einem Adapterstück, zur mechanischen Adaption an den Laserkopf, auswechselbaren Distanzhülsen, zur Einstellung unterschiedlicher Meßabstände, und einem Maßkörper, der über zwei Meßebenen verfügt (Bild 65 links).

Die erste Meßebene des Maßkörpers ist senkrecht zur optischen Achse des Diodenlasers angebracht und enthält eine eingefräßte Kante, die der Sensor erfassen kann (Bild 65 rechts). Auf der gegenüberliegenden Seite des Maßkörpers wurde eine schiefe Ebene mit einem Winkel von fünfzehn Grad zur Horizontalen ausgefräst. Da der Neigungswinkel in die verwendeten Gleichungen eingeht, muß er exakt gefertigt werden. Bei der Herstellung des Maßkörpers wurde deshalb eine CNC-gesteuerte Werkzeugmaschine eingesetzt. Auch die geneigte Ebene muß über eine Kante verfügen, die der Lichtschnittsensor erfassen kann.

Mit der Anordnung nach Bild 66 (linke Abbildung) werden zwei Messungen bei unterschiedlichen Abständen z_1 und z_2 durchgeführt. Dazu werden geeignete Distanzhülsen zwischen Adapterstück und Kalibrierkörper eingebracht. Der Abstand z_1 entspricht dabei dem Fokusabstand der Laseroptik.

Mit dieser Anordnung wird die translatorische Verschiebung des SCP aus dem TCP in Richtung der z-Achse des TCP-Koordinatensystems bestimmt (Abstand z_{SCP}).

Weiterhin wird die Drehung des Sensorkoordinatensystems um die y-Achse des TCP-Koordinatensystems ermittelt (Kippwinkel κ).

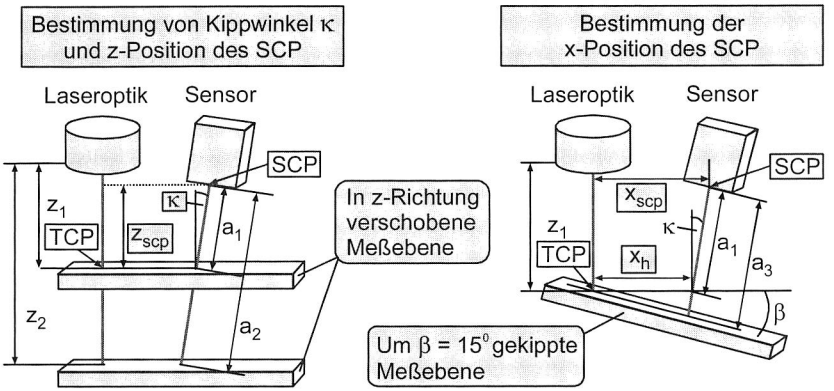


Bild 66: Prinzipskizze zur Kalibrierung des Lichtschnittsensors

Mit den Sensormeßwerten a_1 und a_2 wird der Kippwinkel κ nach Gl. 33 bestimmt.

$$\kappa = \operatorname{acos}\left(\frac{z_2 - z_1}{a_2 - a_1}\right) \qquad \text{Gl. 33}$$

Aus κ und a_1 wird der Abstand z_{SCP} zwischen TCP und SCP nach Gl. 34 berechnet.

$$z_{SCP} = a_1 \cdot \cos(\kappa) \qquad \text{Gl. 34}$$

Für die dritte Messung wird der Maßkörper so angebracht, daß eine Meßebene mit einem Winkel von fünfzehn Grad zur Horizontalen entsteht. Der Schnittpunkt der Meßebene mit der optischen Achse entspricht dem Fokuspunkt im Abstand z_1 (Bild 66 rechts).

Mit den Meßwerten a_1, a_3 und dem Winkel κ wird die Hilfsgröße x_h nach Gl. 35 berechnet.

$$x_h = (a_3 - a_1) \cdot \frac{\sin(90^o - \beta + \kappa)}{\sin(\beta)} \qquad \text{Gl. 35}$$

Aus Gl. 36 kann schließlich der Abstand x_{SCP} von TCP und SCP bestimmt werden.

$$x_{SCP} = x_h + a_1 \cdot \sin(\kappa) \qquad \text{Gl. 36}$$

Die Verschiebung y_{SCP} kann mit dem Kalibrierhilfsmittel nicht bestimmt werden, da dieses nicht am Laserkopf arretiert werden kann, sondern um dessen optische Achse drehbar ist.

Eine Möglichkeit zur Bestimmung von y_{SCP} besteht darin, bei aktivierter Nahtfolgeregelung, einen geradlinigen Überlappstoß abzufahren. Dabei wird die Laserenergie derart eingestellt, daß ein erkennbarer Einbrand auf dem Werkstück erzeugt wird, ohne dieses aufzuschmelzen. Der laterale Abstand zwischen Lasereinbrand und Kante entspricht $-y_{SCP}$. Zur exakten Vermessung ist dieses Vorgehen jedoch ungeeignet, da der Brennfleck eines Diodenlasers nicht punktförmig ist. Das Ziel dieses Vorgehens liegt vielmehr darin, den Parameter y_{SCP} zu variieren, bis der Brennfleck des Lasers eine optimale Position relativ zur Stoßkante einnimmt. Damit wurde y_{SCP} implizit bestimmt.

8.2.3 Transformation der Sensormeßwerte

Aufgrund der eingeschränkten Zugänglichkeit und aus technologischen Gründen wird der Sensorkopf in einem bestimmten Abstand vor dem Werkzeugbezugspunkt (TCP), in Bewegungsrichtung, an den Diodenlaser vorlaufend montiert. Für den Nahtfolgealgorithmus ist es erforderlich, daß die Positionsdaten der vermessenen Kante aus dem Sensorkoordinatensystem in das Werkzeugkoordinatensystem transformiert werden.

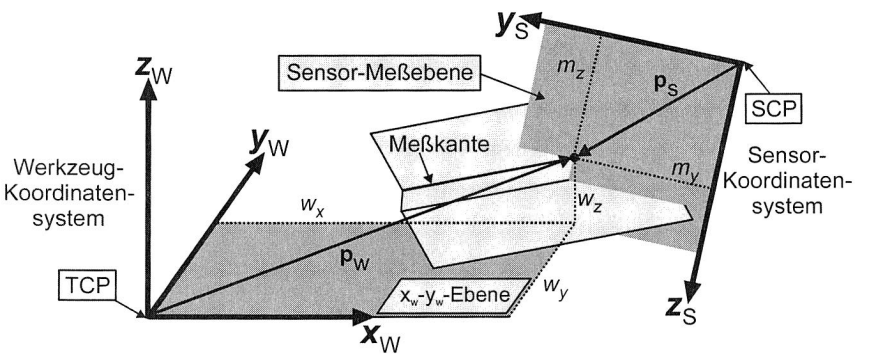


Bild 67: Geometrische Zusammenhänge zur Transformation von Sensormeßwerten

In Kapitel 6.3.1 wurde in allgemeiner Form diskutiert, daß die Transformation eines Positionsvektors $\mathbf{p}_S = (m_x, m_y, m_z)^T$ vom Sensorkoordinatensystem $\{\mathrm{S}\}$ in den entsprechenden Vektor $\mathbf{p}_W = (w_x, w_y, w_z)^T$ des Werkzeugkoordinatensystems $\{\mathrm{W}\}$ nach Gl. 11 erfolgt. Die Transformationsgleichung ${}^{W}_{S}\mathbf{T}$ wird dabei nach Gl. 10 gebildet.

Die freien Parameter der allgemeinen Tranformationsbeziehungen sollen im folgenden, am Beispiel des eingesetzten Sensorsystems, konkretisiert werden.

Die Parameter der translatorischen Verschiebung von $\{S\}$ aus $\{W\}$ werden mit sx, sy, sz bezeichnet. Übertragen auf das Sensorsystem entsprechen diese der translatorischen Verschiebung des SCP aus dem TCP. Diese Parameter werden bei der Kalibrierung des Sensorsystems bestimmt.
Für die gewählte Realisierung gilt: $sx = x_{SCP}$, $sy = 0$ und $sz = z_{SCP}$.

Die Rotationswinkel α, β, γ geben die Drehungen des Sensorkoordinatensystems $\{S\}$ aus dem Werkzeugkoordinatensystem $\{W\}$, um dessen Achsen $\boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W$, an. In der realisierten Versuchsanlage wird die Orientierung des Sensorkoordinatensystems zum Werkzeugkoordinatensystem durch die Parameter $\alpha = -\delta$, $\beta = 180^o + \kappa$ und $\gamma = 0$ beschrieben. Dabei ist der Winkel κ der bei der Kalibrierung bestimmte Kippwinkel des Sensors.

Der Winkel δ beschreibt ein zusätzliches Schwenken des Sensors um die $\boldsymbol{x}_W$-Achse des Werkzeugkoordinatensystems. Dieser Winkel wird bei der Kalibrierung vom Sensor direkt ausgegeben und muß nicht berechnet werden. Ein derartiges Schwenken des Sensors zum Laser ist beispielsweise beim Bearbeiten einer Kehlnaht erforderlich.

Nach Einsetzen der Parameter in die Transformationsgleichung nach Gl. 10 ergibt sich folgende Transformation der vom Sensor erfaßten Meßwerte m_x, m_y, m_z in die Werkzeugkoordinaten w_x, w_y, w_z:

$$\begin{bmatrix} w_x \\ w_y \\ w_z \\ 1 \end{bmatrix} = \begin{bmatrix} -\cos\kappa & \sin\kappa \cdot \sin\delta & -\cos\delta \cdot \sin\kappa & x_{SCP} \\ 0 & \cos\delta & \sin\delta & y_{SCP} \\ \sin\kappa & \cos\kappa \cdot \sin\delta & -\cos\kappa \cdot \cos\delta & z_{SCP} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ m_y \\ m_z \\ 1 \end{bmatrix} \qquad \text{Gl. 37}$$

Da es sich bei dem Lichtschnittsensor um einen zweidimensional messenden Sensor handelt, gilt: $m_x = 0$.

8.3 Sensorrechner zur Verarbeitung der Sensordaten

Marktgängige Robotersteuerungen verfügen derzeit nur über begrenzte Funktionen zur Verarbeitung von Sensordaten. Im wesentlichen beschränken sich diese auf einfache arithmetische Funktionen, die zur Verarbeitung der normalen Robotervariablen in der jeweiligen Roboterprogrammiersprache zur Verfügung stehen. Dieser Sachverhalt ist durch die geringe verfügbare Rechenleistung in proprietären Robotersteuerungen zu erklären.

Ein deutlicher Trend in der Robotertechnik geht in Richtung PC-basierter Robotersteuerungen. Diese bieten die Möglichkeit leistungsfähige Prozessoren aus dem PC-Massenmarkt einsetzen zu können. Allerdings wird die gewonnene Rechenleistung meist für eine grafische Bedienoberfläche wieder verbraucht. Die Ausführung des konfigurierbaren Softwaresystems zur Sensordatenintegration und -verarbeitung muß deshalb auf zusätzliche externe Prozessorleistung verlagert werden.

8.3.1 Auswahl der Hardwareplattform

Eine elegante und kompakte Lösung stellt die Integration eines Einplatinen-Rechners in das interne Bussystem der Robotersteuerung dar. In der industriellen Praxis haben sich zwei Bussysteme durchgesetzt. In PC-basierten Robotersteuerungen ist der ISA-Bus noch weit verbreitet. Durch den Marktdruck aus dem Officebereich wird dieser jedoch mittelfristig vom PCI-Bus substituiert werden.

Die zweite große Gruppe von Robotersteuerungen basiert auf dem VME-Bus. Obwohl es sich beim VME-Bus nicht um einen PC-Standard handelt, sind dennoch Einsteckkarten mit PC-Architektur am Markt verfügbar. Das jeweilige Bussystem versorgt die Einsteckkarte mit der benötigten Betriebsspannung. Eine externe Spannungsversorgung ist nur bei Verwendung besonders leistungsfähiger Prozessoren auf der Einsteckkarte notwendig.

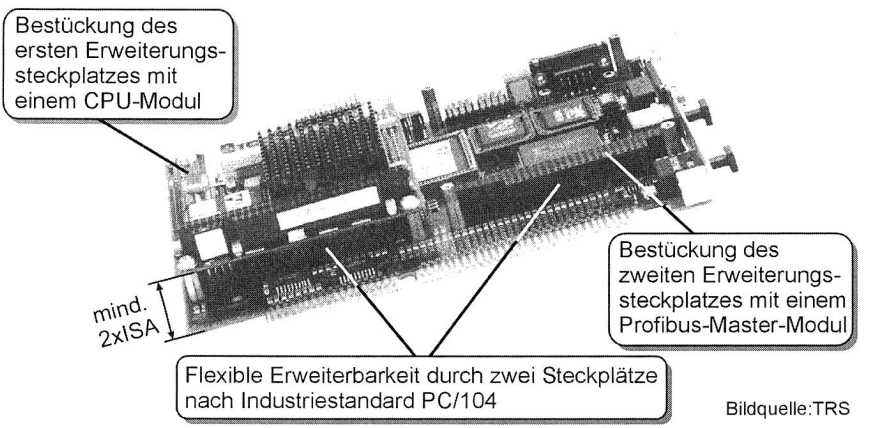


Bild 68: PC-Einsteckkarte für die Sensordatenintegration und -verarbeitung

Im folgenden wird diese Einsteckkarte als Sensorrechner bezeichnet. Bei der Auswahl eines geeigneten Sensorrechners mußten folgende Aspekte berücksichtigt werden:

- Belegung möglichst weniger der vier vorhandenen ISA-Slotplätze auf dem Motherboard der verwendeten PC-basierten Robotersteuerung

- Skalierbarkeit der Rechenleistung
- Flexible Schnittstelle zur Integration von Prozeßsensorik
- Einsatz eines multitaskingfähigen Echtzeitbetriebssystems.

Mit der Steckkarte SPC-300 der Firma TR Systemtechnik, Trossingen, konnten diese Anforderungen erfüllt werden (Bild 68). Es handelt sich bei dem Sensorrechner um eine Steckkarte mit zwei PC/104-Steckplätzen. PC/104 ist kompatibel zu ISA und stellt den Industriestandard im Embedded-PC-Bereich dar [57]. In dem ersten PC/104-Steckplatz steckt ein CPU-Modul mit einem 5x86-Prozessor, 4MB DRAM, einer 4MB Flashdisk und sämtlichen PC-relevanten Schnittstellen (inkl. Ethernet).

Für die Integration von Sensorsignalen verfügt das CPU-Modul nur über eine serielle Schnittstelle. Dies ist bei weitem nicht ausreichend, da Sensoren über eine Vielzahl unterschiedlicher Hardwareschnittstellen verfügen können. Häufig werden die Sensordaten als analoge Spannung ausgegeben. Zur Integration dieser Sensordaten ist ein Analog/Digital-Umsetzermodul erforderlich. Hinzu kommen Module mit digitalen Eingängen zum Einlesen von Statusinformationen des Sensors. Sollen durch das Softwaresystem auch Aktoren gesteuert werden, sind weiterhin Module zur Ausgabe digitaler Steuersignale bzw. analoge Ausgabemodule erforderlich. Moderne Sensoren werden zunehmend mit einer Feldbusschnittstelle ausgestattet. Da die Standardisierungsbemühungen in diesem Bereich gescheitert sind, existieren weiterhin eine Vielzahl inkompatibler Feldbusse [84].

Eine universelle Schnittstelle wird es sicherlich nie geben. Große Hoffnungen werden zur Zeit in Industrial Ethernet [19] gesetzt. Die Entwicklungen stehen jedoch noch am Anfang. Etablierte Technik stellen die bereits erwähnten Feldbusse dar. Ein Feldbussystem stellt, in Form eines Feldbusmastermoduls, eine einheitliche Hardwareschnittstelle für den Sensorrechner bereit. Die Umsetzung der einzulesenden bzw. auszugebenden Signalarten erfolgt durch sogenannte Feldbusslaves, die mit dem Mastermodul busförmig verdrahtet sind. Die Datenein- und ausgabe wird damit dezentralisiert. Auf dem Sensorrechner wird nur ein freier PC/104-Steckplatz benötigt, um den Feldbusmaster aufzunehmen. Die Wahl des verwendeten Feldbussystems ist prinzipiell beliebig. Aufgrund der hohen Bandbreite von 12 MBaud wurde als Feldbussystem Profibus DP ausgewählt [29].

In der beschriebenen Ausbaustufe belegt der Sensorrechner lediglich zwei ISA-Slotplätze auf dem Motherboard der Robotersteuerung.

8.3.2 Kommunikation zwischen Sensorrechner und Roboter

Die Kommunikation zwischen dem Sensorrechner und der Robotersteuerung erfolgt über ein Dual-Ported-Memory, das von der Robotersteuerung aus über den ISA-Bus angesprochen wird. Damit erfolgt die Kommunikation mit maximaler Performance über

Speicherzugriffe. Die Software-Schnittstelle zur Robotersteuerung besteht aus einer speziellen Task (SDB-Task) in der Roboter-Systemsoftware, die die Einbindung eigener Funktionen erlaubt (Bild 69).

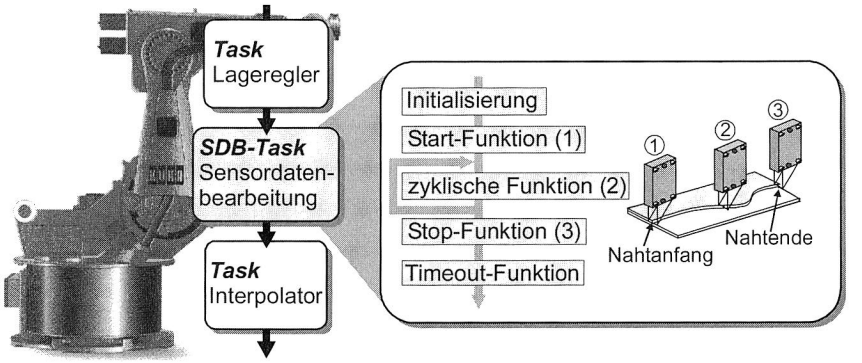


Bild 69: Roboterinterne Task zur Sensordatenbearbeitung (SDB-Task)

Neben einer Funktion zur Initialisierung können Funktionen eingebunden werden, die einmalig beim Starten und Stoppen der Kommunikationsverbindung mit dem Sensorrechner abgearbeitet werden sollen. Außerdem kann eine Funktion zur Timeoutbehandlung bei Zykluszeitüberschreitung und eine zyklische Funktion, die im Interpolatortakt abgearbeitet wird, integriert werden. Über die zyklische Funktion können aktuelle Parameter des Roboters, wie z.B. die momentane kartesische Position und Orientierung des TCP ausgelesen werden. In der Gegenrichtung kann die aktuelle Roboterbahn beeinflußt werden, indem entweder auf die Stützstellen einer programmierten Bahn Korrekturframes addiert werden, oder indem von der Bahnplanung des Sensorrechners neue Stützwerte vorgegeben werden.

Der Datenaustausch über die SDB-Task ermöglicht dem Benutzer einen hohen Grad an Flexibilität. Sie bietet die Möglichkeit die Kommunikation mit dem Dual-Ported-Memory des Sensorrechners individuell zu gestalten. Auf der anderen Seite werden von der SDB-Task alle relevanten Systemvariablen der Robotersteuerung zur Verfügung gestellt.

8.4 Anwendungsbeispiel Laserstrahlhartlöten

Die Entwicklung des Nahtfolgealgorithmus wurde mit Unterstützung eines Kinematiksimulations-Tools durchgeführt. Dadurch konnte sichergestellt werden, daß ein getesteter und optimierter Algorithmus für die weiteren Untersuchungen in der realen Roboterzelle zur Verfügung steht. Die Verifikation der Funktionsfähigkeit wurde

anschließend unter "Laborbedingungen", an CNC-gefrästen Testkonturen, durchgeführt. Einflüsse auf das Regelverhalten, aufgrund von Abweichungen zwischen den idealen Modellen aus der Kinematiksimulation und den kinematischen und dynamischen Eigenschaften der realen Systemkomponenten, konnten damit untersucht werden.

Die Industrietauglichkeit des Nahtfolgesystems kann jedoch aus den Laboruntersuchungen nicht ohne weiteres abgeleitet werden. Der industrielle Einsatz ist gekennzeichnet durch Einflußfaktoren, die im Laborbetrieb nicht auftreten. Zur Verifikation der bisherigen Ergebnisse wurde deshalb eine Anwendung ausgewählt, mit der die Industrietauglichkeit des Nahtfolgesystems nachgewiesen werden konnte.

8.4.1 Beschreibung der Aufgabenstellung

Als Testanwendung wurde das Laserstrahlhartlöten am Bördelfalz einer Karosserietür ausgewählt [72]. Aus Gründen des Korrosionsschutzes muß der Bördelfalz gegen das Eindringen von Feuchtigkeit abgedichtet werden. Bisher wurde auf dem Türrahmen ein Klebstoff aufgebracht, der das umgefalzte Außenblech mit dem Türrahmen verklebte. Zusätzlich wurde, in einem manuellen Arbeitsschritt, ein Dichtmittel auf den Bördelfalz aufgetragen. Durch Alterung des Klebers und des Dichtmittels konnte die Korrosionsbeständigkeit über die Lebensdauer der Tür nicht sichergestellt werden. Wird Ober- und Unterblech am Bördelfalz allerdings durch Hartlöten verbunden, so erreicht man durch die metallische Verbindung einen optimalen Korrosionsschutz und zusätzlich eine Steigerung der mechanischen Festigkeit [52].

Die Aufgabenstellung bestand darin, die aufgebaute Versuchsanlage im allgemeinen und die Nahtfolgeregelung im besonderen für das Laserstrahlhartlöten zu qualifizieren.

8.4.2 Ergebnisse zum roboterbasierten Laserstrahlhartlöten

Ein Diodenlaser ist für das Laserstrahlhartlöten besonders gut geeignet. Der längliche Querschnitt des Brennflecks kann so eingerichtet werden, daß er einerseits das Werkstück bereits vorwärmt bevor das Lot zugeführt wird. Andererseits hält er das geschmolzene Lot längere Zeit flüssig bevor dieses wieder erstarrt. Die kurzzeitige Aufrechterhaltung des flüssigen Aggregatzustandes ist erforderlich, damit das Lot zuverlässig das Ober- und Unterblech am Bördelfalz benetzen kann. Die exakte Positionierung von Brennfleck und Lotdraht zur Fügekante ist deshalb für die Bearbeitungsqualität von entscheidender Bedeutung.

Aus Gründen der Flexibilität und der Wirtschaftlichkeit wird in der Versuchsanlage ein Standard-Industrieroboter eingesetzt. An der letzten Achse des Roboters wird die Aufhängung für den Diodenlaser angeflanscht. Grundsätzlich sind dazu zwei unterschiedliche Konzepte denkbar. Zum einen wird der Laserkopf derart an den Roboterflansch montiert, daß die Laserstrahlausbreitungsrichtung im rechten Winkel zur

Drehachse der sechsten Achse des Roboters steht (Bild 70 links). Eine weitere Möglichkeit besteht darin, den Laserkopf so zu adaptieren, daß die Laserstrahlausbreitungsrichtung mit der Roboterhandachse fluchtet (Bild 70 rechts).

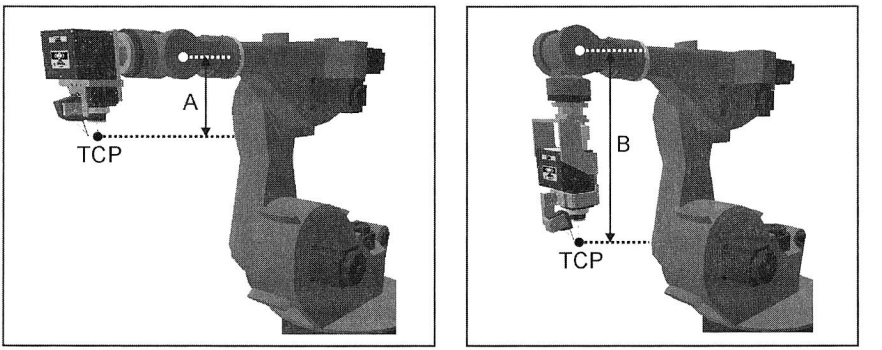


Bild 70: Unterschiedliche Möglichkeiten zur Adaption des Laserkopfes an die sechste Achse des Industrieroboters

Die erstgenannte Möglichkeit des Systemaufbaus hat den Vorteil, daß der Hebel (A) zwischen dem Werkzeugbezugspunkt (TCP) und der Handgelenkachse des Roboters (Achse 5) sehr klein gewählt werden kann. Der gesamte Systemaufbau gewinnt damit an Steifigkeit. Ein Mitführen des nicht symmetrischen Brennflecks des Diodenlasers erfordert allerdings große Ausgleichsbewegungen des Roboters sowohl in den Handachsen als auch in den Grundachsen. Dies kann aufgrund der damit verbundenen Reversierbewegungen in den Handachsen bzw. der geringeren Dynamik der Grundachsen zu Bahnungenauigkeiten führen. Außerdem sind die möglichen Verfahrrichtungen und damit auch der erreichbare Arbeitsraum stark eingeschränkt. So sind beispielsweise nur Bewegungen mit zunehmendem Abstand zwischen TCP und Robotersockel bei begrenzten Drehwinkeln möglich. Da mit dieser Anordnung die Karosserietür nicht vollständig bearbeitet werden konnte, wurde diese Art der mechanischen Kopplung nicht realisiert.

Die zweite, in Bild 70 rechts gezeigte Art der mechanischen Kopplung zwischen Laser und Führungsmaschine, ermöglicht ein Mitführen des Brennflecks durch Drehung meist nur einer Handachse, ohne weitere Ausgleichsbewegungen in den Grundachsen. Allerdings führt ein derartiger Systemaufbau zu einem sehr großen Abstand (B) zwischen Werkzeugbezugspunkt und der fünften Achse. Daraus resultieren hohe Drehmomentbelastungen auf den Roboterflansch und allgemein ein weniger steifer Systemaufbau. Besonders nachteilig wirkt sich dies bei der Bestimmung des Werkzeugbezugspunktes (TCP) aus. Hierbei muß eine ortsfeste Markierung unter

verschiedenen Richtungen angefahren werden. Die unvermeidliche Durchbiegung des Systemaufbaus verhindert eine exakte Bestimmung der TCP-Koordinaten. Da die gemessenen Sensorwerte immer auf den TCP bezogen werden, führt dies zu einer fehlerhaften Bestimmung der Fügekantenposition. Bei Orientierungsänderungen des Laserkopfes ist der Fehlereinfluß besonders groß. Eine Rotation um einen ungenau vermessenen TCP, führt dazu, daß der Brennfleck von dem Fügestoß wegbewegt wird (Bild 71). Während der Bearbeitungsversuche an einer Karosserietür, wurde deutlich, daß diese Problematik die wesentliche Fehlereinflußgröße darstellt.

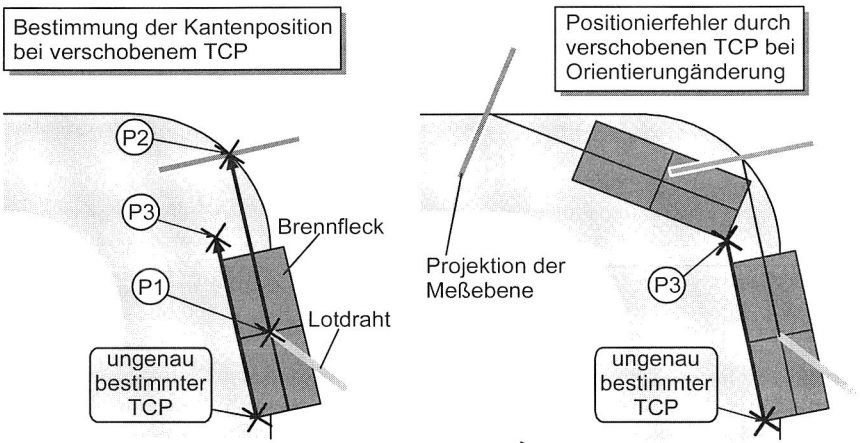


Bild 71: Positionierfehler durch ungenaue TCP-Bestimmung

Zu Beginn der Bearbeitungsversuche wurde untersucht, ob eine Bearbeitung mit einem durch ein Offline-Programmiersystem generiertem Roboterprogramm möglich ist. Genaue CAD-Daten der Karosserietür standen zur Verfügung. Positioniertoleranzen der Spannvorrichtung und Fertigungstoleranzen des Werkstücks machten eine Bearbeitung auf Basis eines Offline-Programms unmöglich. Kantenabweichungen bis zu 1,8 mm wurden ausgemessen. Daraus ergab sich die Schlußfolgerung, daß der Einsatz eines Sensorsystems unbedingt erforderlich ist.

Der Nahtfolgealgorithmus realisiert zwei Betriebsarten. In der ersten Betriebsart wird eine grob geteachte Bearbeitungsbahn bzw. ein Offline-Programm vorgegeben. In Abhängigkeit der Sensormeßwerte wird auf diese Bahn korrigierend eingewirkt. Es

können sowohl die Position der Bahnpunkte als auch die Orientierungswerte beeinflußt werden. Sinnvoll ist diese Art der Sensorregelung, wenn aufgrund von Zugänglichkeitsproblemen Umorientierungen des Bearbeitungskopfes während der Bearbeitung notwendig sind. Da der Sensor Kollisionen des Bearbeitungskopfes mit Spannvorrichtungen oder Werkstückaufbauten nicht erfassen kann, müssen die Orientierungsänderungen im Vorfeld programmiert werden. Dies kann elegant durch ein Offline-Programmiersystem erfolgen. Es findet in diesem Fall nur eine Positionsregelung und keine Orientierungsregelung durch den Nahtfolgealgorithmus statt.

Besteht keine Kollisionsgefahr, so kann der Nahtfolgealgorithmus die umfassende Steuerung des Roboters in allen sechs Freiheitsgraden übernehmen. Dies charakterisiert die zweite Betriebsart. In diesem Fall reduziert sich der Programmieraufwand auf Seiten des Roboterprogramms auf die Programmierung des Nahtanfangspunktes. Die restliche Bearbeitungsbahn wird online aus der Verarbeitung der Sensormeßwerte bestimmt.

Während der Bearbeitungsversuche wurde deutlich, daß der Meßbereich des verwendeten Sensors für eine Nahtfolgeanwendung nicht zufriedenstellend ist. Mit der verfügbaren CMOS-Technologie ist eine Meßgenauigkeit von ca. 10 µm nur zu erreichen, wenn der Meßbereich auf ca. ±3 mm begrenzt wird. Dies bedeutet eine relevante Einschränkung der erfaßbaren Krümmungsradien der Fügekante und der erreichbaren Verfahrgeschwindigkeit. Es zeigte sich, daß ein erweiterter Meßbereich wichtiger ist als eine hohe Meßgenauigkeit, da die Positioniergenauigkeit des Roboters wesentlich schlechter als die Meßgenauigkeit des Sensors ist.

8.4.3 Abschließende Bemerkungen

Mit der Realisierung einer roboterbasierten und sensorgestützten Versuchsanlage zum Laserstrahlhartlöten von Karosserietüren konnte die Funktionsfähigkeit des Gesamtsystems, unter industriellen Einsatzbedingungen, nachgewiesen werden. Die erzielte Bearbeitungsqualität entsprach den Anforderungen der Industriepartner. Optimierungsbedarf besteht noch im Hinblick auf eine höhere Bearbeitungsgeschwindigkeit. Diese wird in der realisierten Versuchsanlage zum einen durch die geringe Leistung des Diodenlasers von 1,5 kW und zum anderen durch den eingeschränkten Meßbereich des Nahtfolgesensors begrenzt. Moderne Hochleistungsdiodenlaser erreichen heute bereits Leistungen bis zu 3 kW, bei vergleichbarer Baugröße [58]. Setzt man neben einem leistungsfähigeren Diodenlaser auch einen alternativen Lichtschnittsensor mit größerem lateralem Meßbereich ein, so sind höhere Bahngeschwindigkeiten realisierbar. In Kapitel 7.3.3 wurde die Abhängigkeit der maximalen Bahngeschwindigkeit von der Größe des Meßbereichs des verwendeten Sensorsystems bereits diskutiert.

9 Informationstechnische Integration einer roboterbasierten Bearbeitungszelle

Die Automatisierung von Produktionsabläufen hat heute bereits ein sehr hohes technisches Niveau erreicht. Auf der Suche nach weiteren Einspar- und Rationalisierungspotentialen rückt die Organisation der Produktion zunehmend in den Mittelpunkt des Interesses. Zukünftig hängt der Erfolg eines Unternehmens nicht mehr so sehr von den traditionellen Produktionsfaktoren ab, sondern von der Fähigkeit den Informationsfluß im Unternehmen zu organisieren und damit anfallende Daten unternehmensweit auszuwerten. Wissen wird dabei zur kostbarsten Ressource.

Der Informationstechnologie kommt damit eine entscheidende Rolle bei der Wettbewerbsfähigkeit von Industrieunternehmen zu. Ihre zentrale Aufgabe ist es, die verschiedenen Stufen der Produktion intelligent zu verknüpfen und mit innovativen Lösungen die Produktionsprozesse zu optimieren. Von ebenso großer Bedeutung ist die informationstechnische Verfügbarkeit von Produktionsdaten für betriebswirtschaftliche Entscheidungsprozesse im Unternehmen (Bild 72). Verallgemeinert lassen sich diese Aufgaben auf eine Aussage fokussieren: es ist notwendig die richtige Information zur richtigen Zeit im richtigen Format am richtigen Ort bereitzustellen.

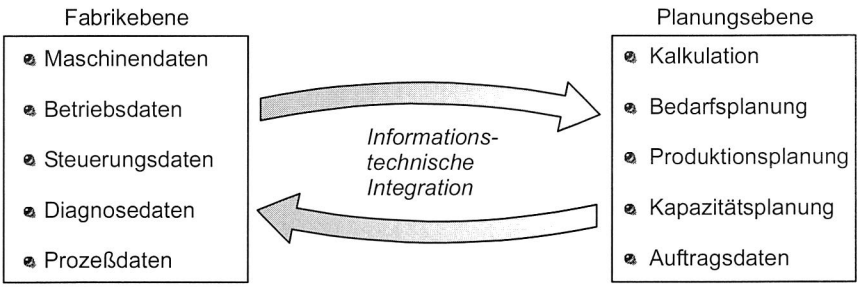


Bild 72: Informationstechnische Integration von Fabrik- und Planungsebene

Mitte der achtziger Jahre wurde unter dem Schlagwort CIM (Computer Integrated Manufacturing) erstmals in großem Maßstab versucht durch den Einsatz von Rechnersystemen eine durchgängige Informationsverarbeitung zu erreichen [3,42]. Zum damaligen Zeitpunkt war die informationstechnische Infrastruktur in den Unternehmen jedoch noch nicht in dem erforderlichen Maße ausgereift. Ein wesentlicher Faktor für das Scheitern der damaligen Bemühungen war das Fehlen standardisierter Schnittstellen und Methoden zum Austausch der relevanten Informationen zwischen der Steuerungsebene, der Zellenebene und der Leitebene. Heute erfährt der CIM-

Gedanke eine Renaissance. Mit der Web-Technologie und der OPC-Technologie stehen zwei leistungsfähige Basisinnovationen zur Verfügung, mit denen eine durchgängige Informationsverarbeitung vom Produktionsbereich bis ins Management in greifbare Nähe rückt.

9.1 Alternative Strategien zur Informationsnutzung

In Bezug auf die Nutzung bzw. Verarbeitung von Informationen ergeben sich im wesentlichen zwei Anwendungsbereiche, die sich in ihren spezifischen Anforderungen unterscheiden. Dies ist zum einen die Informationsnutzung durch Mitarbeiter und zum anderen die Informationsverarbeitung durch Rechnersysteme (Bild 73).

Bei der ersten Gruppe liegt der Schwerpunkt auf der visuellen Aufbereitung und der örtlich ungebundenen Bereitstellung von Informationen. Hier bietet sich ein Internet-Browser als Benutzerschnittstelle an, der die Informationen von einem Web-Server bezieht und grafisch aufbereitet. Nur durch die Möglichkeit zum schnellen Zugriff auf alle benötigten Informationen kann ein Mitarbeiter seine Aufgaben optimal ausführen.

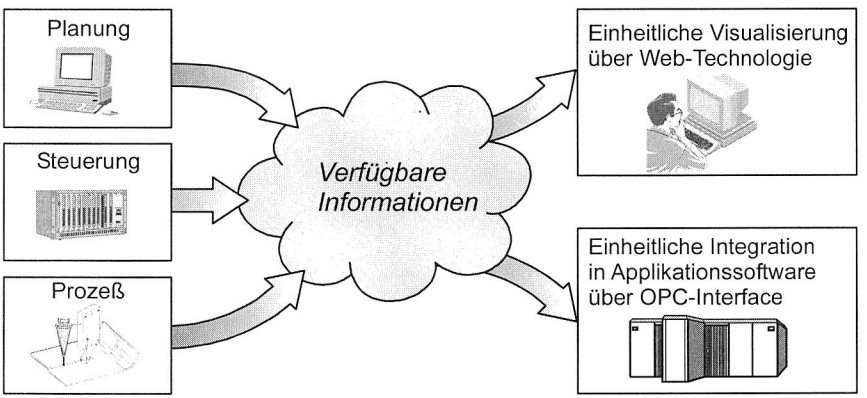


Bild 73: Visualisierung und Integration verfügbarer Informationen

Der zweite Anwendungsbereich umfaßt die Informationsverarbeitung durch Rechnersysteme. Dabei kann es sich sowohl um Rechnersysteme im Produktionsbereich als auch um Rechnersysteme im dispositiven Bereich handeln. Dies setzt einen durchgängigen Informationsfluß zwischen den Automatisierungsprozessen und den Geschäftsprozessen voraus. "OLE for Process Control (OPC)" hat sich mittlerweile als Standardschnittstelle in diesem Bereich etabliert [113]. Dabei stellen OPC-Server die relevanten Daten, in standardisierter Form, unternehmensweit zur Verfügung. Über eine OPC-Client-Funktionalität kann jede Anwendung auf diese Daten zugreifen.

Aufgrund der zunehmenden Bedeutung der informationstechnischen Integration von Produktionszellen in die IT-Landschaft eines Unternehmens, wurde auch für die in den vorausgegangenen Kapiteln vorgestellte roboterbasierte Bearbeitungszelle ein Integrationskonzept realisiert. Dazu wurde sowohl ein Web-Server als auch ein OPC-Server entwickelt. Damit steht ein abgerundetes Systemkonzept zur Verfügung, das es ermöglicht, die informationstechnische Isolation herkömmlicher Bearbeitungszellen zu überwinden.

9.2 Entwicklung eines Web-Servers

Ein Web-Server ermöglicht die Bereitstellung aller relevanten Produktions- und Prozeßdaten über das Intranet bzw. das Internet [39]. Dadurch kann sich der verantwortliche Mitarbeiter über einen Internet-Browser jederzeit beispielsweise über den Fehlerstatus der Bearbeitungszelle oder den aktuellen Produktionsstand informieren. Im Fehlerfall werden alle verfügbaren Fehlerinformationen visualisiert und damit eine Fehlerdiagnose erleichtert. Weiterhin ist es möglich, Hinweise zur Fehlerbehebung über vorbereitete statische HTML-Seiten bereitzustellen, evtl. mit integrierten Videosequenzen. Ebenso können im Wartungsfall auf ständig aktualisierbare Wartungshinweise bzw. Vorschriften zugegriffen werden, ohne langes Suchen der evtl. bereits veralteten Papierdokumente.

9.2.1 Bereitstellung statischer HTML-Dokumente

Neben den traditionellen TCP/IP-basierten Diensten, wie elektronische Nachrichten und Dateitransfer, gibt es eine Untermenge von TCP/IP-basierten Diensten im Internet, die das Hypertext Transfer Protokoll (HTTP) nutzen [11]. Dabei handelt es sich um ein spezielles Übertragungsprotokoll für Hypertext-basierte Dokumente. Diese Art von Dokumenten enthalten Verweise auf weitere Dokumente. Diese Untermenge des Internet bildet das World Wide Web (WWW).

Die Kommunikation zwischen einem Hypertext-anfordernden und einem Hypertext-liefernden Prozeß basiert auf dem Client/Server-Prinzip. Der Begriff Prozeß bedeutet hier ein im Zustand der Ausführung befindliches Programm [106]. Ein HTTP-Server auf Rechner A verwaltet eine Menge von Hypertext-Dokumenten im HTML-Format (Hypertext Markup Language) [93], die von einem WWW-Client auf Rechner B angefordert werden können. Die einfachste Aufgabe, die ein HTTP-Server zu erledigen hat, ist das Senden von statischen HTML-Dokumenten. Da es sich bei HTML-Dokumenten um Hypertext-Dokumente handelt, enthalten diese in der Regel Verweise auf weitere Dokumente. Diese Dokumente sind in einer hierarchischen Struktur organisiert. Das Dokument, das die höchste Ebene dieser Hierarchie bildet, nennt man die Startseite des HTTP-Servers. Die weiteren Hypertext-Dokumente

werden auf der Festplatte des HTTP-Servers in einer Dateistruktur abgelegt, die die hierarchische Organisationsstruktur abbildet.

9.2.2 CGI-Skripte und Serverprogramme

Zu den statischen, immer gleichen Informationen kommen dynamische Informationen hinzu, wenn der Zustand eines Gerätes mit einem Web-Browser abgefragt werden soll. Dynamische Informationen können z.B. Statusmeldungen, Maschinendaten, Prozeßdaten oder Betriebsdaten sein. HTML-Seiten mit dynamischen Informationen werden erst zum Zeitpunkt des Benutzerzugriffs erzeugt. Dazu muß von dem Server ein externes Programm gestartet werden, das auf die dynamischen Informationen des Gerätes zugreifen kann und diese in eine HTML-Seite einbindet. Bei komplexen Anwendungen kann das Programm auch eine Datenbank-Abfrage starten und das Ergebnis als HTML-Tabelle formatieren - hier sind viele Variationen denkbar (Bild 74).

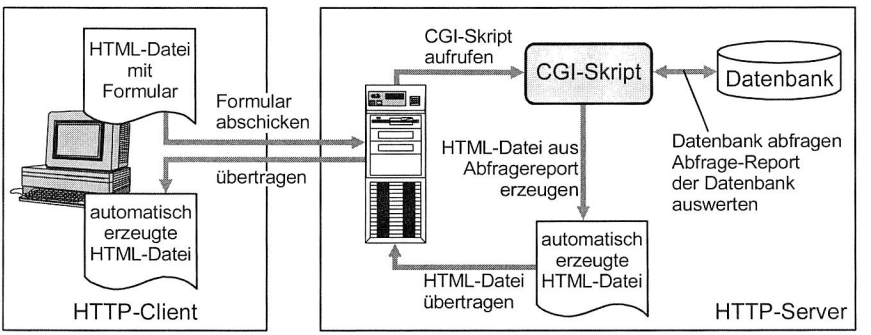


Bild 74: Erzeugen von dynamischen HTML-Seiten mit CGI-Skripttechnik

Programme, die derartige Server-Aktionen ausführen und dabei das "Common Gateway Interface" (CGI) nutzen, heißen CGI-Programme. CGI-Programme lassen sich in allen möglichen Programmiersprachen erstellen. Die Skriptsprache Perl hat sich jedoch geradezu als Standard für die CGI-Programmierung etabliert - daher ist häufig nur von CGI-Skripten die Rede [47].

In der Regel fordert der Benutzer durch Anklicken eines sog. "Aktualisierungs-Buttons" auf der HTML-Seite eines Gerätes die aktuellen Zustandsdaten an. Der Web-Browser sendet daraufhin an den HTTP-Server beispielsweise einen String der Form:

http://www.geraet.de/cgi-bin/script.cgi?anfrage=drehzahl

Dieser startet das CGI-Skript "script.cgi". Die Angaben nach dem Fragezeichen übergibt er als Argumente an das CGI-Programm. Nach Auswertung der Übergabewerte generiert das CGI-Skript eine neue HTML-Seite, in welche die angeforderten

dynamischen Informationen eingebunden sind. Diese HTML-Seite wird an den Standardausgabekanal gesendet, der auf den Eingabekanal des Web-Servers umgeleitet wurde. Über diesen Mechanismus wird die HTML-Seite an den HTTP-Server übergeben. Das CGI-Skript wird daraufhin beendet. Der HTTP-Server sendet schließlich die HTML-Seite an den HTTP-Client (Web-Browser), der damit die Bildschirmausgabe aktualisiert.

Ein weiterer HTML-Mechanismus, die Formularfunktion, gestattet das Ändern von Werten oder Zuständen in einem Gerät. Formulare zeigen im WWW-Client Standardelemente von grafischen Bedienoberflächen an: Check- und Listboxen, Auswahlmenüs, Buttons, Eingabefelder. Die Antwort des Benutzers auf solche Formulare wird der HTTP-Adresse des HTTP-Servers in etwas kryptischer Form angehängt, gut zu beobachten z.B. bei den Suchmaschinen im WWW. Durch derartige Formulare lassen sich z.B. Geräteeinstellungen auswählen oder Voreinstellungen setzen. In Abhängigkeit von den an den Web-Server übertragenen Formulardaten steuert ein CGI-Programm das entsprechende Gerät.

CGI-Programme haben jedoch einen entscheidenden Nachteil: Wenn der HTTP-Server ein CGI-Programm aufruft, so startet dieses in einem eigenen Prozeß. Im Falle eines Perl-Skripts wird also für jeden Aufruf des Skripts ein eigener Perl-Interpreter gestartet. Das Ausführen eines eigenen Prozesses benötigt jedoch relativ viel Rechenzeit und belastet die Ressourcen des Servers, da jedes CGI-Programm in einem eigenen Speicherbereich ausgeführt wird. Folglich sind Web-Anwendungen, die auf CGI basieren nur begrenzt skalierbar.

Als Alternative hierzu besteht die Möglichkeit die Web-Seite im Server verarbeiten zu lassen. Eine proprietäre Lösung stellen die Active-Server-Pages (ASP) dar, da sie nur im Microsoft Internet Information Server ablauffähig sind. Aufgrund der hohen Verbreitung dieses Web-Servers hat sie allerdings eine nicht unerhebliche Bedeutung erlangt. Für Web-Server anderer Hersteller gibt es ähnliche Konzepte, die meistens auf den Programmiersprachen Java oder Perl basieren (z.B. Perl-Interpreter bei Apache oder Java-Servlets bei verschiedenen Servern). Bei der Programmierung von Serverprogrammen ist man im Gegensatz zu CGI-Programmen stärker eingeschränkt auf die Umgebung des Web-Servers, da diese die Ausführung des Programms übernimmt. Der Vorteil dieser Methode liegt darin, daß die Bereitstellung einer dynamischen HTML-Seite schneller geht, da die Verarbeitung der Clientanfrage innerhalb des Web-Servers stattfindet. Damit entsteht kein Performanceverlust durch Schnittstellen und die Übergabe der Daten zwischen verschiedenen Programmen.

9.2.3 Interaktivität durch Java

Bei dynamischen HTML-Dokumenten findet ein Datenaustausch mit dem HTTP-Server nur dann statt, wenn der Benutzer eine Anfrage an den Server sendet. Der

Server wird nicht von selbst aktiv, wenn das Gerät z.B. seinen Betriebszustand ändert. Um zu kontrollieren, ob sich der Zustand geändert hat, müßte man die Seite erneut abrufen bzw. aktualisieren. Echte Interaktion wäre, wenn der Server von sich aus die Betriebszustandsanzeige im Web-Browser ändert. Werden solche Funktionen gefordert, stellt die Programmiersprache Java geeignete Möglichkeiten zur Realisierung bereit [44]. Wählt der Benutzer die Startseite eines Gerätes an, dann wird automatisch ein Java-Applet, das in das HTML-Dokument eingebettet ist, geladen und ausgeführt (Bild 75). Die Ausführung (Interpretierung) des Java-Bytecodes erfolgt in einer Java Virtual Machine (JVM), die in die gängigen Web-Browser bereits integriert ist. Das Applet nimmt über TCP/IP Verbindung mit dem HTTP-Server auf und lädt Java-Objekte. Die Java-Objekte stellen typische Bedien- und Darstellungselemente, wie Schalter, Schieberegler und Balkenanzeigen, auf dem Web-Client dar.

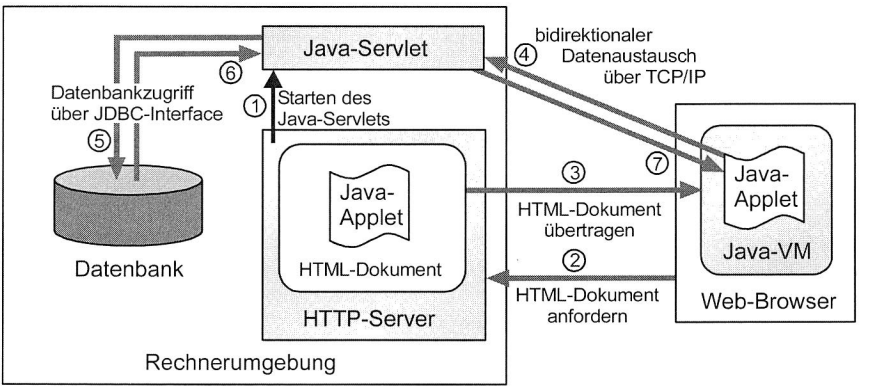


Bild 75: Bidirektionaler Datenaustausch mit Java-Technologie

Danach wird der Web-Server für den Datenaustausch nicht weiter benötigt. Dieses Applet bekommt dann die jeweils aktuellen Daten von einem der Einfachheit halber ebenfalls in Java geschriebenen Server-Programm. Voraussetzung ist, daß das Serverprogramm und der Web-Server auf dem gleichen PC laufen müssen. Dies liegt darin begründet, daß ein Java-Applet aus Sicherheitsgründen nur Kontakt zu dem Server aufnehmen kann, von dem es geladen wurde. Dies wird anhand der IP-Adresse des Servers überprüft. Die Aufgabe des Java-Applets besteht darin, die übertragenen Daten des Serverprogramms darzustellen. Auf diesen Weg ist es möglich, laufend aktuelle Daten im Browser des Benutzers zu visualisieren. In umgekehrter Richtung wird dem Benutzer die Möglichkeit zur Interaktion gegeben, indem Daten vom Applet zum Server zurückgesendet werden können (z.B. für Stop-Funktion).

9.2.4 Realisierung des Web-Konzeptes

Auf dem Sensorrechner sind alle relevanten Informationen verfügbar. Es liegt daher nahe den Web-Server ebenfalls auf dem Sensorrechner, als zusätzliche Task, ablaufen zu lassen. Dieser Ansatz steht ganz im Sinne einer kompakten Lösung auf einer einzigen Hardwareplattform und sollte deshalb favorisiert werden. Für das verwendete Echtzeitbetriebssystem VxWorks von Wind River Systems wird ein Embedded-Web-Server angeboten. Allerdings sind die Kosten bisher noch unverhältnismäßig hoch.

Da im Rahmen der realisierten Versuchszelle der Sensorrechner in eine PC-basierte Robotersteuerung integriert wurde, konnte auf das Angebot kostenfreier Web-Server für das Betriebssystem Microsoft Windows zurückgegriffen werden. Ausgewählt wurde der Apache Web-Server von der Apache Software Foundation (www.apache.org). Dieser Server verfügt über eine hohe Stabilität und gilt in Fachkreisen als recht sicherer Server.

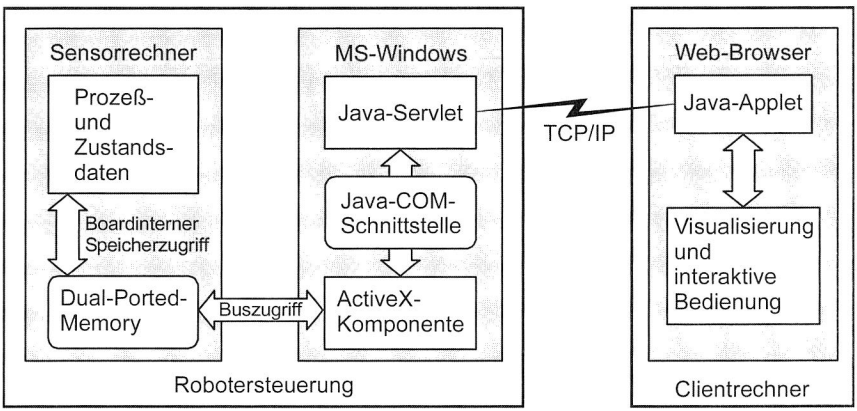


Bild 76: Interaktiver Zugriff auf Sensorrechnerdaten über Java

Durch den Einsatz eines Web-Servers in einer sensorgestützten Roboterzelle bietet sich die Möglichkeit Prozeß- und Zustandsdaten über Standard-Web-Browser visualisieren zu können. Dabei ist es jedoch wünschenswert, daß die dargestellten Daten, bei Veränderungen, automatisch aktualisiert werden. Um dies zu erreichen, wurden geeignete Java-Serverprogramme (Java-Servlets) entwickelt, die an die korrespondierenden Java-Applets im Web-Browser die aktualisierten Daten übertragen. Der Datenaustausch zwischen Serverprogramm und Applet erfolgt im wesentlichen über Java-Mechanismen, die dem Entwickler als Java-Klassen bereitgestellt werden. Die benötigten Rohdaten werden im Dual-Ported-Memory des Sensorrechners bereit

gestellt. Ein Java-Serverprogramm kann aus Sicherheitsgründen nicht auf die Hardware des Rechners zugreifen, auf dem es ausgeführt wird. Als Schnittstelle zwischen dem Dual-Ported-Memory des Sensorrechners und dem Java-Serverprogramm wurde deshalb eine hardwarenahe ActiveX-Komponente entwickelt, die den Dual-Ported-Memory ausliest (Bild 76). Über eine Java-COM-Schnittstelle können die Daten anschließend von dem Java-Serverprogramm eingelesen und weiterverarbeitet werden.

Für den Web-Server der robotergestützen Bearbeitungszelle wurden verschiedene Web-Seiten erstellt. Eine Web-Seite visualisiert relevante Zustands- und Prozeßdaten der Roboterzelle. Der Bediener kann sich dabei über den aktuellen Status des Roboters, des Lasers und des Sensors informieren. Fehlerzustände werden grafisch hervorgehoben, damit sie sofort auffallen und der Benutzer zeitnah reagieren kann. Außerdem kann er wesentliche Produktionsdaten, wie z.B. die bereits gefertigte Stückzahl eines Loses, oder auch Wartungsdaten, wie z.B. den Betriebsstundenzähler, abfragen. Zur Unterstützung der Fehlerdiagnose werden alle Systemmeldungen protokolliert und angezeigt (Bild 77).

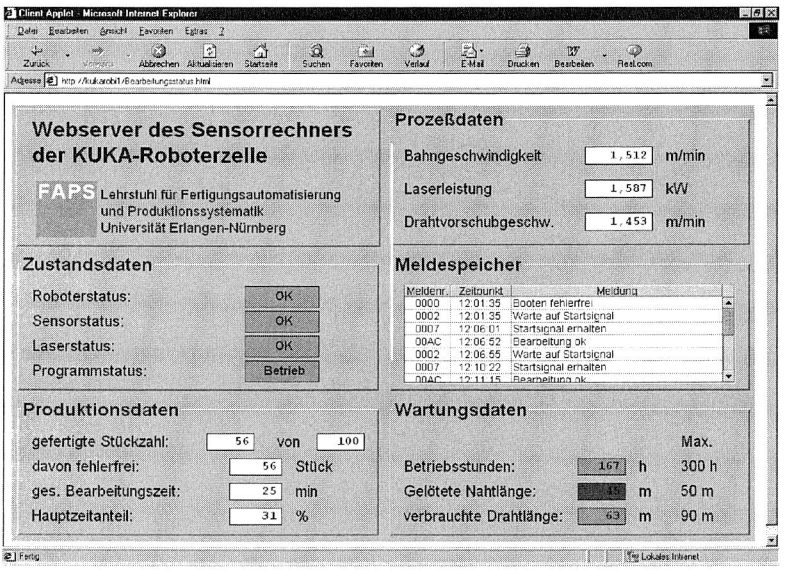


Bild 77: Visualisierung relevanter Zellendaten durch einen Web-Browser

Eine weitere Web-Seite stellt qualitätsrelevante Daten zum Bearbeitungsprozeß zur Verfügung (Bild 78). Vor der Bearbeitung können Toleranzgrenzen für die maximal

zulässige Abweichung von Soll- und Istposition des Bearbeitungswerkzeuges angegeben werden. Außerdem können Schwellwerte für die Sensormeßwerte festgelegt werden. Wird die maximale Positionsabweichung bzw. ein Sensorschwellwert überschritten, so wird die kartesische Position des Bearbeitungswerkzeugs bzw. des Meßortes von dem Java-Servlet protokolliert. Nach der Bearbeitung werden die protokollierten Daten durch das Java-Applet visualisiert. Der Benutzer kann jede protokollierte Position anwählen und die Prozeßdaten für diesen Arbeitspunkt untersuchen. Die visualisierten Daten liefern wichtige Informationen für die Einstellung der Prozeßparameter. Beispielsweise können Rückschlüsse auf die Einstellung der maximal möglichen Bahngeschwindigkeit gezogen werden. Dies dient der Optimierung von Durchlaufzeiten. Vor allem bei der Inbetriebnahme der Bearbeitungszelle, aber auch im laufenden Betrieb, stellt dies eine große Unterstützung für das Einrichte- und Bedienpersonal dar.

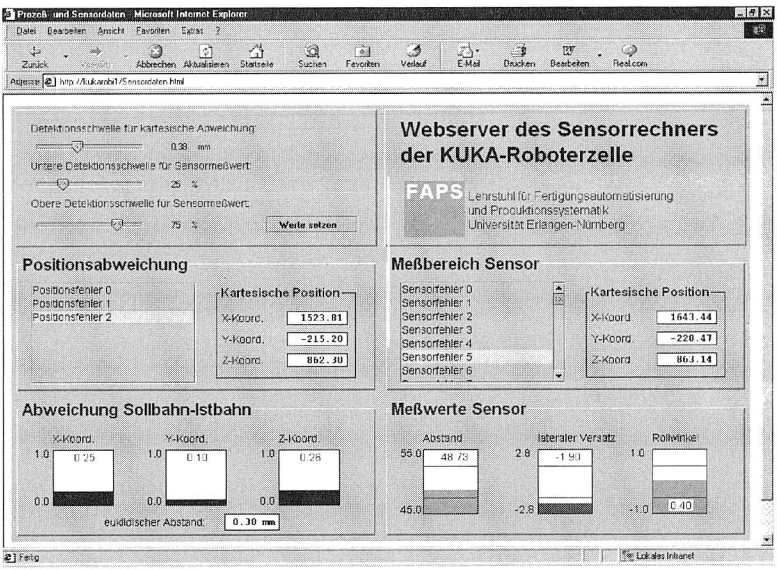


Bild 78: Visualisierung von Positionsabweichungen und Sensormeßwerten

Als zusätzliche visuelle Informationsquelle kann das Livebild einer Web-Cam in die Web-Seite integriert werden. Aus Performancegründen wird das Bild der Kamera von einem externen Kamera-PC über durch eine spezielle Einsteckkarte digitalisiert und komprimiert (Bild 79). Ein Softwareprogramm wandelt diese Daten anschließend in einen Video-Stream um und leitet diesen auf einen weiteren Web-Server auf dem Kamera-PC weiter. Damit kann der Livebild-Video-Stream, nach dem Herunterladen

eines entsprechenden Java-Applets, auf einem herkömmlichen Web-Browser dargestellt werden.

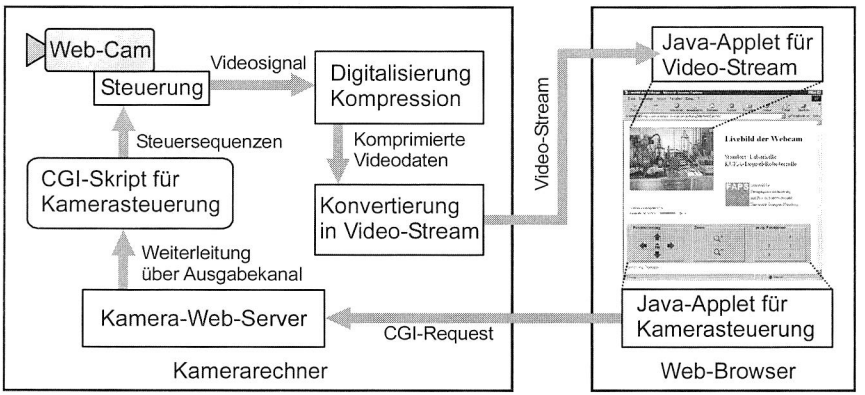


Bild 79: Integration von Video-Streams in HTML-Seiten

In einer HTML-Seite ist es möglich mehrere Java-Applets einzubetten. Jedes Applet kann allerdings nur von dem Web-Server Daten beziehen, von dem es geladen wurde. Es ist damit möglich den Video-Stream der Web-Cam in eine beliebige HTML-Seite der Roboterzelle einzubetten. Bei steuerbaren Kameras können Steuerbefehle vom Benutzer über das Java-Applet an den Web-Server zurückgeschickt werden, der diese Steuerbefehle dann an den Kamera-Rechner weiterleitet. Dadurch kann die Kamera in verschiedene Richtungen gedreht werden und der Zoom eingestellt werden. Zur Steuerung der Kamera wurde eine spezielle HTML-Seite entwickelt.

Das "Web" muß keineswegs immer "World Wide" sein. Abgesehen von Teleservice-Anwendungen spielen sich die webbasierten Anwendungen vor allem innerhalb des firmeneigenen Intranets ab. Im Extremfall könnte es sich auch einfach nur um eine serielle Verbindung handeln, mit der ein Servicetechniker mit seinem Notebook Einstellungen an einem Gerät vornimmt. Das TCP/IP-Protokoll und die Programmiersprache Java bieten standardisierte Mechanismen zum Bedienen und Beobachten, die tatsächlich offen und nicht auf die Windows-Welt beschränkt sind.

9.3 Durchgängiger Datenfluß durch OPC-Technologie

In einer Automatisierungslandschaft mit Automatisierungskomponenten unterschiedlicher Hersteller und den dabei verwendeten Bussystemen mit ihren verschiedenartigen Protokollen und Schnittstellen ist es besonders wichtig, ein standardisiertes Bindeglied zwischen Anwendungsprogrammen der Büro- und Automatisierungswelt

sowie zwischen den diversen Automatisierungskomponenten bereitzustellen. Mit OPC steht eine Technologie zur Verfügung, die diese Anforderungen erfüllt.

9.3.1 Technischer Hintergrund

OPC (OLE for Process Control) basiert auf der OLE-Technologie, die 1991 als Object Linking and Embedding in Zusammenhang mit Verbundprojekten startete und sich zur Kerntechnologie von Microsoft zur Unterstützung von Komponentensoftware entwikkelte [109]. OLE besteht aus einem ganzen Sortiment von Schnittstellen und Diensten. Es stellt einen Rahmen dar, innerhalb dessen eine Entwicklung von wiederverwendbaren, integrierten Softwarekomponenten möglich wird. OLE ist eine offene, erweiterbare Architektur, die auf dem Component Object Model (COM) basiert. Vereinzelt wird in der Fachwelt OLE mit COM gleichgesetzt. Richtig ist, daß COM die Basistechnologie darstellt, auf der OLE als spezielle Funktionalität aufsetzt. Auf der Basis von OLE/COM können Anbieter von Software unabhängig voneinander Komponenten entwickeln, die miteinander Daten austauschen können. DCOM erweitert die COM-Technologie um Zugriffsmechanismen bezüglich der Verteilung von Objekten über mehrere Rechner.

OLE stellt lediglich eine Technologie zur Kommunikation zwischen Anwendungen dar, vergleichbar mit der frühen Welt der Feldbusse. Erst die Standardisierung von Objekten und Methoden ermöglicht Interoperabilität, wie sie die Automatisierungstechnik beispielsweise von den Feldbus-Profilen [68] her kennt. Durch die Erweiterungen zu OLE in Form von Strukturdefinitionen, Interfaces und Techniken für effizienteren Datentransfer, erweitert OPC die Vorteile von OLE in Richtung eines OLE für Anwendungen der Automatisierungstechnik. OLE for Process Control (OPC) definiert Standardobjekte, Standardmethoden und Standardeigenschaften, die den Anforderungen für Interoperabilität von Programmen in der Automatisierungstechnik genügen [54].

Wichtigstes Ziel der OPC Aktivitäten ist es deshalb, eine offene Schnittstelle zu entwickeln, die einen einfachen, standardisierten Datenaustausch zwischen dem Bürobereich und PC-Anwendungen in der Fertigung ermöglicht. Diese Schnittstelle soll für die Nutzer einfach zu handhaben sowie für die Anbieter von Automatisierungssystemen einfach zu implementieren sein. Im September 1996 wurde die OPC Foundation in den USA gegründet, die alle Aktivitäten koordiniert. Auf der ISA Show in Chicago des selben Jahres wurden erstmals Prototypen in einem gemeinsamen Verbund gezeigt. Heute besitzt die OPC Foundation bereits ca. 220 Mitglieder, darunter alle bedeutenden, weltweit aktiven Hersteller von Automatisierungssystemen wie Fisher-Rosemount, Siemens, Rockwell Automation, ABB, National Instruments und Honeywell. Die Mitgliedsfirmen in der OPC Foundation kommen zu 43% aus den USA, zu 30% aus Europa und zu 23% aus Asien [87]. Der bemerkenswert hohe Anteil

von über 20% Endkunden in der OPC Foundation [87] zeigt, daß vor allem Endkunden von der OPC-Technologie signifikante Vorteile erwarten. Auch Microsoft dokumentiert mit der Mitgliedschaft die volle Unterstützung für diesen Standard. In keiner anderen Organisation stehen solche gebündelten Ressourcen an Entwicklungskapazitäten, zusammen mit einem globalen Marketing, zur Verfügung.

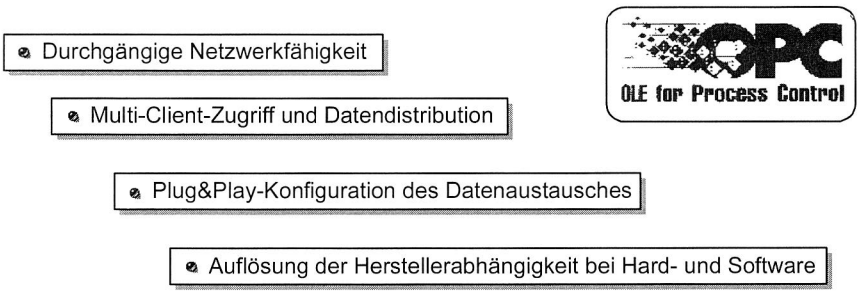


Bild 80:Vorteile der OPC-Technologie

Neben dem Ziel einer generellen Vereinfachung der Verbindung von Anwendungen der Fertigungs- und Verfahrenstechnik mit Business/Office-Anwendungen bietet OPC eine Reihe von weiteren Vorteilen [67]:

- Auflösung der Herstellerabhängigkeit bei Hard- und Software

Der Stand der Technik, vor OPC, war dadurch gekennzeichnet, daß zum Anschluß von Prozeßhardware ein spezieller Treiber den Datenaustausch zu einer PC-Anwendung ermöglicht. OPC bietet ein Standardinterface, unabhängig von der konkreten Anwendung. Für den Hersteller von Hardwarekomponenten bedeutet das, dieser muß nur noch ein OPC-Server-Interface implementieren. Die Hersteller von Softwareprodukten wie Visualisierungen, Meßsysteme, etc. müssen nur noch über eine OPC-Client-Schnittstelle verfügen. Softwareentwickler brauchen keine Treiber anzupassen, falls sich einmal infolge eines neuen Ausgabestandes der Automatisierungskomponenten die Kommunikationsfunktionen ändern. Der Endkunde wiederum kann frei zwischen den verschiedenen Anbietern von Hard- und Softwarekomponenten wählen und sich auf funktionale Auswahlkriterien konzentrieren und nicht darauf achten, daß ein entsprechender Treiber verfügbar ist.

- Plug&Play-Konfiguration des Datenaustausches

Mit OPC wird die Konfiguration des Datenaustausches zur Hardware oder zwischen Anwendungen stark vereinfacht. Existiert ein OPC-Server, wird dieser ausgewählt und anschließend die Daten (Items) des OPC-Servers, in gewohntem Microsoft Explorer

Style mit Mausklick ausgewählt. Dabei können die Items zu Gruppen strukturiert werden, so daß nur die Daten ausgetauscht werden, die wirklich von Interesse sind.

- Multi-Client-Zugriff und Datendistribution

Jeder OPC-Server ist in der Lage Anfragen von mehreren Clients zu bearbeiten. Sollen Daten mehrfach genutzt werden, zum Beispiel von einer Visualisierung, als auch von einer Datenbank, können verschiedene Softwarepakete auf die Daten des OPC-Servers zugreifen. Dafür bedarf es keiner herstellerspezifischen Vereinbarung oder zusätzlichen Implementierung. Der OPC-Server nutzt dafür die Microsoft Funktionen des Betriebssystems (COM-Technologie).

- Durchgängige Netzwerkfähigkeit

OPC nutzt neben der COM-Technologie auch DCOM (Distributed COM), um Netzwerkfähigkeit zu erreichen. Damit stehen nicht nur Datenquellen (OPC-Server), die auf dem lokalen PC verfügbar sind, sondern alle Server des Netzwerkes zur Verfügung. Der OPC-Client bemerkt nicht, ob es sich um eine lokale oder eine verteilte Datenquelle handelt. OPC bzw. DCOM verbirgt den Unterschied.

9.3.2 Funktionalität eines OPC-Servers

OPC ist objektorientiert und besitzt eine Client/Server-Struktur [67]. Programme, die Daten aus der Automatisierungsebene benötigen, fordern diese über Dienste an, sie sind also Clients. Die Automatisierungskomponenten ihrerseits stellen mittels OLE die benötigten Informationen bereit, sie sind also Server.

Die Aufgabe von OPC besteht darin, die OLE-Objekte und Methoden für Anwendungen in der Automatisierungstechnik zu standardisieren. Die OPC-Serverkomponente bietet eine Schnittstelle zu den OPC-Objekten und verwaltet auch diese Objekte. Ein OPC-Server besteht aus drei hierarchisch abgestuften Objekten (Bild 81):

- dem Serverobjekt
- dem Group-Objekt und
- dem Item-Objekt.

Das Server-Objekt liefert Informationen über sich selbst und dient als Container für Group-Objekte. Ein Group-Objekt enthält wiederum Informationen über sich selbst und dient der Verwaltung und Organisation der Item-Objekte. Durch Gruppen wird es dem Client ermöglicht, die Daten und Prozeßgrößen in verschiedenen Einheiten zusammenzufassen. Die Daten innerhalb dieser Einheiten können gelesen oder geschrieben werden. Zwischen einem Client und den Items in einer Gruppe kann eine ereignisgesteuerte Verbindung aufgebaut werden, die je nach den Bedürfnissen aktiviert oder deaktiviert werden kann. Der Client kann für jede Gruppe eine Auffrischungsrate festlegen, die bestimmt, in welchen Zeitabständen ein Client über Werteänderungen

der Items innerhalb einer Gruppe vom Server unterrichtet werden will. Es gibt zwei Arten von Group-Objekten, die öffentlichen ("public") und die privaten ("private" oder "local"). Öffentliche Gruppen können von mehreren Clients genutzt werden, während private Gruppen nur einem Client zugeordnet sind.

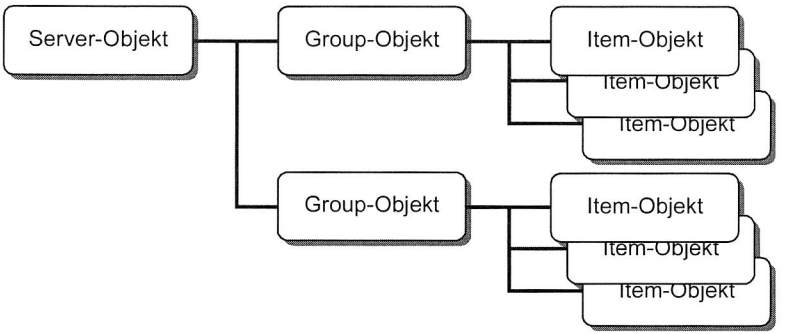


Bild 81: Hierarchische Beziehungen Server-Group-Item

Die Items repräsentieren Verbindungen zu den realen Daten, über die man durch den Server Zugang hat. Ein Item-Objekt ist nicht direkt als ein Objekt ansprechbar, da dafür keine Schnittstelle nach außen vorgesehen ist. Alle Zugriffe auf ein Item müssen über dessen Group-Objekt erfolgen, das dieses Item enthält. Dabei können synchrone und asynchrone Zugriffe erfolgen. Bei synchronen Zugriffen wird die Funktion erst nach vollständiger Bearbeitung beendet. Asynchrone Aufrufe reihen dagegen die Aufträge nur in eine Queue des Servers ein und kehren dann sofort zum Client zurück. Mit jedem Item ist ein Wert, eine Qualität und ein Zeitstempel verbunden. Die Items sind aber nicht die eigentlichen Datenquellen, sondern es sind nur Verbindungen zu diesen.

Eine grundlegende Aufgabe des OPC-Servers ist die Erzeugung und Verwaltung eines Namensraumes. Der Namensraum beschreibt die über OPC erreichbaren Zustands- und Prozeßwerte.

9.3.3 Realisierter OPC-Server

Die OPC-Technologie ist hervorragend dazu geeignet, Prozeß- und Zustandsdaten des Sensorrechners fabrikweit anderen Verarbeitungsprogrammen zur Verfügung zu stellen. Die OPC-Technologie ist bisher jedoch nur auf der Windows-Plattform verfügbar, da sie auf den windowsspezifischen Kommunikationstechnologien OLE und COM basiert. Bei der verwendeten Robotersteuerung von KUKA handelt es sich um eine PC-basierte Steuerung. Das Betriebssystem Microsoft Windows 95 wird in der Steuerung als Mensch-Maschine-Schnittstelle genutzt. Dieses Systemkonzept ermöglicht es, einen OPC-Server auf der Robotersteuerung einzusetzen. Dies hat, im

Gegensatz zu denkbaren Lösungen bei nicht PC-basierten Robotersteuerungen, wesentliche Vorteile. Zum einen handelt es sich um eine integrierte Lösung, die keine zusätzlichen Hardwarekomponenten erfordert. Nicht PC-basierte Robotersteuerungen benötigen hingegen einen externen Rechner mit dem Betriebssystem MS Windows. Zum anderen kann der Datenaustausch zwischen Sensorrechner und OPC-Server über den Dual-Ported-Memory des Sensorrechners abgewickelt werden. Da es sich hierbei um reine Speicherzugriffe handelt, erfolgt der Datenaustausch nahezu verzögerungsfrei und ohne nennenswerten Aufwand von Rechenleistung.

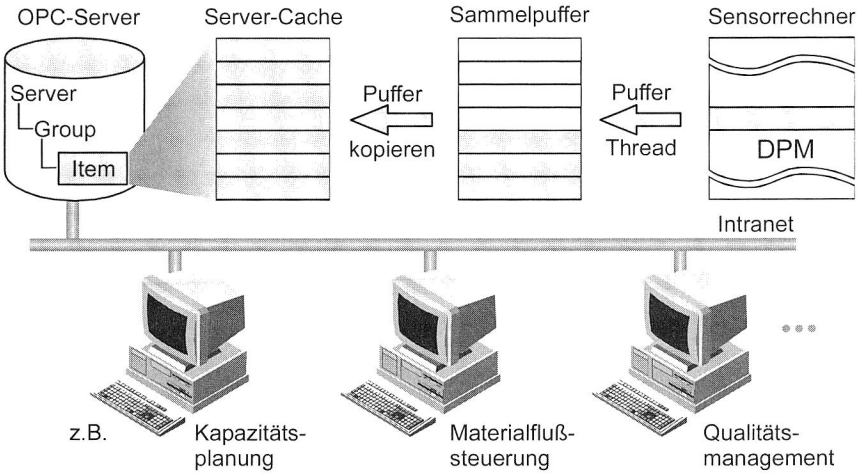


Bild 82: Zwischenpufferung zeitkritischer Prozeßwerte im OPC-Server

Für einige Prozeßgrößen, wie z.B. dem aktuellen Positionsframe, liefert die Steuerung im Interpolationstakt von 12 ms neue Daten. Diese werden an entsprechenden Speicherbereichen des Dual-Ported-Memory abgelegt. Auch wenn der OPC-Server die Daten im Interpolationstakt bereitstellt, ist es für einen OPC-Client, der sich auf einem anderen Rechner im Netzwerk befindet, unmöglich die Daten in so kurzer Zeit über das Netzwerk auszulesen. In diesem Fall gehen wichtige Prozeßwerte verloren. Bei dem realisierten OPC-Server wurde deshalb der Datentyp Array verwendet. Ein Array ist eine Tabelle von Elementen des selben Datentyps. Ein Array von Prozeßwerten wird im OPC-Server wie ein herkömmliches Item verwaltet. Zur Speicherung der Prozeßwerte eines Arrays wurde im OPC-Server ein sog. Sammelpuffer implementiert (Bild 82). In diesen werden die aus dem Dual-Ported-Memory ausgelesenen Prozeßwerte geschrieben. Ist die parametrierte Anzahl von Prozeßwerten erreicht, so werden diese in einen sog. Server-Cache kopiert. Das Datenarray in dem Server-Cache wird nun als eigentliche Prozeßgröße behandelt. Wenn nun ein Lesebefehl auf

ein Item mit dem Datentyp Array ausgeführt wird, so werden die Daten aus dem Server-Cache, anstatt direkt aus dem DPM, ausgelesen. Das Array-Item wird erst wieder aktualisiert, wenn der Sammelpuffer dieses Items wieder vollständig mit neuen Daten aufgefüllt wurde. Durch die Zwischenpufferung von Prozeßwerten ist die lückenlose Aufzeichnung relevanter Prozeßgrößen, durch einen OPC-Client, möglich. Die Entwicklung eines OPC-Servers, zur Bereitstellung von Prozeß- und Zustandsdaten des Sensorrechners, erfolgte mit Hilfe des OPC Server Toolkits [82] von Softing. Dieses Entwicklungssystem stellt ein Framework von C++ Klassen bereit, das die OPC Interfaces und Laufzeitfunktionalitäten implementiert. Damit kann die Entwicklungszeit zur Erstellung eines OPC-Servers wesentlich verkürzt werden.

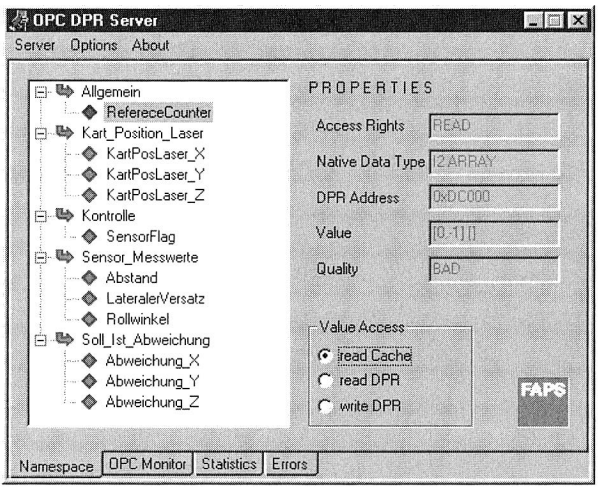


Bild 83: Darstellung des Namensraumbaumes im OPC-Server

Mit Hilfe des OPC Server Toolkits wurde ein OPC-Server für die KUKA-Robotersteuerung entwickelt. Dieser bietet die Möglichkeit einen Namensraum zu verwalten (Bild 83). Über eine grafische Benutzeroberfläche besteht die Möglichkeit im Namensraum die Verknüpfung zwischen OPC-Items und den Adressen der betreffenden Speicherstellen im DPM zu editieren. Ein OPC-Monitor gibt Auskunft über den Verbindungszustand des OPC-Servers. Durch Anwahl des entspechenden Fensters werden die mit dem Server verbundenen Clients und die von ihnen genutzten Items angezeigt. In einem weiteren Fenster werden statistische Informationen visualisiert. Die statistischen Informationen umfassen die Anzahl von OPC-Requests, Zeiten für Lesen und Schreiben von Prozeßwerten und Anzahl der aktuell verwalteten Objekte. Abschließend wird ein Fenster zur Darstellung von Fehlermeldungen angeboten.

9.3.4 Realisierter OPC-Client

Zur Verdeutlichung der Anwendungsbereiche von OPC wurde beispielhaft eine Visual-Basic-Anwendung mit einer OPC-Client-Schnittstelle implementiert. Damit ist es möglich auf Daten aus der Roboterzelle zuzugreifen. Durch die Implementierung von Sammelpuffern im OPC-Server ist gewährleistet, daß alle Daten lückenlos zum OPC-Client übertragen werden. Aufgrund der Verfügbarkeit aller Daten kann eine umfangreiche Verarbeitung und Visualisierung dieser Daten durch die Visual-Basic-Anwendung erfolgen. Bei Bedarf können die Rohdaten bzw. die Verarbeitungsergebnisse zur Archivierung auf Datenträgern gespeichert oder in eine Datenbank geschrieben werden.

Analog zu den realisierten HTML-Seiten des Web-Servers (Kapitel 9.2.4) ermöglicht auch die Visual-Basic-Anwendung die Visualisierung von Prozeß- und Zustandsdaten der robotergestützten Bearbeitungszelle. Es gibt jedoch wesentliche Unterschiede.

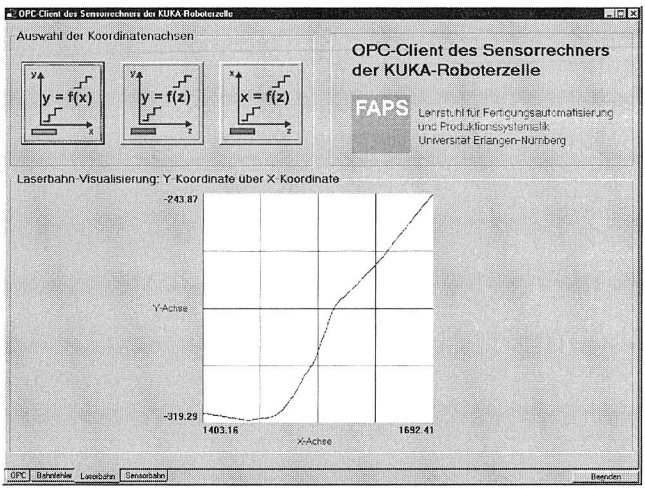


Bild 84: Visualisierung des Bewegungspfades durch den OPC-Client

Das Java-Servlet kann, aus Sicherheitsgründen, die vom Sensorrechner ausgelesenen Daten nicht auf einem Datenträger abspeichern. Hinzu kommt, daß in Bearbeitungszellen Web-Anwendungen vor allem auf Embedded Systems ablaufen, deren Speicherresourcen stark limitiert sind. Das Java-Servlet wurde deshalb so implementiert, daß nur die Daten im Speicher gehalten werden und an des Java-Applet übertragen werden, die für eine Visualisierung benötigt werden.

Die Visual-Basic-Anwendung verfügt als OPC-Client jedoch über alle vom Sensorrechner bereitgestellten Daten. Damit sind die Möglichkeiten zur Nachbearbeitung der Daten wesentlich umfangreicher. Beispielsweise können die eingestellten Toleranzgrenzen der Sensormeßwerte bzw. der Positionsabweichungen nachträglich verändert werden und anschließend die Daten neu verarbeitet werden. Außerdem ist es möglich die kartesischen Positionen und die Sensormeßwerte lückenlos zu visualisieren. In einem Fenster des beispielhaft realisierten OPC-Client-Programms werden alle erfaßten Positionen des Bewegungspfades des Bearbeitungswerkzeugs visualisiert. Da es sich um eine zweidimensionale Projektion handelt, kann der Benutzer die Projektionsebene auswählen (Bild 84).

In einem weiteren Fenster werden alle erfaßten Sensormeßwerte einer Bewegungssequenz dargestellt. Der Benutzer kann dabei über mehrere Zoomstufen relevante Bereiche zeitlich feiner auflösen (Bild 85).

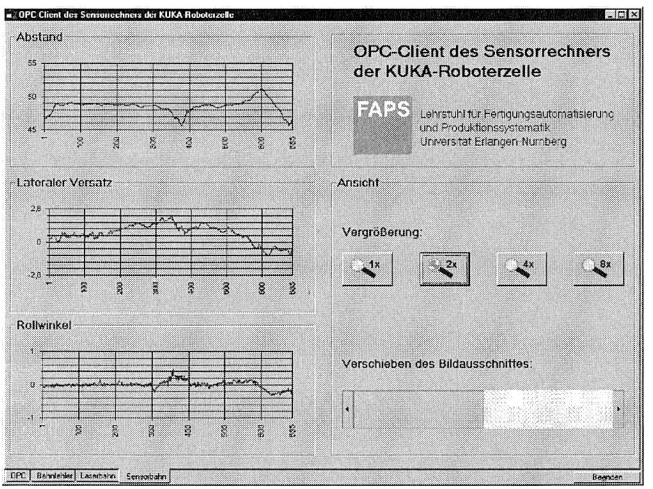


Bild 85: Visualisierung aller erfaßten Sensormeßwerte durch den OPC-Client

Neben den Möglichkeiten zur Visualisierung und Verarbeitung der Daten des Sensorrechners, können diese auch auf einem Datenträger abgelegt bzw. in eine Datenbank eingetragen werden. Vor allem bei der Produktion sicherheitsrelevanter Baugruppen ist diese Funktionalität, im Hinblick auf das Produkthaftungsgesetz, unbedingt erforderlich. Darüber hinaus können die Daten in anderen Softwareanwendungen weiter verarbeitet werden. Denkbare Anwendungen reichen von einer statistischen Prozeßkontrolle auf Zellenebene bis zur Nutzung von Produktionsdaten für die Produktionsplanung und -steuerung.

10 Zusammenfassung

Die meisten Märkte zeigen heute Sättigungserscheinungen, die einen Wandel vom quantitativen zum qualitativen Wachstum bewirken. Nicht mehr Mengenwachstum, sondern eine zunehmende Anzahl von Varianten mit abnehmenden Losgrößen kennzeichnen die Produktion. In engem Zusammenhang damit steht eine steigende Innovationsdynamik, die immer kürzere Produktlebenszyklen zur Folge hat.

Kleinserienfertigung und "Build-to-order"-Strategien sind aktuelle Trends, die eine hohe Flexibilität der Produktionsanlagen und Automatisierungssysteme erfordern. Industrieroboter bieten den erforderlichen Flexibilitätsgrad. Eine Analyse der Fehlereinflußfaktoren in Roboterzellen zeigte jedoch, daß Roboter- und Werkzeugungenauigkeiten, Werkstücktoleranzen und Programmierfehler einen nicht zu vernachlässigenden Einfluß auf die Bearbeitungsqualität haben.

Moderne Bearbeitungstechnologien, wie die Lasermaterialbearbeitung, stellen hohe Ansprüche an die exakte Einhaltung der prozeßtechnisch erforderlichen geometrischen Beziehungen zwischen Werkzeug und Werkstück. In roboterbasierten Produktionsanlagen sind hohe Genauigkeitsanforderungen nur durch den Einsatz von Sensorik erreichbar. Die Sensordaten sind dabei für die Korrektur der Bewegungsbahn vor der Bearbeitung bzw. während der Bearbeitung nutzbar. Eine Analyse des Spektrums an verfügbaren Sensoren, unter besonderer Berücksichtigung wirtschaftlicher Rahmenbedingungen, ergab eine Fokussierung auf einige wenige Sensortypen. Die industrielle Praxis bestätigt dieses Ergebnis.

Die rasante Entwicklung der Mikroelektronik führte in den letzten Jahren zu einer wesentlichen Leistungssteigerung im Sensorikbereich. Geringe Meßraten, die früher die Einsatzfelder von Sensorik stark einschränkten, konnten in dem erforderlichen Maße gesteigert werden. Problematisch ist jedoch immer noch die Integration und Verarbeitung der Sensordaten in den Robotersteuerungen. Fehlende Schnittstellenstandards und proprietäre Softwareprodukte erfordern einen unverhältnismäßig hohen Engineeringaufwand. Die Realisierung einer Sensoranbindung wird deshalb häufig an eine spezifische Problemstellung adaptiert und verfügt nur über einen geringen Wiederverwendungsgrad.

Ein wesentliches Ziel dieser Arbeit war die Entwicklung und Umsetzung eines offenen Integrations- und Verarbeitungsansatzes für Sensordaten. Da der Begriff der "Offenheit" unterschiedlich ausgelegt wird, wurden die wesentlichen Kriterien, die ein Offenes System charakterisieren, deutlich herausgearbeitet.

Da die Leistungsfähigkeit moderner Automatisierungslösungen heute maßgeblich durch die eingesetzte Software bestimmt wird, wurde auf die Konzeption und Entwicklung einer offenen Softwarearchitektur besonderer Wert gelegt. Dabei wurde

ein Komponentenansatz verfolgt, der sowohl die Offenheit als auch die notwendige Flexibilität der Softwarearchitektur unterstützt.

Das realisierte Softwaresystem besteht aus unterschiedlichen Softwaremodulen, die über spezifische Funktionalitäten verfügen. Als Erweiterung des klassischen Komponentenmodells werden die benötigten Softwaremodule erst zur Laufzeit, zu dem eigentlichen Softwaresystem konfiguriert. Die dynamische Konfigurierbarkeit erhöht den Flexibilitätsgrad des Softwaresystems wesentlich. Eine derartige Softwarearchitektur verfügt über einen hohen Wiederverwendungsgrad und damit zu reduzierten Entwicklungs- und Inbetriebnahmekosten.

Neben systemrelevanten Softwaremodulen, wie einem Konfigurator oder einer Ablaufsteuerung, wurden Softwaremodule zur Integration und Verarbeitung von Sensordaten entwickelt. Diese werden durch Parametrierung an anwendungsspezifische Anforderungen angepaßt.

Zum Nachweis der industriellen Einsatzfähigkeit wurde eine Nahtfolgeanwendung aufgebaut. Die Entwicklung des Nahtfolgealgorithmus fand unter Verwendung eines 3D-Kinematik-Simulationssystems statt. Dazu war es erforderlich die Funktionalität des verwendeten Lichtschnittsensors in dem Simulationssystem abzubilden. Um möglichst realitätsnahe Simulationsergebnisse erzielen zu können, wurden bei der Modellierung auch dynamische Eigenschaften von Roboter und Sensorsystem berücksichtigt. Mit der aufgebauten Simulationsumgebung war es möglich, die freien Parameter des Nahtfolgealgorithmus effizient zu optimieren.

Aufbauend auf den Simulationsstudien wurde eine Bearbeitungszelle zum Laserstrahlhartlöten realisiert. Anhand einer industriellen Aufgabenstellung konnte die Industrietauglichkeit nachgewiesen werden. Dies war die Grundlage für eine erfolgreiche Vermarktung des in dieser Arbeit entwickelten Gesamtsystems.

Zur Abrundung des Gesamtkonzepts wurde ein Web-Server und ein OPC-Server implementiert. Damit ist die informationstechnische Integration der roboterbasierten Bearbeitungszelle in die innerbetriebliche Informationsverarbeitung möglich. Da Information in Zukunft einen der wichtigsten Produktionsfaktoren darstellt, kommt dieser Funktionalität eine besondere Bedeutung zu.

Eine erhöhte industrielle Akzeptanz des Sensoreinsatzes ist sicherlich dann zu erwarten, wenn die Integration und Verarbeitung von Sensordaten in Robotersteuerungen mit einem akzeptablen technischen und wirtschaftlichen Aufwand realisierbar ist. Mit dieser Arbeit wurde ein Beitrag zur Förderung des Sensoreinsatzes geleistet, um eine höhere Produktivität und Qualität in roboterbasierten Produktionsanlagen zu erzielen.

Literaturverzeichnis

1. Altintas, Y.; Munasinghe, W. K.: A Hierarchical Open-Architecture CNC system for machine tools, Annals of the CIRP, 43 (1994) 1, S. 349-354
2. ANSI/RIA 15.05-1-1990: Robot systems - Point to Point and Static Performance Charcteristics, New York, 1989
3. Ausschuß für wirtschaftliche Fertigung (Hrsg.), Integrierter EDV-Einsatz in der Produktion - CIM, AWF Eschborn, 1984
4. Backes, F.: Technologieorientierte Bahnplanung für die 3D-Laserstrahlbearbeitung, Reihe Fertigungstechnik Erlangen, Band 68, Meisenbach Verlag, Bamberg, 1997
5. Backhaus, K.; Erichson, B.; Plinke, W.; Weiber, R.: Multivariante Analysemethoden - Eine anwendungsorientierte Einführung, Springer Verlag, Berlin, 1996
6. Barg, J.: Specification of Reference Architecture, OSACA Deliverable D121, ESPRIT III-CIME IV.1.2, Project 6379, Selbstverlag der Europäischen Union, Brüssel, 1993
7. Bauer, L.; Trunzer, W.: 3D-Laseranlagen off-line programmieren, Schweizer Maschinenmarkt, 96 (1995) 41, S. 22-25
8. Bauernfeind, H.: Einsatz von FEM-Simulation im Werkzeugmaschinenbau, In: iwb Seminarberichte, Band 47: Virtuelle Produktion - Prozeß- und Produktsimulation, Herbert Utz Verlag, München, 1999
9. Begemann, C.; von Cieminzki, G.: Produktion auf der Überholspur, phi, (2001) 4, S. 3-5
10. Berg, J.O.: Robot Accuracy: A Matter of Programming, Internat. Journal of Advanced Manufacturing Technology, (1992) 7, S. 193-197
11. Berners-Lee, T., et al.: Hypertext Transfer Protocol - HTTP/1.1 RFC 2616, Network Working Group, 1999
12. Bernstein, P.A.: Middleware: A Model for Distributed System Services, Communications of the ACM, 39 (1996) 2, S. 86-98
13. Bestle, D.: Analyse und Optimierung von Mehrkörpersystemen, Springer Verlag, Berlin, 1994
14. Blank, H-J.: Sensoren am PC, Verlag Markt&Technik, Haar bei München, 1996
15. Bolliger, R.: Rohstoff Software - Bedeutung der Software in der Wertschöpfungskette, MegaLink, 8 (2001) 8, S. 52-55

16. Brinch-Hansen, P.: Operating System Principles, Verlag Prentice Hall, Englewood Cliffs, 1973

17. Brodtmann, T.: Robotik und Automation - Portrait der Branche 2001, VDMA Frankfurt, 2001

18. Broy, M.; Spaniol, O.: VDI-Lexikon Informatik und Kommunikationstechnik, Springer Verlag, Berlin, 1999

19. Buchwitz, M.: Das IDA-Konzept: Weit mehr als Ethernet, Computer&Automation, (2001) 7-8, S. 46-50

20. Bues, M.: Offene Systeme: Strategien, Konzepte und Techniken für das Informationsmanagement, Springer Verlag, Berlin, 1994

21. Chrysler, Ford Motor Co., General Motors: Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry, White Paper - Version 1.1, December 1994

22. Colombo, W.: Development and Implementation of Hierarchical Control Structures of Flexible Production Systems Using High-Level Petri Nets, Reihe Fertigungstechnik Erlangen, Band 82, Meisenbach Verlag, Bamberg, 1998

23. Daniel, C.: Dynamisches Konfigurieren von Steuerungssoftware für offene Systeme, Forschung und Praxis, Band 112, Springer Verlag, Berlin, 1996

24. Demmer, A.; Zaboklicki, A. K.: Licht aus dem Halbleiter - Hochleistungs-Diodenlaser für die Produktionstechnik, VDI-Z, 139 (1997) 6, S. 8-9

25. Dijkstra, E.: Co-operating Sequential Processes, In: Genyus, F. (Hrsg.): Programming Languages, Academic Press, 1967, S. 43-112

26. DIN 19237: Steuerungstechnik, Begriffe, Beuth Verlag, Berlin, 1985

27. DIN 19240: Messen, Steuern, Regeln - Peripherieschnittstellen elektronischer Steuerungen, Beuth Verlag, Berlin, 1985

28. DIN 44300: Informationsverarbeitung, Teile 1-9: Begriffe, Beuth Verlag, Berlin, 1988

29. DIN EN 50170/2: Universelles Feldkommunikationssystem, Band 2/3: Profibus, Beuth Verlag, Berlin, 1997

30. DIN 66025, Programmaufbau für numerisch gesteuerte Arbeitsmaschinen, Beuth Verlag, Berlin, 1983

31. DIN 66259-1: Elektrische Eigenschaften der Schnittstellenleitungen; Doppelstrom, unsymmetrisch bis zu 20 kbit/s, Beuth Verlag, Berlin, 1981

32. Eddon, G.: Inside distributed COM, Microsoft Press, Redmond USA, 1998

33. Feldmann, K.; Slama, S.: Highly Flexible Assembly - Scope and Justification, Annals of the CIRP, 50 (2001) 2, Zur Veröffentlichung freigegeben

34. Feldmann, K.; Wenk, M.; Zeller, J.: Fast Sensor Guidance of Industrial Robots - Experimental Results, Robotica, (1999) 17, S. 17-21

35. Feldmann, K.; Wenk, M.: Simplified Mechanisms for the Integration and Processing of Multidimensional Sensor Data in Robot-Based Manufacturing Systems, In: Proceedings of the 9th International Conference for Sensors, Transducers and Systems, Nürnberg, 20.05.1999, S. 479-484

36. Feldmann, K.; Wenk, M.: Konfigurierbare Softwarearchitektur für die einfache Integration und Verarbeitung von Sensorsignalen in Robotersteuerungen, VDI-Tagung Robotik 2000, Berlin, 29.-30.06.2000, In: VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (Hrsg.): VDI-Berichte 1552 -Robotik 2000, VDI-Verlag, Düsseldorf, 2000, S. 391-396

37. Feldmann, K.; Wenk, M.: Flexible Sensor Control Solutions for Robot Controls by Reconfigurable Software Architectures, In: The American Society of Mechanical Engineers (Hrsg): Proc. of the 2000 Japan-USA Symposium on Flexible Automation, Ann Arbor USA, 23.-26.07.2000, ASME, New York, 2000

38. Feldmann, K.; Wenk, M.: Configurable Software Architecture for Easy Integration and Processing of Sensor Data in Robot Controls, Production Engineering, VII/2, Annals of the German Academic Society for Production Engineering, 2000, S. 125-128

39. Feldmann, K.; Wenk, M.: Open Control System Based on Software Components and Web-enabling Technologies, In: Proceedings of the 2nd International Symposium on Open Control Systems 2001, Helsinki, 03.09.2001, Eigenverlag University of Tampere, CD-ROM Edition

40. Free Software Foundation, http://www.fsf.org/home.de.html

41. Fujita, S.; Yoshida, T.: OSE: Open System Environment for Controller, Proc. of the 7th Int'l. Machine Tool Engineers Conf., Nov. 16-17, 1996, S. 234-243

42. Geitner, U. (Hrsg.): CIM-Handbuch, Verlag Vieweg, Wiesbaden, 1987

43. Glas, J.: Standardisierter Aufbau anwendungsspezifischer Zellenrechnersoftware, Forschungsberichte, Band 61, Springer Verlag, Berlin, 1993

44. Gosling, J.; Joy, B.; Steele, G.; Bracha, G.: The Java Language Specification, Second Edition, Sun Microsystems, 2000

45. Gräser, R.G.: Ein Verfahren zur Kompensation temperaturinduzierter Verformungen an Industrierobotern, iwb-Forschungsbericht 125, Herbert Utz Verlag, München, 1999

46. Grötsch, E.; PC-based Automation – 7 Trends, Totally Integrated Automation, (2000) 3, S. 8-10

47. Guelich, S.; Gundavaram, S.; Birzniek, G.: CGI-Programmierung mit Pearl, Verlag O'Reilly, 2001

48. Haasis, S.: Integrierte CAD-Anwendungen - Rationalisierungspotentiale und zukünftige Einsatzgebiete, Springer Verlag, Berlin, 1995

49. Hänel, R.: Effektiver Robotereinsatz verkürzt Stillstandzeiten, A&D-Newsletter, (2000) 5/6, publish-industry Verlag, München, S. 22

50. Hagmann, M.; Schäfer, G.; Bracht, U.: Das integrierte Ebenenkonzept als neuer Ansatz in der Fabrik- und Anlagenplanung, In: Tagungsband zum 12. Symposium Simulationstechnik 9/98, Hochschulverlag ETH Zürich, S. 267 – 274

51. Happacher, M.: Der steinige Weg zur Offenen Steuerung, Elektronik, München, 46 (1997) 24, S. 68-74

52. Hanebuth, H.: Laserstrahllöten mit Zweistrahltechnik, Fertigungstechnik Erlangen, Band 55, Meisenbach Verlag, Bamberg, 1996

53. Heisel, U. L.; Reinhart, G.: Eine Fehlerquelle wird beseitigt - Einfluß des thermischen Verhaltens auf die Arbeitsgenauigkeit, Roboter, (1995) 8, S. 32-34

54. Himstedt, S.: Release 2.0 - OLE for Process Control wird erwachsen, SPS-Magazin, 12 (1999) 8/9, S. 26,28

55. Hoare, C.: Monitors: An Operating System Structuring Concept, Communications of the ACM, 17 (1974) 10, S. 549-557

56. Holling, G.H.: Abschätzung von Bahnfehlern in Robotersystemen, Vieweg Verlag, Braunschweig, 1990

57. Homepage des PC/104-Konsortiums, http://www.pc104.org

58. Homepage der Firma Rofin-Sinar Laser GmbH, http://www.rofin-sinar.com

59. Horn, A.: Optische Sensorik zur Bahnführung von Industrierobotern mit hohen Bahngeschwindigkeiten, ISW Forschung und Praxis, Band 103, Springer Verlag, Berlin, 1994

60. IEC 14750, Information technology - Open Distributed Processing - Interface Definition Language, 1999

61. IEC 61131-3, Programmable controllers - Part 3: Programming languages, International Electrotechnical Commission, 1993

62. IEEE STD 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, 1985

63. IEEE STD 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE, 1990

64. IEEE STD 1003.0-1995, IEEE Guide to the POSIX® Open System Environment (OSE) (Identical to ISO/IEC TR 14252), IEEE, 1995

65. ISO/IEC 7498, Informationstechnik - Kommuniktion Offener Systeme, Beuth Verlag, Berlin, 1994

66. ISO 9283: Manipulating industrial robots - Performance criteria and related test methods, Beuth Verlag, Berlin, 1998

67. Iwanitz, F.; Lange, J.: OLE for Process Control, Grundlagen, Implementierung, Anwendung, Hüthig Verlag, Heidelberg, 2001

68. Jecht, U.: Prozeßanalysegeräte jetzt feldbusfähig - ein neues Profil für Profibus PA und seine Implementierung in Geräte, Automatisierungstechnische Praxis - atp, 42 (2000) 7, S. 33-34, 37-42

69. Jochem, M.; Deisenroth, J.: Offenheit als Mittel zur Kosteneinsparung, A&D Kompendium 2001, publish-industry Verlag, München, 2001, S. 55-57

70. Kabitzsch, K.: Programmierung verteilter Systeme, Design&Elektronik, (1991) 21, S.86-91

71. Koren, Y.; Jovane, F.; Pritschow, G.: Open Architecture Control Systems - Summary of Global Activity, ITIA Series, Vol. 2, 1998

72. Kugler, P.; Wenk, M.: Low cost Lasersystem für die Materialbearbeitung von Leichtbauwerkstoffen, In: Geiger, M.; Lenfert, K. (Hrsg.): Innovationen durch Laser Engineering - Berichtsband zum Zwischenkolloquium FORLAS II, Meisenbach Verlag, Bamberg, 1999, S. 113-133

73. Lange, F.; Hirzinger, G.: Learning to Improve the Path Accuracy of Position Controlled Robots, Proc. of Int. Conf. on Intelligent Robots and Systems, München, 1994, S. 494-501

74. Lemme, H.: Sensoren in der Praxis, Franzis Verlag, München, 1990

75. Lutz, P.; Sperling, W.: OSACA the vendor neutral Control Architecture, Proc. of the European Conf. on Integration in Manufacturing Iim'97, Selbstverlag der TU Dresden, 1997

76. Lutz, H., Wendt, W.: Taschenbuch der Regelungstechnik, Verlag Harri Deutsch, Thun und Frankurt a. Main, 1995

77. Marietta, M.: Next Generation Controller (NGC) – Specification for an Open System Architecture Standard, Manufacturing Technology Directorate, Wright Laboratory, 1994

78. Meyberg, K.; Vachenauer, P.: Höhere Mathematik 1 - Differential- und Integralrechnung, Springer Verlag, Berlin, 1990

79. Mollath, G., Nickolay, B.: Untersuchung von Systemen zur mehrdimensionalen Kraft- und Momentenmessung, In: Rogos, J. (Hrsg.): Intelligente Sensorsysteme in der Fertigungstechnik, Springer Verlag, Berlin, 1989

80. Müller, M.: Ungewöhnliche Gedanken zur Steuerungstechnik – SPS oder PC-Based Automation, MessTec&Automation, (2001) 1/2, S. 32-35

81. National Center of Manufacturing Science, Next Generation Controller (NGC) – Specification for an Open System Architecture Standard, Rev 2.5, 1994

82. N.N.: OPC Server Toolkit Programmierhandbuch Version 2.00, Softing GmbH, Haar b. München, 1999

83. N.N.: Roboter kennen keine Konjunktursorgen, Fabrikautomation, (2001) 2, S. 8-10

84. N.N.: Feldbus-Vielfalt: Automatisierung am Scheideweg, Markt&Technik, (2000) 33, S. 40-42

85. N.N.: Trend: exakte Simulation kompletter Fabriken, Markt&Technik, (2001) 39, S. 38-41

86. Object Management Group (OMG): The Common Object Request Broker: Architecture and Specification, Rev. 2.3.1, OMG Doc. formal/99-10-07, Oct. 1999

87. OPC Foundation, http://www.opcfoundation.org

88. Parnas, D. L.: On the Criteria to be Used in Decomposing Systems into Modules, Communication of the ACM, 15 (1972) 12, S. 1053 - 1058

89. Patterson, D.A.: Mikroprozessoren im Jahre 2020, In: Spektrum der Wissenschaft, Spezial 4: Schlüsseltechnologien, Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 1995

90. Pritschow, G.; Klingel, H.; Bauder, M.; Horn, A.: Erhöhung der Bahngenauigkeit von Industrierobotern, Robotersysteme, (1992) 8, S. 162-170

91. Proctor, F.: Michaloski, J.: Enhanced Machine Controller Architecture Overview, Technical Report 5331, National Institute of Standards and Technology (NIST), 1993

92. Qu, Z.; Kauppila, J.; Moisio, T.: A Seam Tracking System for Sheet Metal Pipe Fabrication by Laser Beam Welding, Proc. of Int. Conf. Laser Assisted Net Shape Engineering (LANE), Erlangen, 1994, S. 741-748

93. Ragett, D.; Le Hors, A.; Jacobs, I.: HTML 4.0 Specification, W3C Recommendation, 24. December 1999

94. Reinhart, G.; Dürrschmidt, S.; Hirschberg, A.; Krüger, A.: Strategien gegen Turbulenzen - Wettbewerbsfaktor: Stückzahl- und Variantenflexibilität, Produktion (1999) 38/39, S. 3

95. Reinhart, G.; Klingel, R.: Produktivitätssteigerung durch gezieltes Informationsmanagement auf Mitarbeiterebene, In: Reinhart, G. (Hrsg.): Montage-Management. Lösungen zum Montieren am Standort Deutschland, Transfer-Centrum GmbH, München, 1998, S. 32-36

96. Richtlinie VDI 2861, Blatt 2: Kenngrößen für Industrieroboter - Einsatzspezifische Kenngrößen, VDI-Verlag, Düsseldorf 1988

97. Röder, L.: Komponentenentwicklung mit dem "Component Object Model", ObjektSpektrum, (1999) 3, S. 18-25

98. Rogos, J.: Die Bedeutung der Deutschen Vornorm DIN V 66311 "Sensorschnittstelle für Roboter und Fertigungssysteme" und Zukunftsaspekte, In: Rogos, J. (Hrsg.): Intelligente Sensorsysteme in der Fertigungstechnik, Springer Verlag, Berlin, 1989

99. Seidel, M.: Schnelle, sensorgeführte Konturbearbeitung mit Industrierobotern, Fortschrittsbericht VDI-Reihe 2, Nummer 277, VDI-Verlag, Düsseldorf, 1992

100. Schanz, G.W.(Hrsg.): Sensortechnik aktuell - Ausgabe 2002 -Trends, Produkte und Entscheidungshilfen, Oldenbourg Industrieverlag, München, 2002

101. Scharf, A.: PC hat Produktion preiswert im Griff, VDI Nachrichten, (1999) 45, S. 15

102. Schauerhuber, M.: Produktionswirtschaftliche Flexibilität, Schriftenreihe "Forschungsergebnisse der Wirtschaftsuniverstät Wien", Service Fachverlag, Wien, 1998

103. Schlögl, W.: Integriertes Simulationsdaten-Management für Maschinenentwicklung und Anlagenplanung, Reihe Fertigungstechnik Erlangen, Band 101, Meisenbach Verlag, Bamberg, 2000

104. Schmid, D.: Sensor simuliert Werkzeug, Roboter, (1990) 2, S. 14-17

105. Schneider, G.: Feuerprobe bestanden - PC-Technologie in der Robotersteuerungstechnik erfolgreich eingeführt, Elektronik, 47 (1998) 8, S. 82-88

106. Schneider, H-J.: Lexikon der Informatik und Datenverarbeitung, 4. Auflage, R. Oldenbourg Verlag, München, 1997

107. Son, S-Y.; Olsen, T.L.; Yip-Hoi, D.: Economic Benefits of Reconfigurable Manufacturing Systems, Proc. of the 2000 Japan-USA Symposium on Flexible Automation (2000 JUSFA), International Conference on Manufacturing Systems: Innovations for the 21th Century, July 23-26, 2000, Ann Arbor, MI, USA

108. Stahl, M.: Des Knaben Wunderhorn, OBJEKTspektrum, (1999) 1, S. 18-20

109. Schwarz, A.: Software aus dem Baukasten - Technik und Funktionsweise von OLE, Elektronik, München, 45 (1996) 25, S. 84-90

110. Schwarz, M.: "Moment, ich verbinde...", c't - Magazin für Computertechnik, (1997) 3, S. 256-273

111. Tradowsky, K.: Laser - Grundlagen, Technik, Basisanwendungen, 4. Auflage, Vogel-Verlag, Würzburg, 1988

112. Unseld, R.: Der IPC- und SPS-Markt: Geräteklassen verschwimmen, Markt&Technik, (2001) 11, S. 34-36

113. Wenk, M.: Rationalisierungspotentiale in der Montage durch PC-basierte Steuerungskonzepte, In: VDI-Bildungswerk (Hrsg.): Auslegung und Betrieb flexibler Montagesysteme, VDI-Verlag, Düsseldorf, 1998

114. Wenk, M.; Hoffmann, P.; Kugler, P.: Low Cost Lasersystem für die Materialbearbeitung von Leichtbauwerkstoffen, In: Geiger, M.; Lenfert, K. (Hrsg.): Innovationen durch Laser Engineering II - Berichtsband zum Abschlußkolloqium FORLAS II, Erlangen, 19.07.2000, Meisenbach Verlag, Bamberg, 2000, S. 113-136

115. Wenzel, M.: The Importance of the Control for Robotic Path Welding, Proc. of the 25th Int. Symp. in Industrial Robots, Hannover, 1994, S. 435-440

116. Wiendahl, H-P; Röhrig, M.; Hegenscheidt, M.: Beschäftigungsförderliche Rationalisierung, In: Forschungsberichte FZKA-PFT, Förderungsprogramm Fertigungstechnik des BMFT, Band 201, 1999

117. Wind River Systems Inc.: VxWorks Programmer's Guide, 5.3.1, Edition 1, 1997

118. Wloka, D.: Robotersysteme I, Technische Grundlagen, Springer Verlag, Berlin, 1992

119. Wohlschläger, R.: IEC61161-3 weltweit akzeptiert?, SPS Magazin, (2001) 4/5, Technik-Dokumentations-Verlag, Marburg, 2001

120. Yen, J.: Overview of OMAC Activities and Global Open Architecture Control Development Effort, Konferenz zur „Trendwende in der Steuerungstechnik", EMO, Hannover, 1997

121. Zeller, F-J.: Sensorplanung und schnelle Sensorregelung für Industrieroboter, Reihe Fertigungstechnik Erlangen, Band 51, Carl Hanser Verlag, München, 1996

122. Zhao, W.: Sensorgeführte Industrieroboter zur Bahnverfolgung, Reihe Produktionstechnik Berlin, Band 79, Carl Hanser Verlag, München, 1990

Abkürzungen und Formelzeichen

Verzeichnis verwendeter Abkürzungen:

Abkürzungen, die im Text erklärt werden, wurden nicht in dieses Verzeichnis aufgenommen.

ASCII	American Standard Code for Information Interchange
DIN	Deutsches Institut für Normung e.V.
DRAM	Dynamic Random Access Memory
DSP	Digitaler Signalprozessor
ESPRIT	European Strategic Program for Research in Information Technologies
FTP	File Transfer Protocol
FORLAS	Forschungsverbund Lasertechnik
IEEE	Institute of Electrical and Electronic Engineers
ISA-Bus	Industry Standard Architecture Bus
IT	Informationstechnologie
PCI-Bus	Peripheral Component Interconnect Bus
SCADA	Supervisory Control and Data Acquisition
SDB	Sensor-Daten-Bearbeitung
SCP	Nullpunkt des Sensorkoordinatensystems
SI	Systeme International d'Unites
SM	Softwaremodul
VGA	Video Graphics Array
VME	Versa Module Europe

Verzeichnis verwendeter Formelzeichen:

$\{\mathrm{A}\}$	Bezeichnung für Koordinatensystem A
$\mathbf{x}_A, \mathbf{y}_A, \mathbf{z}_A$	Koordinatenachsen von $\{\mathrm{A}\}$
$\{S\}$	Sensorkoordinatensystem
$\{W\}$	Werkzeugkoordinatensystem

$\{R\}$	Roboterkoordinatensystem
$\boldsymbol{p}_A$	Positionsvektor bzgl. {A}
${}^A\mathbf{o}_B$	Translationsvektor des Ursprunges von {B} bezgl. {A}
$\mathbf{T}$	Matrix T
${}^A_B\mathbf{T}$	Homogene Transformation von {A} nach {B}
${}^A_B\mathbf{R}$	Rotationsmatrix von {A} nach {B}
$Rot(\,)$	Rotationsoperator
$\alpha, \beta, \gamma, \delta, \lambda, \mu$	Drehwinkel
T_{sync}	Synchronisationszeit zwischen RC und Sensorrechner
T_{verz}	Verzögerung zwischen Meßzeitpunkt und Beginn IPO-Takt
T_m	Zeitspanne für Meßwertaufnahme
T_w	Zeitstempel bei Meßwertaufnahme
T_f	Zeitspanne bis zum Empfang von Feldbusdaten
T_{IPO}	Interpolationszykluszeit
T_t	Totzeit
T_1	Verzögerungszeitkonstante der Regelstrecke
G(s)	Übertragungsfunktion der Regelstrecke
K_P	Proportionalverstärkung der Regelstrecke
K_I	Integrierverstärkung der Regelstrecke
v_B	Bahngeschwindigkeit
RV_{opt}	optimaler Vorlauf
a_R	Achsenabschnitt der Regressionsgeraden
b_R	Regressionskoeffizient der Regressionsgeraden
i	Laufvariable
x_i, y_i	sensorisch erfaßte Kantenpunkte
$\bar{x}, \bar{y}$	arithmetische Mittel aus allen x_i, y_i
κ	Kippwinkel
$(x_{SCP}, y_{SCP}, z_{SCP})^T$	Verschiebungsvektor des SCP aus dem TCP
m_x, m_y, m_z	Sensormeßwerte bezüglich $\{S\}$
w_x, w_y, w_z	Sensormeßwerte bezüglich $\{W\}$

Summary

Industrial robots offer a high degree of flexibility. Therefore, they are increasingly used in manufacturing automation. But the high number of fault possibilities is problematic. There are e.g. inaccuracies within the robot kinematics, the tools and the workpieces.

On the other hand, modern manufacturing technologies require a high accuracy of the handling machine. In robot based manufacturing cells the accuracy demands are only realizable with a sensor guidance of the robot. But the integration and processing of sensor data in robot controls is difficult. The reasons are the lack of standardisation in data interfaces and the great variety of application specific processing algorithms. The consequence are expensive manufacturer specific solutions on the market.

In this dissertation a new approach was developed, which is based on a interface card. This embedded system will be integrated into a robot control. Thus, it is the linking element between the robot control and the sensor system. The integration of sensor data is done by a fieldbus system. The processing of sensor data is handled by a modular software system, running on the embedded system. The functionality of the software system could be changed by replacing functionality modules. With this mechanism the software system could be adapted easily to any application specific effort.

For realising a high degree of flexibility the component technology was used. The functionality modules are realized by software components. The enduser chooses the needed software components out of a software library. A special component, the so called configurator, is then building up the software system with the choosen software components. The enduser does not have to implement one line of sourcecode. He has only to define the configuration of the software system. This could be done user-friendly within a graphical user interface.

With that approach the engineering costs for development and start-up of sensor control solutions could be reduced. That will enforce the use of sensor systems within robot based applications. Thereby, the availability of the production system and the quality of produced parts will be increased.

Lebenslauf

Matthias Wenk
geboren am 14. Januar 1969 in Treuchtlingen/Bay.
verheiratet mit Monika Wenk, geb. Steindl, 2 Kinder

09/1975 - 07/1979	Grundschule in Gunzenhausen
09/1979 - 05/1988	Simon-Marius-Gymnasium in Gunzenhausen Abschluß: Abitur
07/1988 - 09/1989	Grundwehrdienst in Feuchtwangen
10/1989 - 09/1995	Studium der Elektrotechnik an der Universität Erlangen-Nürnberg Abschluß: Dipl.-Ing. (Univ.)
seit 10/1995	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik an der Universität Erlangen-Nürnberg Leiter: Prof. Dr.-Ing. Klaus Feldmann
seit 10/1998	Oberingenieur der Forschungsgruppe Planung und Steuerungstechnik

Reihe Fertigungstechnik - Erlangen

www.mb.uni-erlangen.de

Band 1 - 52
Carl Hanser Verlag, München

ab Band 53
Meisenbach Verlag, Bamberg
45,-- Euro

Band 1: Andreas Hemberger
Innovationspotentiale in der rechnerintegrierten Produktion durch wissensbasierte Systeme
208 Seiten, 107 Bilder. 1988.

Band 2: Detlef Classe
Beitrag zur Steigerung der Flexibilität automatisierter Montagesysteme durch Sensorintegration und erweiterte Steuerungskonzepte
194 Seiten, 70 Bilder. 1988.

Band 3: Friedrich–Wilhelm Nolting
Projektierung von Montagesystemen
201 Seiten, 107 Bilder, 1 Tabelle. 1989.

Band 4: Karsten Schlüter
Nutzungsgradsteigerung von Montagesystemen durch den Einsatz der Simulationstechnik
177 Seiten, 97 Bilder. 1989.

Band 5: Shir–Kuan Lin
Aufbau von Modellen zur Lageregelung von Industrierobotern
168 Seiten, 46 Bilder. 1989.

Band 6: Rudolf Nuss
Untersuchungen zur Bearbeitungsqualität im Fertigungssystem Laserstrahlschneiden
206 Seiten, 115 Bilder, 6 Tabellen. 1989.

Band 7: Wolfgang Scholz
Modell zur datenbankgestützten Planung automatisierter Montageanlagen
194 Seiten, 89 Bilder. 1989.

Band 8: Hans–Jürgen Wißmeier
Beitrag zur Beurteilung des Bruchverhaltens von Hartmetall–Fließpreßmatrizen
179 Seiten, 99 Bilder, 9 Tabellen. 1989.

Band 9: Rainer Eisele
Konzeption und Wirtschaftlichkeit von Planungssystemen in der Produktion
183 Seiten, 86 Bilder. 1990.

Band 10: Rolf Pfeiffer
Technologisch orientierte Montageplanung am Beispiel der Schraubtechnik
216 Seiten, 102 Bilder, 16 Tabellen. 1990.

Band 11: Herbert Fischer
Verteilte Planungssysteme zur Flexibilitätssteigerung der rechnerintegrierten Teilefertigung
201 Seiten, 82 Bilder. 1990.

Band 12: Gerhard Kleineidam
CAD/CAP: Rechnergestützte Montagefeinplanung
203 Seiten, 107 Bilder. 1990.

Band 13: Frank Vollertsen
Pulvermetallurgische Verarbeitung eines übereutektoiden verschleißfesten Stahls
XIII u. 217 Seiten, 67 Bilder, 34 Tabellen. 1990.

Band 14: Stephan Biermann
Untersuchungen zur Anlagen– und Prozeßdiagnostik für das Schneiden mit CO_2–Hochleistungslasern
VIII u. 170 Seiten, 93 Bilder, 4 Tabellen. 1991.

Band 15: Uwe Geißler
Material– und Datenfluß in einer flexiblen Blechbearbeitungszelle
124 Seiten, 41 Bilder, 7 Tabellen. 1991.

Band 16: Frank Oswald Hake
Entwicklung eines rechnergestützten Diagnosesystems für automatisierte Montagezellen
XIV u. 166 Seiten, 77 Bilder. 1991.

Band 17: Herbert Reichel
Optimierung der Werkzeugbereitstellung durch rechnergestützte Arbeitsfolgenbestimmung
198 Seiten, 73 Bilder, 2 Tabellen. 1991.

Band 18: Josef Scheller
Modellierung und Einsatz von Softwaresystemen für rechnergeführte Montagezellen
198 Seiten, 65 Bilder. 1991.

Band 19: Arnold vom Ende
Untersuchungen zum Biegeumformen mit elastischer Matrize
166 Seiten, 55 Bilder, 13 Tabellen. 1991.

Band 20: Joachim Schmid
Beitrag zum automatisierten Bearbeiten von Keramikguß mit Industrierobotern
XIV u. 176 Seiten, 111 Bilder, 6 Tabellen. 1991.

Band 21: Egon Sommer
Multiprozessorsteuerung für kooperierende Industrieroboter in Montagezellen
188 Seiten, 102 Bilder. 1991.

Band 22: Georg Geyer
Entwicklung problemspezifischer Verfahrensketten in der Montage
192 Seiten, 112 Bilder. 1991.

Band 23: Rainer Flohr
Beitrag zur optimalen Verbindungstechnik in der Oberflächenmontage (SMT)
186 Seiten, 79 Bilder. 1991.

Band 24: Alfons Rief
Untersuchungen zur Verfahrensfolge Laserstrahlschneiden und –schweißen in der Rohkarosseriefertigung
VI u. 145 Seiten, 58 Bilder, 5 Tabellen. 1991.

Band 25: Christoph Thim
Rechnerunterstützte Optimierung von Materialflußstrukturen in der Elektronikmontage durch Simulation
188 Seiten, 74 Bilder. 1992.

Band 26: Roland Müller
CO_2-Laserstrahlschneiden von kurzglasverstärkten Verbundwerkstoffen
141 Seiten, 107 Bilder, 4 Tabellen. 1992.

Band 27: Günther Schäfer
Integrierte Informationsverarbeitung bei der Montageplanung
195 Seiten, 76 Bilder. 1992.

Band 28: Martin Hoffmann
Entwicklung einer CAD/CAM–Prozeßkette
für die Herstellung von Blechbiegeteilen
149 Seiten, 89 Bilder. 1992.

Band 29: Peter Hoffmann
Verfahrensfolge Laserstrahlschneiden und –schweißen :
Prozeßführung und Systemtechnik in der 3D–Laserstrahlbearbeitung
von Blechformteilen
186 Seiten, 92 Bilder, 10 Tabellen. 1992.

Band 30: Olaf Schrödel
Flexible Werkstattsteuerung mit objektorientierten Softwarestrukturen
180 Seiten, 84 Bilder. 1992.

Band 31: Hubert Reinisch
Planungs– und Steuerungswerkzeuge
zur impliziten Geräteprogrammierung in Roboterzellen
XI u. 212 Seiten, 112 Bilder. 1992.

Band 32: Brigitte Bärnreuther
Ein Beitrag zur Bewertung des Kommunikationsverhaltens
von Automatisierungsgeräten in flexiblen Produktionszellen
XI u. 179 Seiten, 71 Bilder. 1992.

Band 33: Joachim Hutfless
Laserstrahlregelung und Optikdiagnostik
in der Strahlführung einer CO_2-Hochleistungslaseranlage
175 Seiten, 70 Bilder, 17 Tabellen. 1993.

Band 34: Uwe Günzel
Entwicklung und Einsatz eines Simulationsverfahrens für operative
und strategische Probleme der Produktionsplanung und –steuerung
XIV u. 170 Seiten, 66 Bilder, 5 Tabellen. 1993.

Band 35: Bertram Ehmann
Operatives Fertigungscontrolling durch Optimierung
auftragsbezogener Bearbeitungsabläufe in der Elektronikfertigung
XV u. 167 Seiten, 114 Bilder. 1993.

Band 36: Harald Kolléra
Entwicklung eines benutzerorientierten Werkstattprogrammiersystems
für das Laserstrahlschneiden
129 Seiten, 66 Bilder, 1 Tabelle. 1993.

Band 37: Stephanie Abels
Modellierung und Optimierung von Montageanlagen
in einem integrierten Simulationssystem
188 Seiten, 88 Bilder. 1993.

Band 38: Robert Schmidt–Hebbel
Laserstrahlbohren durchflußbestimmender Durchgangslöcher
145 Seiten, 63 Bilder, 11 Tabellen. 1993.

Band 39: Norbert Lutz
Oberflächenfeinbearbeitung keramischer Werkstoffe
mit XeCl–Excimerlaserstrahlung
187 Seiten, 98 Bilder, 29 Tabellen. 1994.

Band 40: Konrad Grampp
Rechnerunterstützung bei Test und Schulung
an Steuerungssoftware von SMD–Bestücklinien
178 Seiten, 88 Bilder. 1995.

Band 41: Martin Koch
Wissensbasierte Unterstützung der Angebotsbearbeitung
in der Investitionsgüterindustrie
169 Seiten, 68 Bilder. 1995.

Band 42: Armin Gropp
Anlagen– und Prozeßdiagnostik
beim Schneiden mit einem gepulsten Nd:YAG–Laser
160 Seiten, 88 Bilder, 7 Tabellen. 1995.

Band 43: Werner Heckel
Optische 3D–Konturerfassung und on–line Biegewinkelmessung mit dem Lichtschnittverfahren
149 Seiten, 43 Bilder, 11 Tabellen. 1995.

Band 44: Armin Rothhaupt
Modulares Planungssystem zur Optimierung der Elektronikfertigung
180 Seiten, 101 Bilder. 1995.

Band 45: Bernd Zöllner
Adaptive Diagnose in der Elektronikproduktion
195 Seiten, 74 Bilder, 3 Tabellen. 1995.

Band 46: Bodo Vormann
Beitrag zur automatisierten Handhabungsplanung komplexer Blechbiegeteile
126 Seiten, 89 Bilder, 3 Tabellen. 1995.

Band 47: Peter Schnepf
Zielkostenorientierte Montageplanung
144 Seiten, 75 Bilder. 1995.

Band 48: Rainer Klotzbücher
Konzept zur rechnerintegrierten Materialversorgung in flexiblen Fertigungssystemen
156 Seiten, 62 Bilder. 1995.

Band 49: Wolfgang Greska
Wissensbasierte Analyse und Klassifizierung von Blechteilen
144 Seiten, 96 Bilder. 1995.

Band 50: Jörg Franke
Integrierte Entwicklung neuer Produkt– und Produktionstechnologien für räumliche spritzgegossene Schaltungsträger (3-D MID)
196 Seiten, 86 Bilder, 4 Tabellen. 1995.

Band 51: Franz–Josef Zeller
Sensorplanung und schnelle Sensorregelung für Industrieroboter
190 Seiten, 102 Bilder, 9 Tabellen. 1995.

Band 52: Michael Solvie
Zeitbehandlung und Multimedia–Unterstützung in Feldkommunikationssystemen
200 Seiten, 87 Bilder, 35 Tabellen. 1996.

Band 53: Robert Hopperdietzel
Reengeneering in der Elektro- und Elektronikindustrie
180 Seiten, 109 Bilder, 1 Tabelle. 1996.
ISBN 3-87525-070-2

Band 54: Thomas Rebhan
Beitrag zur Mikromaterialbearbeitung mit Excimerlasern – Systemkomponenten und Verfahrensoptimierungen
148 Seiten, 61 Bilder, 10 Tabellen. 1996.
ISBN 3-87525-075-3

Band 55: Henning Hanebuth
Laserstrahlhartlöten mit Zweistrahltechnik
157 Seiten, 58 Bilder, 11 Tabellen. 1996.
ISBN 3-87525-074-5

Band 56: Uwe Schönherr
Steuerung und Sensordatenintegration für flexible Fertigungszellen mit kooperierenden Robotern
188 Seiten, 116 Bilder, 3 Tabellen. 1996.
ISBN 3-87525-076-1

Band 57: Stefan Holzer
Berührungslose Formgebung mit Laserstrahlung
162 Seiten, 69 Bilder, 11 Tabellen. 1996.
ISBN 3-87525-079-6

Band 58: Markus Schultz
Fertigungsqualität beim 3D–Laserstrahlschweißen von Blechformteilen
165 Seiten, 88 Bilder, 9 Tabellen. 1997.
ISBN 3-87525-080-X

Band 59: Thomas Krebs
Integration elektromechanischer CA–Anwendungen über einem STEP–Produktmodell
198 Seiten, 58 Bilder, 8 Tabellen. 1997.
ISBN 3-87525-081-8

Band 60: Jürgen Sturm
Prozeßintegrierte Qualitätssicherung in der Elektronikproduktion
167 Seiten, 112 Bilder, 5 Tabellen. 1997.
ISBN 3-87525-082-6

Band 61: Andreas Brand
Prozesse und Systeme zur Bestückung räumlicher elektronischer Baugruppen (3D-MID)
182 Seiten, 100 Bilder. 1997.
ISBN 3-87525-087-7

Band 62: Michael Kauf
Regelung der Laserstrahlleistung und der Fokusparameter einer CO_2-Hochleistungslaseranlage
140 Seiten, 70 Bilder, 5 Tabellen. 1997.
ISBN 3-87525-083-4

Band 63: Peter Steinwasser
Modulares Informationsmanagement in der integrierten Produkt– und Prozeßplanung
190 Seiten, 87 Bilder. 1997.
ISBN 3-87525-084-2

Band 64: Georg Liedl
Integriertes Automatisierungskonzept für den flexiblen Materialfluß in der Elektronikproduktion
196 Seiten, 96 Bilder, 3 Tabellen. 1997.
ISBN 3-87525-086-9

Band 65: Andreas Otto
Transiente Prozesse beim Laserstrahlschweißen
132 Seiten, 62 Bilder, 1 Tabelle. 1997.
ISBN 3-87525-089-3

Band 66: Wolfgang Blöchl
Erweiterte Informationsbereitstellung an offenen CNC–Steuerungen zur Prozeß– und Programmoptimierung
168 Seiten, 96 Bilder. 1997.
ISBN 3-87525-091-5

Band 67: Klaus–Uwe Wolf
Verbesserte Prozeßführung und Prozeßplanung zur Leistungs– und Qualitätssteigerung beim Spulenwickeln
186 Seiten, 125 Bilder. 1997.
ISBN 3-87525-092-3

Band 68: Frank Backes
Technologieorientierte Bahnplanung für die 3D–Laserstrahlbearbeitung
138 Seiten, 71 Bilder, 2 Tabellen. 1997.
ISBN 3-87525-093-1

Band 69: Jürgen Kraus
Laserstrahlumformen von Profilen
137 Seiten, 72 Bilder, 8 Tabellen. 1997.
ISBN 3-87525-094-X

Band 70: Norbert Neubauer
Adaptive Strahlführungen für CO_2-Laseranlagen
120 Seiten, 50 Bilder, 3 Tabellen. 1997.
ISBN 3-87525-095-8

Band 71: Michael Steber
Prozeßoptimierter Betrieb flexibler Schraubstationen in der automatisierten Montage
168 Seiten, 78 Bilder, 3 Tabellen. 1997.
ISBN 3-87525-096-6

Band 72: Markus Pfestorf
Funktionale 3D–Oberflächenkenngrößen in der Umformtechnik
162 Seiten, 84 Bilder, 15 Tabellen. 1997.
ISBN 3-87525-097-4

Band 73: Volker Franke
Integrierte Planung und Konstruktion von Werkzeugenfür die Biegebearbeitung
143 Seiten, 81 Bilder. 1998.
ISBN 3-87525-098-2

Band 74: Herbert Scheller
Automatisierte Demontagesysteme und recyclinggerechte Produktgestaltung elektronischer Baugruppen
184 Seiten, 104 Bilder, 17 Tabellen. 1998.
ISBN 3-87525-099-0

Band 75: Arthur Meßner
Kaltmassivumformung metallischer Kleinstteile – Werkstoffverhalten, Wirkflächenreibung, Prozeßauslegung
164 Seiten, 92 Bilder, 14 Tabellen. 1998.
ISBN 3-87525-100-8

Band 76: Mathias Glasmacher
Prozeß– und Systemtechnik zum Laserstrahl–Mikroschweißen
184 Seiten, 104 Bilder, 12 Tabellen. 1998.
ISBN 3-87525-101-6

Band 77: Michael Schwind
Zerstörungsfreie Ermittlung mechanischer Eigenschaften von Feinblechen mit dem Wirbelstromverfahren
124 Seiten, 68 Bilder, 8 Tabellen. 1998.
ISBN 3-87525-102-4

Band 78: Manfred Gerhard
Qualitätssteigerung in der Elektronikproduktion durch Optimierung der Prozeßführung beim Löten komplexer Baugruppen
179 Seiten, 113 Bilder, 7 Tabellen. 1998.
ISBN 3-87525-103-2

Band 79: Elke Rauh
Methodische Einbindung der Simulation in die betrieblichen Planungs– und Entscheidungsabläufe
192 Seiten, 114 Bilder, 4 Tabellen. 1998.
ISBN 3-87525-104-0

Band 80: Sorin Niederkorn
Meßeinrichtung zur Untersuchung der Wirkflächenreibung bei umformtechnischen Prozessen
99 Seiten, 46 Bilder, 6 Tabellen. 1998.
ISBN 3-87525-105-9

Band 81: Stefan Schuberth
Regelung der Fokuslage beim Schweißen mit CO_2–Hochleistungslasern unter Einsatz von adaptiven Optiken
140 Seiten, 64 Bilder, 3 Tabellen. 1998.
ISBN 3-87525-106-7

Band 82: Armando Walter Colombo
Development and Implementation of Hierarchical Control Structures of Flexible Production Systems Using High Level Petri Nets
216 Seiten, 86 Bilder. 1998.
ISBN 3-87525-109-1

Band 83: Otto Meedt
Effizienzsteigerung bei Demontage und Recycling durch flexible Demontagetechnologien und optimierte Produktgestaltung
186 Seiten, 103 Bilder. 1998.
ISBN 3-87525-108-3

Band 84: Knuth Götz
Modelle und effiziente Modellbildung
zur Qualitätssicherung in der Elektronikproduktion
212 Seiten, 129 Bilder, 24 Tabellen. 1998.
ISBN 3-87525-112-1

Band 85: Ralf Luchs
Einsatzmöglichkeiten leitender Klebstoffe
zur zuverlässigen Kontaktierung elektronischer Bauelemente in der SMT
176 Seiten, 126 Bilder, 30 Tabellen. 1998.
ISBN 3-87525-113-7

Band 86: Frank Pöhlau
Entscheidungsgrundlagen zur Einführung
räumlicher spritzgegossener Schaltungsträger (3–D MID)
144 Seiten, 99 Bilder. 1999.
ISBN 3-87525-114-8

Band 87: Roland T. A. Kals
Fundamentals on the miniaturization of sheet metal working processes
128 Seiten, 58 Bilder, 11 Tabellen. 1999.
ISBN 3-87525-115-6

Band 88: Gerhard Luhn
Implizites Wissen und technisches Handeln
am Beispiel der Elektronikproduktion.
252 Seiten, 61 Bilder, 1 Tabelle. 1999.
ISBN 3-87525-116-4

Band 89: Axel Sprenger
Adaptives Streckbiegen von Aluminium–Strangpreßprofilen
114 Seiten, 63 Bilder, 4 Tabellen. 1999.
ISBN 3-87525-117-2

Band 90: Hans–Jörg Pucher
Untersuchungen zur Prozeßfolge Umformen, Bestücken
und Laserstrahllöten von Mikrokontakten
158 Seiten, 69 Bilder, 9 Tabellen. 1999.
ISBN 3-87525-119-9

Band 91: Horst Arnet
Profilbiegen mit kinematischer Gestalterzeugung
128 Seiten, 67 Bilder, 7 Tabellen. 1999.
ISBN 3-87525-120-2

Band 92: Doris Schubart
Prozeßmodellierung und Technologieentwicklung
beim Abtragen mit CO_2-Laserstrahlung
133 Seiten, 57 Bilder, 13 Tabellen. 1999.
ISBN 3-87525-122-9

Band 93: Adrianus L. P. Coremans
Laserstrahlsintern von Metallpulver – Prozeßmodellierung,
Systemtechnik, Eigenschaften laserstrahlgesinterter Metallkörper
184 Seiten, 108 Bilder, 12 Tabellen. 1999.
ISBN 3-87525-124-5

Band 94: Hans-Martin Biehler
Optimierungskonzepte für Qualitätsdatenverarbeitung
und Informationsbereitstellung in der Elektronikfertigung
194 Seiten, 105 Bilder. 1999.
ISBN 3-87525-126-1

Band 95: Wolfgang Becker
Oberflächenausbildung und tribologische Eigenschaften
excimerlaserstrahlbearbeiteter Hochleistungskeramiken
175 Seiten, 71 Bilder, 3 Tabellen. 1999.
ISBN 3-87525-127-x

Band 96: Philipp Hein
Innenhochdruck-Umformen von Blechpaaren:
Modellierung, Prozeßauslegung und Prozeßführung
129 Seiten, 57 Bilder, 7 Tabellen. 1999.
ISBN 3-87525-128-8

Band 97: Gunter Beitinger
Herstellungs- und Prüfverfahren
für thermoplastische Schaltungsträger
169 Seiten, 92 Bilder, 20 Tabellen. 1999.
ISBN 3-87525-129-6

Band 98: Jürgen Knoblach
Beitrag zur rechnerunterstützten verursachungsgerechten Angebotskalkulation von Blechteilen mit Hilfe wissensbasierter Methoden
155 Seiten, 53 Bilder, 26 Tabellen. 1999.
ISBN 3-87525-130-X

Band 99: Frank Breitenbach
Bildverarbeitungssystem zur Erfassung der Anschlußgeometrie elektronischer SMT-Bauelemente
147 Seiten, 92 Bilder, 12 Tabellen. 2000.
ISBN 3-87525-131-8

Band 100: Bernd Falk
Simulationsbasierte Lebensdauervorhersage
für Werkzeuge der Kaltmassivumformung
134 Seiten, 44 Bilder, 15 Tabellen. 2000.
ISBN 3-87525-136-9

Band 101: Wolfgang Schlögl
Integriertes Simulationsdaten-Management
für Maschinenentwicklung und Anlagenplanung
169 Seiten, 101 Bilder, 20 Tabellen. 2000.
ISBN 3-87525-137-7

Band 102: Christian Hinsel
Ermüdungsbruchversagen hartstoffbeschichteter Werkzeugstähle in der Kaltmassivumformung
130 Seiten, 80 Bilder, 14 Tabellen. 2000.
ISBN 3-87525-138-5

Band 103: Stefan Bobbert
Simulationsgestützte Prozessauslegung
für das Innenhochdruck-Umformen von Blechpaaren
123 Seiten, 77 Bilder. 2000.
ISBN 3-87525-145-8

Band 104: Harald Rottbauer
Modulares Planungswerkzeug
zum Produktionsmanagementin der Elektronikproduktion
166 Seiten, 106 Bilder. 2001.
ISBN 3-87525-139-3

Band 105: Thomas Hennige
Flexible Formgebung von Blechen durch Laserstrahlumformen
119 Seiten, 50 Bilder. 2001.
ISBN 3-87525-140-7

Band 106: Thomas Menzel
Wissensbasierte Methoden für die rechnergestützte Charakterisierung und Bewertung innovativer Fertigungsprozesse
152 Seiten, 71 Bilder. 2001.
ISBN 3-87525-142-3

Band 107: Thomas Stöckel
Kommunikationstechnische Integration der Prozeßebene in Produktionssysteme durch Middleware-Frameworks
147 Seiten, 65 Bilder, 5 Tabellen. 2001.
ISBN 3-87525-143-1

Band 108: Frank Pitter
Verfügbarkeitssteigerung von Werkzeugmaschinen durch Einsatz mechatronischer Sensorlösungen
158 Seiten, 131 Bilder, 8 Tabellen. 2001.
ISBN 3-87525-144-X

Band 109: Markus Korneli
Integration lokaler CAP-Systeme in einen globalen Fertigungsdatenverbund
121 Seiten, 53 Bilder, 11 Tabellen. 2001.
ISBN 3-87525-146-6

Band 110: Burkhard Müller
Laserstrahljustieren mit Excimer-Lasern – Prozeßparameter und Modelle zur Aktorkonstruktion
128 Seiten, 36 Bilder, 9 Tabellen. 2001
ISBN 3-87525-159-8

Band 111: Jürgen Göhringer
Integrierte Telediagnose via Internet zum effizienten Service von Produktionssystemen
178 Seiten, 98 Bilder, 5 Tabellen. 2001.
ISBN 3-87525-147-4

Band 112: Robert Feuerstein
Qualitäts- und kosteneffiziente Integration neuer Bauelementetechnologien in die Flachbaugruppenfertigung
161 Seiten, 99 Bilder, 10 Tabellen. 2001.
ISBN 3-87525-151-2

Band 113: Marcus Reichenberger
Eigenschaften und Einsatzmöglichkeiten alternativer Elektroniklote in der Oberflächenmontage (SMT)
165 Seiten, 97 Bilder, 18 Tabellen. 2001.
ISBN 3-87525-152-0

Band 114: Alexander Huber
Justieren vormontierter Systeme mit dem Nd:YAG-Laser unter Einsatz von Aktoren
122 Seiten, 58 Bilder, 5 Tabellen. 2001.
ISBN 3-87525-153-9

Band 115: Sami Krimi
Analyse und Optimierung von Montagesystemen in der Elektronikproduktion
155 Seiten, 88 Bilder, 3 Tabellen. 2001.
ISBN 3-87525-157-1

Band 116: Marion Merklein
Laserstrahlumformen von Aluminiumwerkstoffen - Beeinflussung der Mikrostruktur und der mechanischen Eigenschaften
122 Seiten, 65 Bilder, 15 Tabellen. 2001.
ISBN 3-87525-156-3

Band 117: Thomas Collisi
Ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten
181 Seiten, 105 Bilder, 7 Tabellen. 2002.
ISBN 3-87525-164-4

Band 118: Markus Koch
Rationalisierung und ergonomische Optimierung im Innenausbau durch den Einsatz moderner Automatisierungstechnik
176 Seiten, 98 Bilder, 9 Tabellen. 2002.
ISBN 3-87525-165-2

Band 119: Michael Schmidt
Prozeßregelung für das Laserstrahl-Punktschweißen in der Elektronikproduktion
152 Seiten, 71 Bilder, 3 Tabellen. 2002.
ISBN 3-87525-166-0

Band 120: Nicolas Tiesler
Grundlegende Untersuchungen zum Fließpressen metallischer Kleinstteile
In Druck

Band 121: Lars Pursche
Methoden zur rechnergestützten Programmierung für die 3D- Lasermikrobearbeitung
in Druck

Band 122: Jan-Oliver Brassel
Prozeßkontrolle beim Laserstrahl-Mikroschweißen
in Druck

Band 123: Mark Geisel
Prozeßkontrolle und –steuerung beim Laserstrahlschweißen
in Druck

Band 124: Gerd Eßer
Laserstrahlunterstützte Erzeugung metallischer Leiterstrukturen auf Thermoplastsubstraten für die MID-Technik
148 Seiten, 60 Bilder, 6 Tabellen. 2002.
ISBN 3-87525-171-7

Band 125: Marc Fleckenstein
Qualität laserstrahl-gefügter Mikroverbindungen elektronischer Kontakte
159 Seiten, 77 Bilder, 7 Tabellen. 2002.
ISBN 3-87525-170-9

Band 126: Stefan Kaufmann
Grundlegende Untersuchungen zum Nd:YAG- Laserstrahlfügen von Silizium für Komponenten der Optoelektronik
159 Seiten, 100 Bilder, 6 Tabellen. 2002.
ISBN 3-87525-172-5

Band 127: Thomas Fröhlich
Simultanes Löten von Anschlußkontakten elektronischer Bauelemente mit Diodenlaserstrahlung
in Druck

Band 128: Achim Hofmann
Erweiterung der Formgebungsgrenzen beim Umformen von Aluminiumwerkstoffen durch den Einsatz prozessangepasster Platinen
in Druck

Band 129: Ingo Kriebitzsch
3-D MID Technologie in der Automobilelektronik
129 Seiten, 102 Bilder, 10 Tabellen. 2002.
ISBN 3-87525-169-5

Band 130: Thomas Pohl
Fertigungsqualität und Umformbarkeit laserstrahlgeschweißter Formplatinen aus Aluminiumlegierungen
in Druck

Band 131: Matthias Wenk
Entwicklung eines konfigurierbaren Steuerungssystems für die flexible Sensorführung von Industierobotern
167 Seiten, 85 Bilder, 1 Tabelle. 2002.
ISBN 3-87525-174-1

Band 132: Nicolas Tiesler
Grundlegende Untersuchungen zum Fließpressen metallischer Kleinstteile
126 Seiten, 78 Bilder, 12 Tabellen. 2002.
ISBN 3-87525-175-X