Technologieorientiertes Programmier- und Steuerungssystem für Industrieroboter

von Diplom-Ingenieur
Thomas Friedrich
aus Berlin

von der Fakultät V – Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr.-Ing. G. Seliger

Gutachter: Prof. Dr. h. c. Dr.-Ing. E. Uhlmann

Gutachter: Prof. Dr.-Ing. A. Verl

Tag der wissenschaftlichen Aussprache: 19. Juli 2010

Berlin 2010
D83

Vorwort des Herausgebers

Industrieroboter stellen heute eine Standard-Automatisierungskomponente in der Automobilindustrie dar, sie werden zur Kostensenkung durch Rationalisierung eingesetzt. Aber nicht nur große Konzerne, sondern auch kleine und mittlere Unternehmen (KMU) spüren den Kostendruck, der durch die industrielle Produktion und den internationalen Wettbewerb entsteht. KMU benötigen flexible und leicht bedienbare Industrieroboter, anderenfalls ist deren Einsatz oft unrentabel. Diese Forderungen bedingen die Entwicklung von innovativen Programmierverfahren, welche den spezifischen Produktionsbedingungen von kleinen und mittleren Unternehmen Rechnung tragen.

Das Ziel dieser am Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) der Technischen Universität Berlin angefertigten Arbeit ist daher die Entwicklung eines Programmier- und Steuerungssystems, das einen effizienten Einsatz von Industrierobotern bei kleinen Losgrößen ermöglicht, wobei zusätzlich eine einfache Anpassung der Steuerungsparameter gewährleistet wird. Darüber hinaus müssen Roboterprogramme flexibel an unterschiedliche Aufgabenstellungen angepasst werden können. Herr Friedrich entwickelte ein technologieorientiertes Programmier- und Steuerungssystem, das den Ansatz nutzt, bestehende Anwenderprogramme in kleinste generische Einheiten, die Elementaranweisungen, zu unterteilen. Die Kombination von Offline programmierten Elementaranweisungen und die Online-Verknüpfung mit den entsprechenden Steuerungsinformationen ergibt ein lauffähiges Anwenderprogramm, wobei dieses nur ein virtuelles Programm darstellt.

Das entwickelte System wurde prototypisch am Beispiel der Demontage realisiert, da die Demontage von Gebrauchtgütern durch eine enorme Vielfalt an Produktvarianten sowie -zuständen gekennzeichnet ist und somit höchste Flexibilitätsanforderungen an automatisierte Demontagesysteme gestellt werden. Hierzu wurde das technologieorientierte Programmier- und Steuerungssystem in das Pilot-Demontagesystem des IWF der TU Berlin implementiert.

Berlin im Oktober 2010 Eckart Uhlmann

Vorwort des Verfassers

Die vorliegende Arbeit entstand während meiner Tätigkeit als Wissenschaftlicher Mitarbeiter am Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) der Technischen Universität Berlin.

Mein besonderer Dank gilt Herrn Professor Dr. h. c. Dr.-Ing. Eckart Uhlmann, dem Leiter des Fachgebietes Werkzeugmaschinen und Fabrikbetrieb sowie des Fraunhofer-Instituts für Produktionsanlagen und Konstruktionstechnik, für die fachliche Unterstützung, die eingehende Durchsicht sowie die hilfreichen Anregungen bei der Erstellung dieser Arbeit. Herrn Professor Dr.-Ing. Alexander Verl, dem geschäftsführenden Direktor des Instituts für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) an der Universität Stuttgart sowie des Fraunhofer-Instituts für Produktionstechnik und Automatisierung, danke ich für das dieser Arbeit entgegengebrachte Interesse. Herrn Professor Dr.-Ing. Günther Seliger, dem Leiter des Fachgebietes Montagetechnik und Fabrikbetrieb, danke ich für die Übernahme des Vorsitzes im Promotionsausschuss.

Ich danke allen Professoren und Kollegen für ihr herausragendes Engagement bei Aufbau und Weiterentwicklung des DFG Sonderforschungsbereiches 281 „Demontagefabriken zur Rückgewinnung von Ressourcen in Produkt- und Materialkreisläufen", in dessen Rahmen diese Arbeit entstand.

Allen Freunden und Kollegen im Produktionstechnischen Zentrum Berlin danke ich für die langjährige, sehr gute Zusammenarbeit. Namentlich möchte ich mich sehr herzlich bedanken bei den Herren Jens König, Bernd Duchstein, Christoph König und Thomas Keil. Für ihre hervorragende Mitarbeit danke ich ferner allen studentischen Mitarbeitern und Diplomanden, insbesondere den Herren Gordon Bach, Thomas Mahnkopf und Jakob Wolf.

Nicht zuletzt danke ich meinen Eltern sowie meiner lieben Frau Tina für Liebe, Geborgenheit und Verständnis während der vielen Wochenenden und langen Abende, die für die Erstellung dieser Arbeit erforderlich waren.

Berlin im Oktober 2010 Thomas Friedrich

Abstract

Für die Programmierung von Industrierobotern bei kleinen und mittleren Unternehmen (KMU) müssen neue Programmierverfahren entwickelt werden, welche die spezifischen Anforderungen von KMU wie die Fertigung kleiner Losgrößen beachten. Ein technologieorientiertes Programmier- und Steuerungssystem wird in dieser Doktorarbeit vorgestellt, welches die Programmierung basierend auf dem Wissen eines Facharbeiters ermöglicht. Das Programmier- und Steuerungssystem ist ein hybrides Programmierverfahren, das durch die Kombination von Offline programmierten Roboteranweisungen und der Online-Verknüpfung von Steuerungsinformationen ein Anwenderprogramm darstellt. Dieses System kann durch einen Facharbeiter ohne ausgewiesene Roboterprogrammierkenntnisse bedient werden. Die exemplarische Realisierung wurde für ein Pilot-Demontagesystem durchgeführt, das am Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) der Technischen Universität Berlin aufgebaut wurde.

Abstract

New methods for the programming of industrial robots have to be developed because the costs for programming of small lot sizes are a key factor of the economic efficiency of small and medium sized enterprises (SME). In the doctoral thesis a technology orientated programming and control system for programming of industrial robots is presented, which is programmed based on technological knowledge of a workman. This programming and control system is a hybrid programming system, which is a combination of the off-line programmed robot instruction and the on-line combination with the control information. This system can be used by an end-user, who is not explicitly qualified for programming of industrial robots. The exemplary implementation is realized at a pilot disassembly system, which was built up in the Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) of the Technische Universität Berlin.

Inhaltsverzeichnis ..... Seite

0 Formelzeichen und Abkürzungen IV

1 Einleitung 1

2 Stand der Erkenntnisse 3

2.1 Begriffliche Grundlagen 3

2.2 Programmierung von Industrierobotern 6

2.2.1 Onlineprogrammierverfahren 7

2.2.2 Offlineprogrammierverfahren 10

2.2.3 Hybride Programmierverfahren 24

2.3 Steuerung von Industrierobotern 26

2.4 Demontage 30

3 Handlungsbedarf 33

3.1 Herausforderungen für kleine und mittlere Unternehmen 33

3.2 Defizite aktueller Programmierverfahren hinsichtlich des Einsatzes bei kleinen und mittleren Unternehmen 35

3.3 Anforderungen an technologieorientierte Programmier- und Steuerungssysteme 39

4 Zielsetzung und Vorgehensweise 42

5 Konzeption technologieorientierter Programmier- und Steuerungssysteme 46

5.1 Struktur des Gesamtsystems 46

5.2 Systemkomponenten 48

5.2.1 Allgemeines 48

5.2.2 Robotersteuerung 49

5.2.3 Prozesssteuerung 54

5.2.4 Datenbanksystem 55

5.2.5 Programmierleitstand 57

5.2.6 Schnittstellen 61

5.2.6.1 Interne Schnittstellen 62

5.2.6.2 Externe Benutzerschnittstelle 63

5.3 Inbetriebnahme des technologieorientierten Programmier- und Steuerungssystems 65

6 Prototypische Realisierung des Konzepts am Beispiel eines Pilot-Demontagesystems 68

6.1 Pilot-Demontagesystem 68
6.2 Adaption existierender Komponenten 73
6.2.1 Robotersteuerung 73
6.2.2 Speicherprogrammierbare Steuerung 79
6.3 Realisierung zusätzlicher Komponenten 84
6.3.1 Datenbanksystem 84
6.3.2 Programmierleitstand 91
6.4 Verwendete Schnittstellen 98
6.5 Implementierung des technologieorientierten Programmier- und Steuerungssystems 99

7 Erprobung und Bewertung des technologieorientierten Programmier- und Steuerungssystems 102

7.1 Erprobung des Systems 102
7.2 Vergleich und Bewertung der Demontageprozesse vor und nach der Implementierung des Programmier- und Steuerungssystems 104
7.2.1 Produktwechsel 106
7.2.2 Fehler während des Demontageablaufs 109
7.3 Wirtschaftlichkeitsabschätzung 111
7.4 Konzeptergänzende Lösungen 121
7.4.1 Hilfesystem 121
7.4.2 Unternehmensnetzwerk 122
7.4.3 Softwareerweiterungsmodule für einen effizienten Einsatz 124

8 Zusammenfassung und Ausblick 127

9 Literaturverzeichnis 130

10 Anhang 141

A 1 - Übersicht typischer Elementaranweisungen 141
A 2 - Parameter der Elementaranweisungen 143

A 3 - Darstellung des Prozesses „Demontage Seitenwand mit Zerschleifwerkzeug“ für KUKA II 145
A 4 - Definition des Datenbausteins für die Zwischenspeicherung des Demontageplans eines Industrieroboters 148
A 5 - Ablaufdiagramm „TOPS-Steuermodul“ für die Komponenten „KUKA II“ und „SYSTEM“ 149
A 6 - Darstellung der Datenbankstruktur 150

0 Formelzeichen und Abkürzungen

Formelzeichen

Zeichen	Einheit	Erläuterung
BK_{MH}	€	Bruttomaschinenstundensatz
E_D	€/h	Erlöse des Demontagesystems pro Stunde
K_A	€/h	Abschreibungskosten pro Stunde
K_E	€/h	Energiekosten pro Stunde
K_I	€/h	Instandhaltungskosten pro Stunde
K_R	€/h	Raumkosten pro Stunde
K_{ST}	€/h	Kosten für Stillstands- und Testzeiten pro Stunde
K_W	€/h	Werkzeugkosten pro Stunde
K_Z	€/h	Zinskosten pro Stunde
L_{PR}	€/h	Lohnkosten pro Stunde
NK_{Mh}	€	Nettomaschinenstundensatz
T_{LA}	h	jährliche Laufzeit der Einrichtung
Z_{iPR}	h	Programmierzeiten

Abkürzungen

Zeichen	Erläuterung
AB	Arbeitsstationsbasis
AS	Arbeitsstation
AWT	Abstract Window Toolkit
CAD	Computer Aided Design
CAR	Computer Aided Robotics
DB	Datenbank

DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
ELA	Elementaranweisung
EU	Europäische Union
FC	Function Code (Funktion)
FB	Function Block (Funktionsbaustein)
GPL	General Public License (Allgemeine Öffentliche Lizenz)
GUI	Graphical User Interface (Grafische Benutzeroberfläche)
IR	Industrieroboter
IRDATA	Industrial Robot Data
JDBC	Java Database Connectivity
KMU	Kleine und mittlere Unternehmen
MMS	Mensch-Maschine-Schnittstelle
OB	Objektbasis
OCI	Oracle Call Interface
ODBC	Open Database Connectivity
OPC	Openness, Productivity, Collaboration (vorherig: OLE for Process Control)
PDS	Pilot-Demontagesystem
PHP	Hypertext Preprocessor
PLS	Programmierleitstand
Profibus	Process Field Bus
RC	Robot Control (Robotersteuerung)
SME	Small and Medium-sized Enterprises (Kleine und mittlere Unternehmen)
SPS	Speicherprogrammierbare Steuerung

SQL	Structured Query Language (Strukturierte Abfragesprache)
TB	Transportmittelbasis
TCP/IP	Transmission Control Protocol/Internet Protocol
TOPS	Technologieorientiertes Programmier- und Steuerungssystem
URL	Uniform Resource Locato (Einheitlicher Quellenanzeiger)
WB	Weltbasis
WM	Waschmaschine
XAMPP	Akronym: „X“ für das Betriebssystem, „A“ für den Webserver „Apache“, „M“ für das Datenbankverwaltungssystem „MySQL“, „P“ für die Programmiersprache „Perl“, „P“ für die Programmiersprache „PHP“
XML	Extensible Markup Language

1 Einleitung

Industrieroboter stellen heute eine Standard-Automatisierungskomponente in der Automobilindustrie dar, sie werden zur Kostensenkung durch Rationalisierung eingesetzt. Aber nicht nur große Konzerne, sondern auch kleine und mittlere Unternehmen (KMU) spüren den Kostendruck, der durch die industrielle Produktion und den internationalen Wettbewerb entsteht [Mey-06]. Industrieroboter werden etwa seit Anfang der siebziger Jahre in größeren Stückzahlen eingesetzt, in Deutschland sind derzeit mehr als 100.000 Industrieroboter im Einsatz, dadurch zählt es zu den roboterfreundlichsten Nationen der Welt, vgl. **Bild 1.1** [IFR-08]. Pro 10.000 Beschäftigte in der verarbeitenden Industrie verrichten hierzulande 234 Roboter ihre Tätigkeit, wobei Industrieroboter hauptsächlich von größeren Unternehmen für die Fertigung in Großserien eingesetzt werden [Arm-06b, Möb-96].

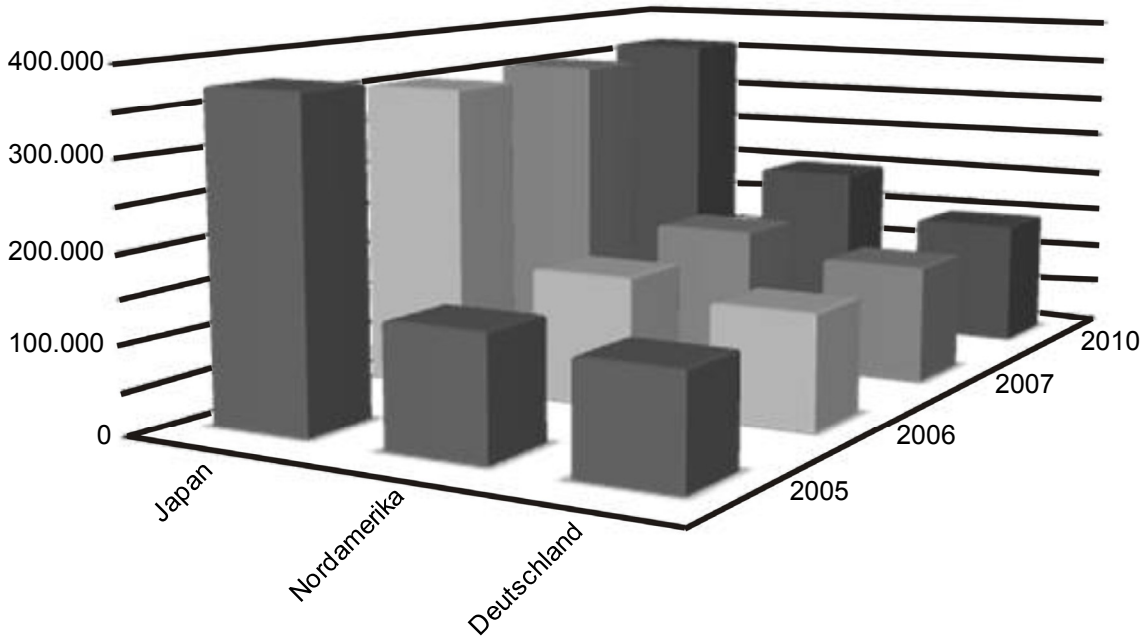


Bild 1.1: Internationaler Bestand von Industrierobotern – Installation im Einsatz 2005, 2006; Prognose 2007, 2010 [IFR-08]

Dies liegt zum Teil daran, dass für die Produktionsbedingungen von kleinen und mittleren Unternehmen (KMU) keine geeigneten technischen und wirtschaftlichen Lösungen für den Einsatz von Industrierobotern existieren [Arm-06b, Lie-01]. Diese Betriebe setzen Roboter deshalb nicht ein, da es für ihre spezifischen Anwendungs-

bedingungen ihrer Ansicht nach noch keine geeigneten technischen Roboterentwicklungen existieren oder sie erwarten, dass ein Einsatz der verfügbaren Robotertechnik unter ihren Produktionsbedingungen derzeit nicht wirtschaftlich wäre [Arm-06a]. KMU benötigen flexible und leicht bedienbare Roboter, ansonsten ist deren Einsatz oft unrentabel [Pro-07]. Der Hauptgrund hierfür sind die hohen Investitions- und Betriebskosten existierender Lösungen, die lange Amortisierungsdauer aufweisen und sich dadurch bei relativ kleinen Stückzahlen sowie geringen Auslastungen der Geräte nicht tragen. Die Roboterprogrammierung verursacht einen wesentlichen Bestandteil der Betriebskosten eines Roboters und kann zu Stillstandzeiten führen [Neu-97]. Darüber hinaus erfordern Installation und Bedienung von Industrierobotern Fachwissen, dessen Aneignung für die Belegschaft eines KMU nicht immer realisierbar ist [Kam-00, Ram-07, Som-07, Wec-04]. Weiterhin gewinnen im Hinblick auf den zunehmenden Einsatz von Robotern in der Kleinserien- und Einzelfertigung die Kosten zur Erstellung der Roboterprogramme eine immer größere Bedeutung [Val-95].

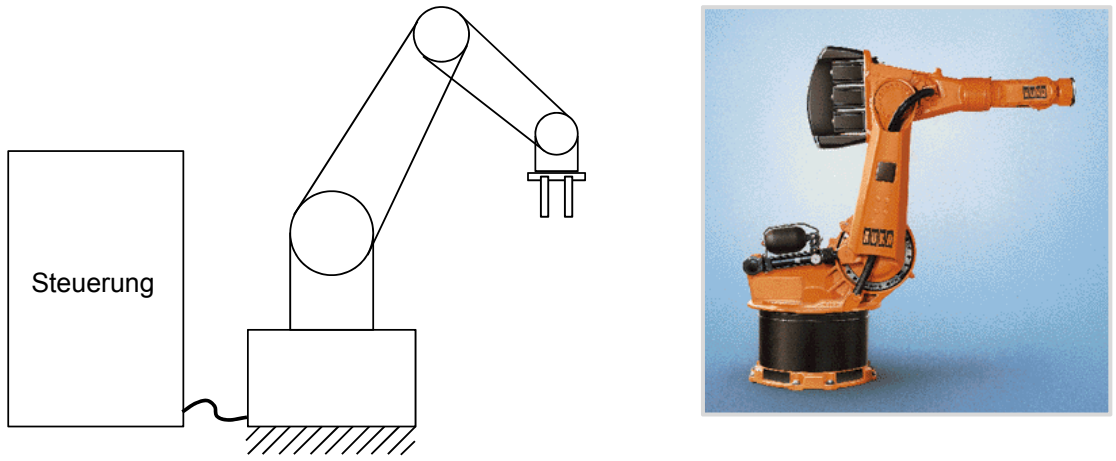
Die Bedeutung der KMU für die wirtschaftliche Entwicklung, insbesondere in den europäischen Ländern, ist erheblich. Es gibt in der Europäische Union (EU) nach offiziellen Angaben etwa 23 Mio. KMU, das sind 99 % aller Unternehmen. KMU stellen 100 Mio. Arbeitsplätze, was mehr als drei Viertel aller Stellen der EU entspricht [Ent-09].

Da die Demontage von Gebrauchtgütern durch eine enorme Vielfalt an Produktvarianten sowie -zuständen gekennzeichnet ist und sich der Zustand des zu demontierenden Produkts während seines Lebenszyklus verändern kann, ist die industrielle Demontage durch die Losgröße Eins charakterisiert [Gra-98, Här-05, Kei-04, Uhl-07]. Die Programmierkosten von Industrierobotern sind bei kleinen Stückzahlen für eine Wirtschaftlichkeitsbetrachtung entscheidend, daher wird das entwickelte Programmier- und Steuerungssystem am Beispiel der Demontage vorgestellt.

2 Stand der Erkenntnisse

2.1 Begriffliche Grundlagen

Der Begriff *Roboter* ist abgeleitet aus dem tschechischen Wort „robota“ und wurde 1921 bei der Uraufführung des Theaterstückes „Rossum,s Universal Robots“ von Karel Capek als Bezeichnung für einen Leibeigenen, Zwangsarbeiter oder unterwürfigen Bediensteten gebraucht [Wol-04]. *Industrieroboter* (IR) sind Arbeitsmaschinen, die, zur selbsttätigen Handhabung von Objekten mit zweckdienlichen Werkzeugen ausgerüstet, in mehreren Bewegungsachsen hinsichtlich Orientierung, Position sowie Arbeitsablauf programmierbar sind. Da sich der mechanische Aufbau von Handhabungsgeräten durch kinematische Ketten darstellen lässt, ist die Anzahl der Freiheitsgrade eines Handhabungsgeräts gleich der Anzahl der unabhängig zu bewegenden Glieder der kinematischen Kette, wenn jedes Gelenk nur einen Freiheitsgrad hat [Spu-79]. Die DIN EN ISO 8373 definiert Industrieroboter als Mehrzweckmanipulatoren mit Greifer oder Werkzeugen (Endeffektoren), die automatisch geführt sowie mit drei oder mehr frei programmierbaren Bewegungsachsen ausgerüstet sind und die entweder ortsfest oder mobil in industriellen Anwendungen eingesetzt werden [ISO 8373], **Bild 2.1**.



a) Prinzipieller Aufbau eines ortsfesten Industrieroboters

b) Knickarmroboter der Firma KUKA Roboter GmbH [Kuk-08]

Bild 2.1: Industrieroboter

Gemäß der Britischen Roboter-Gemeinschaft BARA (British Association for Robotics and Automation) werden Roboter wie folgt definiert:

“An industrial robot is a re-programmable device designed to both manipulate and transport parts, tools, or specialised manufacturing implements through variable programmed motions for the performance of specific manufacturing tasks” [Bar-08]. (Ein Industrieroboter ist ein umprogrammierbares Gerät, das entwickelt wurde, Teile, Werkzeuge oder spezielle Fertigungshilfseinrichtungen mit Hilfe von veränderlichen programmierbaren Bewegungen für die Ausführung von spezifischen Fertigungsaufgaben zu handhaben und transportieren.)

Unter einem *Programmierverfahren* ist das planmäßige Vorgehen zur Erzeugung von Anwenderprogrammen zu verstehen, vgl. **Bild 2.2**. Nach VDI-Richtlinie 2863 ist dabei ein Anwenderprogramm eine Sequenz von Anweisungen mit dem Zweck, eine vorgegebene Fertigungsaufgabe zu erfüllen [VDI 2863]. Programmiersysteme ermöglichen die Erstellung von Anwenderprogrammen und stellen hierzu entsprechende Programmierhilfen zur Verfügung [Spu-04]. Bei der Roboterprogrammierung werden die einer Roboteraufgabe zugehörigen Abläufe zunächst definiert und in einem bestimmten Datenformat als Roboterprogramm abgelegt. Diese Abläufe enthalten sowohl Bewegungsdaten des Roboters als auch Kommunikationszyklen, Berechnungen und Schleifen [Got-01].

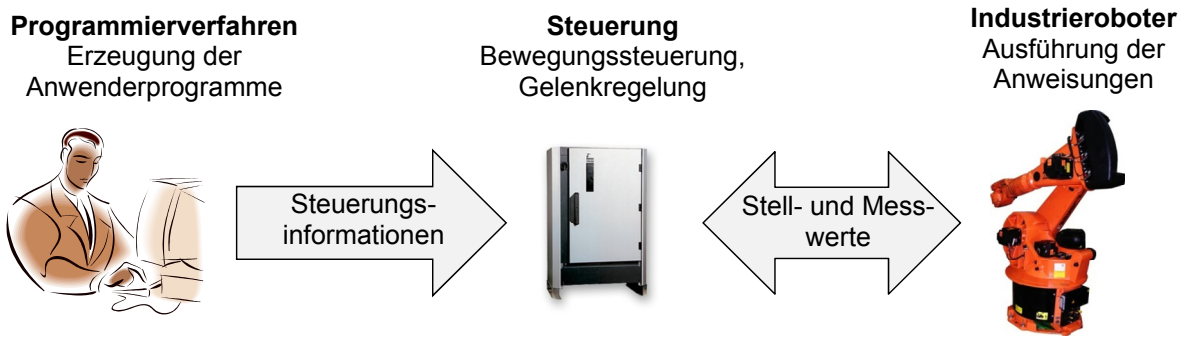


Bild 2.2: Programmierung von Industrierobotern

Die DIN EN ISO 8373 definiert die *Robotersteuerung* als einen Satz von logischen Steuer- und Leistungsfunktionen, die eine Überwachung der mechanischen Struktur des Roboters und die Kommunikation mit der Umgebung ermöglicht [ISO 8373]. Hierbei umfasst die Steuerung folgende Funktionalitäten:

- die Entgegennahme und Abarbeitung von einzelnen Roboterbefehlen oder -programmen,
- die Steuerung und Überwachung von Bewegungs- bzw. Handhabungssequenzen und Fahraufträgen,

- die Synchronisation und Anpassung des Manipulators an den Handhabungsprozess sowie
- die Vermeidung bzw. Auflösung von Konfliktsituationen [Hau-06].

Eine weitere Definition für eine Steuerung wird vom Verband für Arbeitsstudien und Betriebsorganisation e. V. (REFA) genannt. Hiernach ist eine Steuerung das Veranlassen, Überwachen und Sichern der Aufgabendurchführung hinsichtlich Menge, Termin, Qualität, Kosten und Arbeitsbedingungen. Das Veranlassen der Aufgabendurchführung ist das mengen- und terminorientierte Vorbereiten sowie Auslösen aller notwendigen Maßnahmen. Das Überwachen der Aufgabendurchführung ist das kontinuierliche oder in zeitlichen Abständen erfolgende Feststellen der Istdaten und das Ermitteln der Abweichungen der Ist- von den Solldaten. Sichern der Aufgabendurchführung ist das Veranlassen und Durchführen von Maßnahmen zum Vermeiden oder Vermindern von Abweichungen zwischen Ist- und Solldaten [Ref-93]. Darüber hinaus wird Steuerung durch die DIN 19226 definiert als Vorgang in einem System, bei dem eine oder mehrere Größen als Eingangsgrößen die Ausgangsgrößen aufgrund der dem System eigentümlichen Gesetzmäßigkeiten beeinflussen [DIN 19226-1]. Weiterhin unterteilt die DIN 19226 Steuerungen in

- synchrone und
- asynchrone Steuerungen,
- Verknüpfungssteuerungen sowie
- Ablaufsteuerungen [DIN 19226-5].

Eine synchrone Steuerung ist eine Steuerung, bei der die Signalverarbeitung synchron zu einem Taktsignal erfolgt, wohingegen eine asynchrone Steuerung ohne Taktsignal arbeitet. Eine Verknüpfungssteuerung ordnet den Zuständen der Eingangssignale durch boolesche Verknüpfung definierte Zustände der Ausgangssignale zu. Eine Ablaufsteuerung ist beschrieben als Steuerung mit zwangsläufig schrittweise erfolgendem Ablauf, bei der das Weiterschalten von einem Schritt zum nächsten zeitabhängig oder prozessabhängig erfolgt. Die Schrittfolge kann in besonderer Weise programmiert sein, z. B. mit Sprüngen, Schleifen, Verzweigungen [DIN 19226-5]. Der in dieser Arbeit verwendete Begriff „Steuerung“ im Kontext des technologieorientierten Programmier- und Steuerungssystems entspricht dem Begriff der Ablaufsteuerung.

Kleine und mittlere Unternehmen (KMU) sind Unternehmen mit 10 bis 249 Mitarbeitern und einem Jahresumsatz bis 50 Mio. Euro oder einer Jahresbilanzsumme bis 43 Mio. Euro. Hierbei werden Unternehmen unabhängig von ihrer Rechtsform als eine Einheit spezifiziert, die eine wirtschaftliche Tätigkeit ausübt [Emp-03].

Demontage ist die Gesamtheit aller geplanten Vorgänge, die der Vereinzelung von Mehrkörpersystemen zu Baugruppen, Bauteilen und/oder formlosem Stoff durch Trennen dienen. Eine Gruppe bzw. Baugruppe ist dabei ein aus zwei oder mehr Teilen und/oder Gruppen niedrigerer Ordnung bestehender Gegenstand [Hen-96, Här-05]. Ein Einzelteil bzw. Bauteil ist ein Gegenstand, der nicht zerstörungsfrei zerlegt werden kann [DIN199-1].

2.2 Programmierung von Industrierobotern

Bei der Roboterprogrammierung existieren verschiedene Programmierverfahren, die sich jedoch alle in die Unterkategorien der Online- und Offlineverfahren oder Hybridverfahren unterteilen lassen, siehe **Bild 2.3**. Die Programmierverfahren unterscheiden sich hinsichtlich der verwendeten Medien und der die Roboterbewegungen beschreibenden Elemente, wie Raumpunkte, Bahnkurven, Zustände und Arbeitszellenobjekten [Hec-95].

Hierbei versteht man unter Onlineverfahren Programmierverfahren, die mit direkter Hilfe des Roboters und seiner Steuerung durchgeführt werden [Wec-06]. Die Onlineverfahren sind geeignet, um den Industrieroboter schnell und einfach durch vorgegebene Operationen zu programmieren.

Dagegen sind die Offlineverfahren unabhängig von der eigentlichen Produktionseinrichtung anwendbar, wobei die Programmierung einer Aufgabe indirekt mit Hilfe einer problemorientierten Sprache erfolgt [Wec-06]. Darüber hinaus sind simulationsgestützte Systeme verfügbar, durch die sich die Bewegungsabläufe am Bildschirm generieren bzw. simulieren lassen. Hierbei sind umfangreiche informationstechnische Kenntnisse für die Simulation und Generierung der Roboterprogramme erforderlich. Die Hybridverfahren sind aus einer Kombination von Online- und Offlineverfahren hervorgegangen. Diese Verfahren sind dadurch gekennzeichnet, dass die Handhabungsaufgabe in der Arbeitsvorbereitung programmiert und dann vor Ort durch zusätzliche Festlegungen von Bewegungsabläufen ergänzt wird [Wec-06].

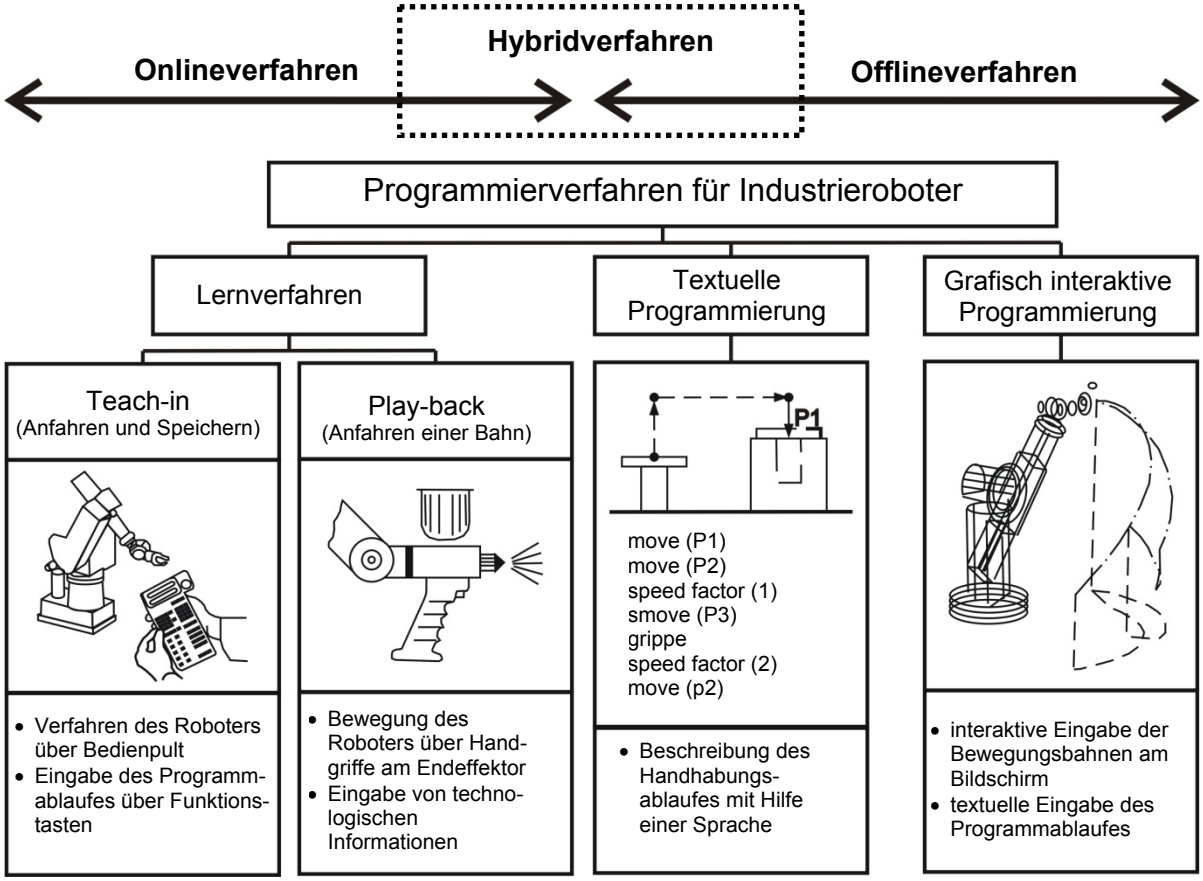


Bild 2.3: Programmierverfahren der Industrieroboter [in Anlehnung an Wec-06]

In den folgenden Abschnitten werden die derzeit relevanten Forschungsaktivitäten zur Entwicklung von Programmierverfahren vorgestellt, wobei diese in die Kategorien

- Online- und
- Offlineverfahren sowie
- Hybridverfahren

eingeordnet werden.

2.2.1 Onlineprogrammierverfahren

Bei den Onlineprogrammierverfahren erfolgt die Programmierung eines Roboters direkt am oder mit dem Roboter. Zu den Verfahren der Onlineprogrammierung zählen:

- Teach-in-Verfahren,
- Play-back-Verfahren und
- Werkstattorientierte Programmiersysteme.

Das *Teach-in- und das Play-back-Verfahren* zählen zu den Lernverfahren. Dabei führt der Programmierer den Roboter oder eine vergleichbare Konstruktion zwecks Koordinatenerfassung, während die Robotersteuerung die für die spätere Wiederholung der Bewegung erforderlichen geometrischen Daten speichert [Wec-06]. Mit W*erkstattorientierten Programmiersystemen* sind solche Systeme gemeint, die den Programmierer bei der Programmerstellung durch deutlich über dem gängigen Funktionsumfang von Robotersteuerungen hinausgehende Hilfsmittel unterstützen [Dre-97].

An der Rheinisch-Westfälischen Technischen Hochschule Aachen (RWTH Aachen) wurde in dem Projekt „*UP - Unscharfe Roboterprogrammierung“* unter der Leitung von WECK [Wec-03] ein neues Programmierkonzept für Industrieroboter erforscht. Gegenstand des Projekts „UP“ war es, ein neues Programmierkonzept zu entwickeln, mit dem eine geplante Sensoraufgabe automatisierter Handhabungssysteme nicht mehr explizit, sondern aufgabenorientiert und auf intuitive Weise in den Programmierprozess einbezogen wird. Die grundlegende Idee der unscharfen Programmierung besteht darin, Angaben, die im Vorfeld nicht exakt definiert werden können, im Programm ungenau durch Sollinformationen zu beschreiben [Wec-03]. Um die Programmierung von sensorgeführten Roboteraufgaben auf der Basis von ungenau definierten Angaben zu unterstützen, muss die Lageerkennung von Werkstücken schnell und konfigurierbar sein, siehe **Bild 2.4**.

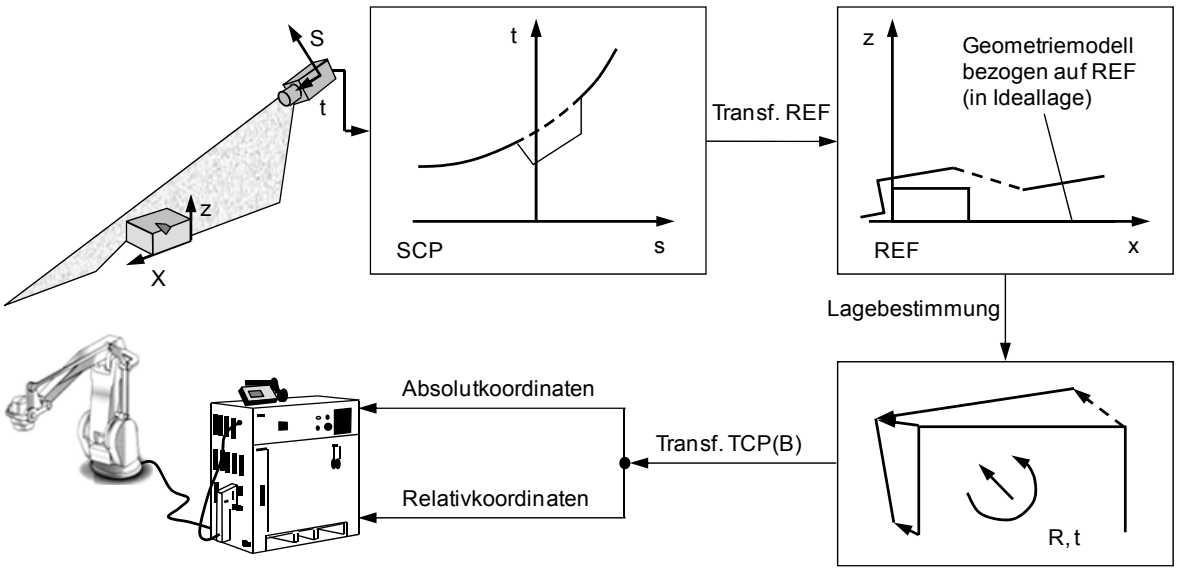


Bild 2.4: Ansatz für die schnelle, adaptive Lageerkennung [Wec-03]

Im Rahmen des Projekts wurde ein schneller Lageerkennungsalgorithmus entwickelt, der in kurzer Zeit die Lage von Bauteilen mit hoher Genauigkeit bestimmt [Wec-05]. Um Sensoren zu konfigurieren, wird das dreidimensionale CAD-Modell eines Bauteils als Referenzmodell verwendet.

An der Universität Karlsruhe wurde am Institut für Prozessrechentechnik und Robotik die prinzipielle Anwendbarkeit des Programmierens durch Vormachen (PdV) im Bereich der Robotik untersucht sowie ein prototypisches System realisiert [Mün-94]. Die Untersuchung wurde dabei zunächst auf Industrieroboter eingeschränkt, die in gut strukturierten Umgebungen eingesetzt werden und überwiegend einfache Handhabungsaufgaben (Pick-and-Place) ausführen. Eines der Hauptergebnisse dieser Studie ist eine Architektur für ein sogenanntes RPD-System (Robot Programming by Demonstration) [Mün-08]. Die Architektur besteht aus einem Systemkern, der möglichst unabhängig sein soll, um variabel einsetzbar zu sein, und verschiedenen Werkzeugen wie dem Robotersystem, einer Simulationsumgebung sowie verschiedenen Eingabehilfsmitteln etc., vgl. **Bild 2.5**.

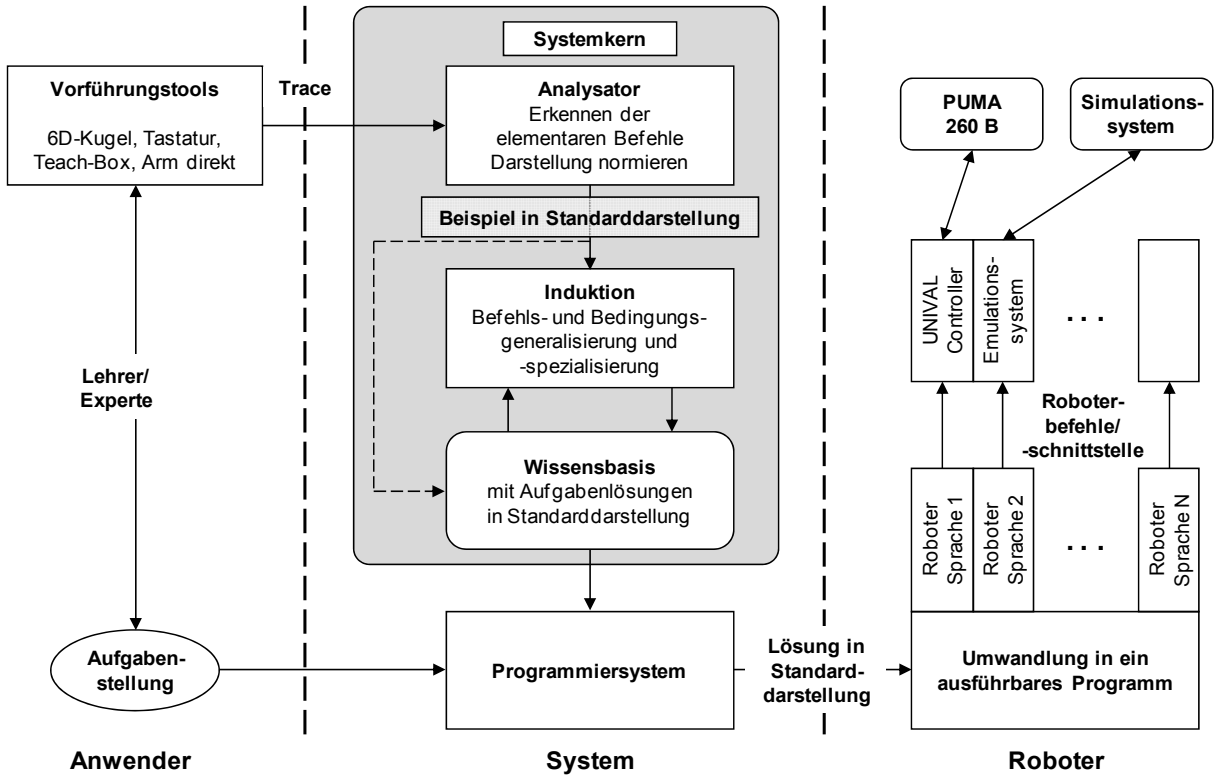


Bild 2.5: Architektur eines RPD-Systems [Mün-08]

Um Wissen aus den Benutzervorführungen ableiten zu können, werden verschiedene Ansätze aus dem maschinellen Lernen aufgegriffen und derart ab-

gewandelt, dass der Benutzer in einzelne Verarbeitungsschritte integriert werden kann. Er leitet das System, gibt Antworten, trifft eine Auswahl oder gibt Werte vor, wenn die Methoden des maschinellen Lernens allein ein Problem nicht oder nur mit (zu) großem Aufwand lösen können.

An der RWTH Aachen wurden Programmiersysteme für das automatisierte Lichtbogenschweißen entwickelt, welche die Idee der Makroprogrammierung berücksichtigen. Von ZABEL [Zab-93] wurde darauf aufbauend ein System für die werkstattorientierte Programmierung von Industrierobotern erarbeitet, wobei Makros zur automatischen Generierung der Roboterprogramme benutzt werden, die eine implizite Beschreibung des Prozesses in Abhängigkeit von der jeweiligen Nahtform ermöglichen. Es werden die Vorzüge der direkten Bewegungsführung mit den Vorteilen grafisch-interaktiver Benutzeroberflächen vereint, um den wirtschaftlichen Einsatz von Schweißrobotern auch in Kleinserien zu ermöglichen [Alm-06, Zab-93]. Durch die Entwicklung eines zugeschnittenen, werkstattorientierten Programmierverfahrens einschließlich der erforderlichen Programmierwerkzeuge wird das Einsatzgebiet von Industrierobotern im Bereich des automatisierten Lichtbogenschweißens auf die Kleinserienproduktion und die auftragsgebundene Einzelfertigung erweitert.

2.2.2 Offlineprogrammierverfahren

Bei der Offlineprogrammierung des Roboters wird dieser nicht benötigt, die Programmentwicklung erfolgt offline an einem vom Roboter unabhängigen Computer. Während der Programmierung kann der Roboter weiter betrieben werden, es gibt keine Stillstandzeiten. Zu den Offlineprogrammierverfahren zählen die

- Textuelle Programmierung,
- Programmierung per grafischer Simulation,
- Grafische Programmierung mit Ablaufdiagrammen und
- Aufgabenorientierte Roboterprogrammierung.

Bei der *Textuellen Programmierung* werden die Aufgaben auf der Basis einer problemorientierten Sprache beschrieben. Das Verfahren ist vergleichbar mit dem Programmieren in einer höheren Programmiersprache. Bei der *Programmierung aufgrund einer grafischen Simulatio*n kann der Roboter auf der Basis von CAD-Daten oder einem virtuellen generierten System programmiert werden. Hierbei werden in der Regel die Umgebung des Roboters und seine Werkzeuge abgebildet. In Ko-

operation mit der Firma Robert Bosch GmbH wurde an der RWTH Aachen das Programmiersystem BAPS plus entwickelt. Das PC-basierte, grafisch strukturorientierte System ermöglicht die Programmerstellung unabhängig von der Syntax einer Programmiersprache mit grafischen Ablaufplänen. Systeme, die ausgehend von einer Beschreibung des Ist- und des Sollzustands einer Zelle oder eines Bauteils die erforderlichen Bewegungen und Anweisungen CAD-basiert selbstständig erstellen, werden als *Aufgabenorientierte Programmiersysteme* bezeichnet. Wichtigster Schwerpunkt ist dabei, eine vollautomatische Generierung der entsprechenden Roboterprogramme zu ermöglichen [Wec-06].

Das Ziel des Forschungsvorhabens „*IROPROG“* (Intuitive Roboter-Programmiermethoden) war es, die Programmierung von Robotersystemen signifikant zu vereinfachen und zu verbessern sowie neue intuitive und automatische Programmierverfahren zu entwickeln, die den Programmieraufwand deutlich verringern. An diesem Projekt waren neben Industriepartnern das Institut für Fertigungstechnik und Werkzeugmaschinen der Universität Hannover, das Zentrum für Lern- und Wissensmanagement, der Lehrstuhl Informatik im Maschinenbau der RWTH Aachen und das Institut für Prozessrechentechnik, Automation und Robotik der Universität Karlsruhe beteiligt. Als ein Teilergebnis dieses Projektes beschreibt ZÄH [Zäh-04] eine Möglichkeit der beschleunigten Programmierung von Industrierobotern mit Hilfe des Augmented-Reality-Einsatzes zur intuitiven Mensch-Maschine-Interaktion. Hierbei wird der Einsatz von Augmented Reality (AR) zum interaktiven Generieren und Visualisieren von Roboterbahnkurven in 3D untersucht. Weiterhin wird gezeigt, wie sich mit Hilfe dieser zusätzlichen Benutzerschnittstelle die Bewegungsprogrammierung intuitiver und somit schneller gestalten lässt. Hierbei wird erreicht, dass im Vergleich zur klassischen Offlineprogrammierung der Modellierungsaufwand durch Programmierung in der realen Umgebung reduziert wird und gleichzeitig, anders als bei der Onlineprogrammierung (z. B. Teach-in-Verfahren), das Verhalten des Roboters gefahrlos in der realen Umgebung betrachtet werden kann. DENKENA [Den-05] untersuchte die Nutzerinteraktion mit einer Programmierumgebung für Robotersysteme, um die Programmierung von Industrierobotern zu vereinfachen. Ein oft vernachlässigter Aspekt bezüglich des Einsatzes von Industrierobotern ist die Frage der Effizienz ihrer Programmierung. Es wurden neuartige Ansätze und innovative Entwicklungen im Bereich der Offline-

programmierung von Robotern vorgestellt [Wör-04]. Im Zentrum steht dabei die anwendungsgetriebene Integration haptischer Eingabegeräte und leistungsfähiger Algorithmen zur automatisierten Bahnplanung in Simulationssystemen.

An der Universität des Saarlandes, Lehrstuhl für Fertigungstechnik, wurde unter der Leitung von BLEY [Ble-01] eine robotergestützte Zellenkalibrierung als Basis einer featurebasierten Montageplanung entwickelt. Der Ansatz war hier, die bei der Offlineprogrammierung notwendige Kalibrierung der Roboterzelle zu vereinfachen und außerdem die Daten, die bei der Kalibrierung gewonnen wurden, an ein Simulationsprogramm (z. B. ROBCAD) zu übertragen und dort zu verwenden [See-99]. Im Rahmen der Forschungsarbeiten ist eine Möglichkeit entwickelt worden, mit einem kostengünstigen Messsystem und auf Basis einer geringen Menge auszuwertender Wegmessdaten die Lage und Orientierung der betrachteten Komponenten einer Zelle zu berechnen. Auf der Basis dieser Messdaten kann eine Kalibrierung der Montage-/Fertigungszelle erfolgen. Darüber hinaus besteht die Möglichkeit, über eine Schnittstelle die zur Kalibrierung notwendigen Informationen in ein Simulationssystem zu übertragen und das Simulationsmodell der Zelle ebenfalls an die reale Zellenumgebung zu adaptieren. Darauf aufbauend können offline programmierte Arbeitspunkte an die reale Zelle angepasst werden. Im Verbund mit der Zelle und dem zur Messung eingesetzten Industrieroboter kann das Simulationssystem als wertvolles Planungssystem eingesetzt werden.

An der Universität Bayreuth wurde im Projekt „*ViRoP“ (*Virtuelle Roboterprogrammierung*)* unter Leitung von HENRICH [Hen-02] ein System zur virtuellen Roboterprogrammierung entwickelt, das kein detailliertes Wissen über die Eigenschaften des Systems und des zu greifenden Objektes benötigt, um beispielsweise die Handhabung von flexiblen Objekten mittels IR zu ermöglichen. Dabei wurde besonders auf das Problem der mangelnden Flexibilität der Roboterprogramme eingegangen. Zur Lösung wurden ein grafisches Robotersimulationssystem und ein Eingabegerät mit sechs Freiheitsgraden miteinander verbunden, siehe **Bild 2.6**. Diese Kombination erlaubt eine intuitive Handhabung jedes möglichen Gegenstandes in der virtuellen Zelle und liefert ein taktiles Feedback.

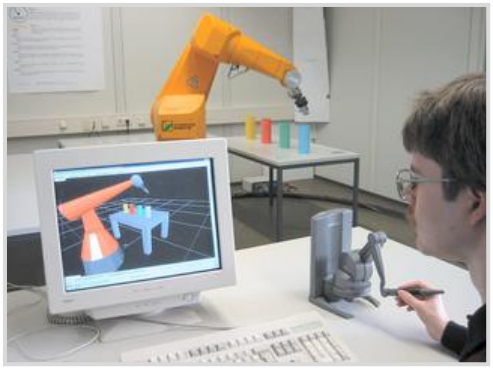


Bild 2.6: Grafisch-interaktive Roboterprogrammierung [Kah-08]

Es wird davon ausgegangen, dass diese direkte und intuitive Handhabung des Werkstückes in solch einem virtuellen System die Aufgabe der Roboterprogrammierung erheblich vereinfacht, bei gleicher Minderung der Einbeziehung des tatsächlichen Roboters [Hen-02, Hen-04, Kah-08].

Das Ziel des Verbundprojekts *„PORTHOS“* (Portable Handhabungssysteme für den ortsflexiblen Einsatz in der Produktion) an der RWTH Aachen unter der Leitung von BRECHER [Bre-04a] war es, ein portables Roboter-Gesamtsystem zur Bestückung von Werkzeugmaschinen zu entwickeln, das auf einfache Weise in verschiedenen Bearbeitungszellen eingesetzt werden kann, sich automatisch an seine neue Umgebung anpasst und intuitiv programmiert wird sowie seinen Gefahrenbereich mittels Sensoren überwacht. Das Teilprojekt *„Programmier- und Benutzungsplattform“* hatte das Ziel, eine Programmier- und Benutzungsplattform zu realisieren, die einen unkomplizierten und intuitiven Umgang mit Robotern ermöglicht. Personen mit wenig Robotererfahrung sollen mit Hilfe der Plattform in der Lage sein, komplexe Roboterprogramme zu generieren, um die vorhandenen Potenziale zur Automatisierung der Fertigung in kleinen und mittleren Unternehmen besser nutzen zu können. Das Programmiersystem basiert auf den Phasen der Modellierung und des Betriebes, um Neuprogrammierung von Handhabungsaufgaben zu ermöglichen, vgl. **Bild 2.7**.

In der ersten Phase wird ein Datenmodell aller Arbeitsplätze erstellt, an denen das Robotersystem eingesetzt werden soll. Das Ergebnis sind Datenmodelle, die alle zur Programmerstellung notwendigen Informationen enthalten, welche die Grundlage für die automatische Programmgenerierung bilden. Zur Erstellung des Modells können sowohl vordefinierte Komponenten als auch Standardkomponenten, die durch entsprechende Parameter an die reale Zelle anpassbar sind, verwendet werden. Um

den Modellierungsaufwand so gering wie möglich zu halten, wird auf eine vollständige geometrische Beschreibung der Zelle verzichtet. Da über das Programmiersystem nur Handhabungsaufgaben programmiert werden sollen, ist es ausreichend, eine Roboterzelle auf die Komponenten Roboter, Greifer, Werkstück und Werkstückspeicher zu beschränken [Bre-04a]. An die Modellierung schließt sich die Betriebsphase an. Dabei wird die Handhabungsaufgabe durch die grafisch-interaktive Zusammensetzung einzelner Operationsblöcke definiert [Bre-04b].

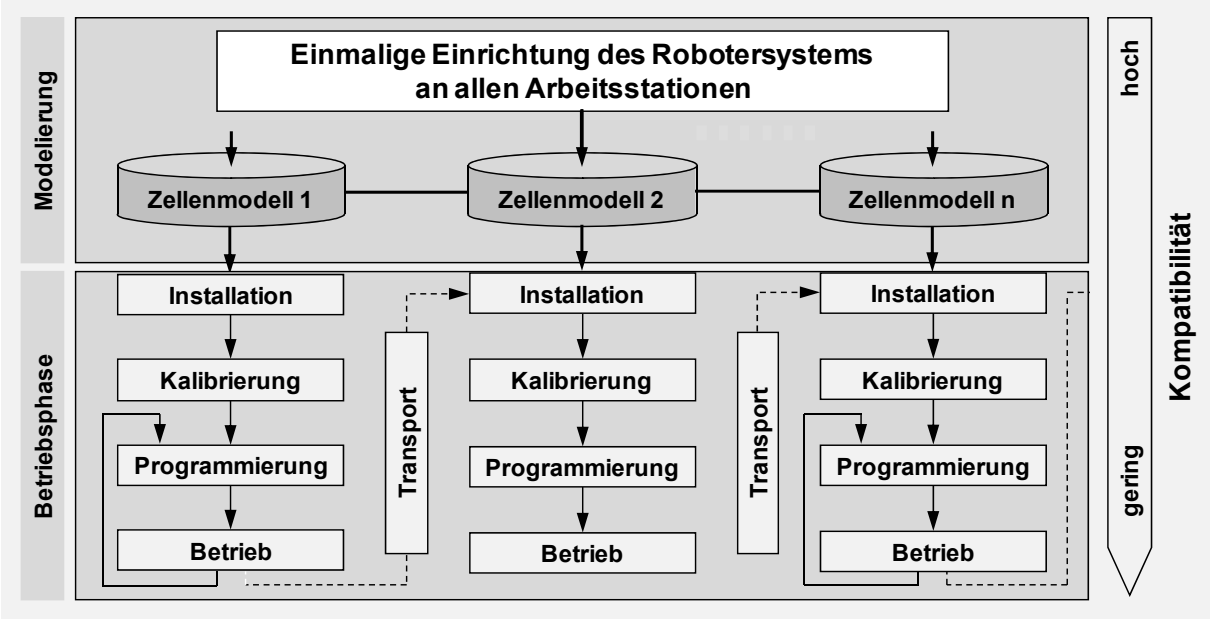


Bild 2.7: Zweistufiges Benutzungskonzept [Bre-04a]

LÜDEMANN-RAVIT [Lüd-05] entwickelte ein System zur Automatisierung der Planung und Programmierung von industriellen Roboterapplikationen unter Verwendung von CAR-Systemen (Computer Aided Robotics). Dieses System wurde konzipiert und realisiert, um automatisiert steuerungsspezifische Roboterprogrammvarianten zu erstellen, auszuführen sowie zu bewerten, wobei die Erweiterbarkeit und Modularität im Vordergrund standen. Ein zentraler Bestandteil des Systems ist der Generierungsplan, der festlegt, welche Roboterprogrammvarianten das System erstellen soll [Fre-02]. Dabei kann der CAR-Anwender den Ablauf der Programmgenerierung selbstständig formulieren. Nachdem ein Generierungsplan für ein festgelegtes Produktspektrum erstellt wurde, können auch Personen, die die umfangreichen Kenntnisse eines CAR-Anwenders nicht besitzen, neue Roboterprogramme für dieses Produktspektrum erzeugen, siehe **Bild 2.8** [Lüd-05]. Mit Hilfe der sogenannten Planaddition ist es möglich, lediglich die Änderungen eines Plans anzu-

geben und diese mit dem bestehenden Plan zu verknüpfen, statt einen ähnlichen Plan neu zu formulieren. Planschablonen stellen ein Gerüst dar, mit dem die Erstellung neuer Pläne für ähnliche Probleme vereinfacht und beschleunigt wird. Weiterhin wurde eine steuerungsunabhängige Syntax zur Spezifikation generierbarer Roboterprogramme entwickelt. Neben der Vereinfachung der Generierung von Roboterprogrammen war ein weiteres Ziel, das entwickelte System zur verbesserten Planung, Simulation und Optimierung zu nutzen.

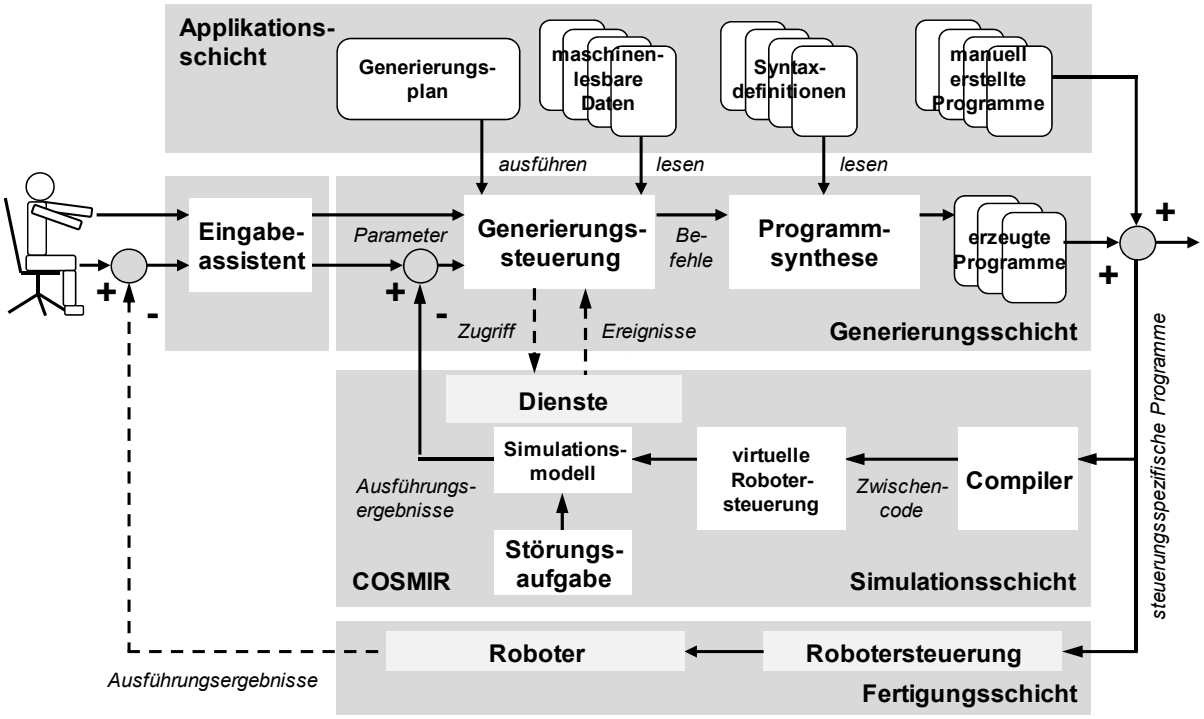


Bild 2.8: Konzept des Gesamtsystems [Lüd-05]

Das Ziel des Forschungsprojekts „*SMErobot*“ (eine europäische Robotik-Initiative zur Stärkung der Wettbewerbsfähigkeit von kleinen und mittelständischen Produktionsbetrieben) ist die Entwicklung einer neuen Robotergeneration, die speziell an die Bedürfnisse von KMU (engl. SME) angepasst ist. Das Projekt wird von der Europäischen Union gefördert und besteht aus einem Konsortium von 16 internationalen Unternehmen, die durch das Fraunhofer IPA koordiniert werden. In dem Projekt „*SMErobot“* wird eine neue modulare und interaktive Robotergeneration entwickelt, die drei Ziele verfolgt:

- Der Roboter soll leicht erlernbare, intuitive Befehle verstehen.
- Er soll alle Sicherheitsvoraussetzungen erfüllen, um am selben Arbeitsplatz wie die menschlichen Kollegen einsetzbar zu sein.

- Er soll binnen drei Tagen installiert und in Betrieb genommen werden können [Mey-06].

Der Roboter soll über Kommunikationswege bedient und programmiert werden können, die dem Menschen vertraut sind. Dazu zählen Spracheingabe und Gestenerkennung sowie das kraftgesteuerte Führen des Roboters mit der Hand. Im Gegensatz zum industriellen Einsatz kann der Roboter in KMU aus Platzmangel oft nicht in abgesicherten Roboterzellen betrieben werden, sondern muss sich den Arbeitsraum mit dem Menschen teilen [Pir-06]. Um dem Roboter leicht erlernbare Befehle geben zu können, wurden unter anderem zwei Methoden zur intuitiven Programmierung von Robotern vorgestellt:

- Transfer von Handzeichnungen in Roboterprogramme und
- Programmieren durch Vormachen, siehe **Bild 2.9**.

Anwendbar sind diese Ansätze unter anderem zur Programmierung von Fräs- und Schneidbahnen, für Schleifaufgaben und den Auftrag von Klebstoffen. Mit diesen Technologien ist es möglich, Roboter schnell und ohne lange Einlernzeit zu programmieren. Der Robotereinsatz wird auch für kleinere Losgrößen wirtschaftlich sinnvoll, die Wettbewerbsfähigkeit steigt [Mey-06].

a) Zeichnung mit digitalem Stift b) in Blech gravierte Zeichnung

Bild 2.9: Erzeugung von Roboterprogrammen mittels digitaler Stifte [Mey-06]

ROKOSSA [Rok-99] entwickelte einen umfassenden Ansatz zur prozessorientierten Offlineprogrammierung von Industrierobotern, um den Anwender während der Programmerstellung in geeigneter Weise zu unterstützen und somit die Programmierzeiten zu reduzieren. Bei der prozessorientierten Offlineprogrammierung

wird der Bearbeitungsprozess als Initiator für eine notwendige, robotergestützte Fertigung eines Produktes betrachtet, vgl. **Bild 2.10**. Auf diese Weise verbindet die prozessorientierte Offlineprogrammierung den abstrakt betrachteten Bearbeitungsprozess mit der realen Prozessausführung im Fertigungsbetrieb [Fre-98, Rok-99]. Grundlegend für die Realisierung der prozessorientierten Offlineprogrammierung ist eine geeignete Modellbildung für die Elemente der Fertigungszelle. Daher wurde im Rahmen einer makroskopischen Modellierung ein hierarchisch strukturierter Aufbau für die Arbeitszellenobjekte entworfen und durch die Verwendung von Octrees der effiziente Zugriff auf die geometrischen Daten der Umwelt ermöglicht [Rok-99].

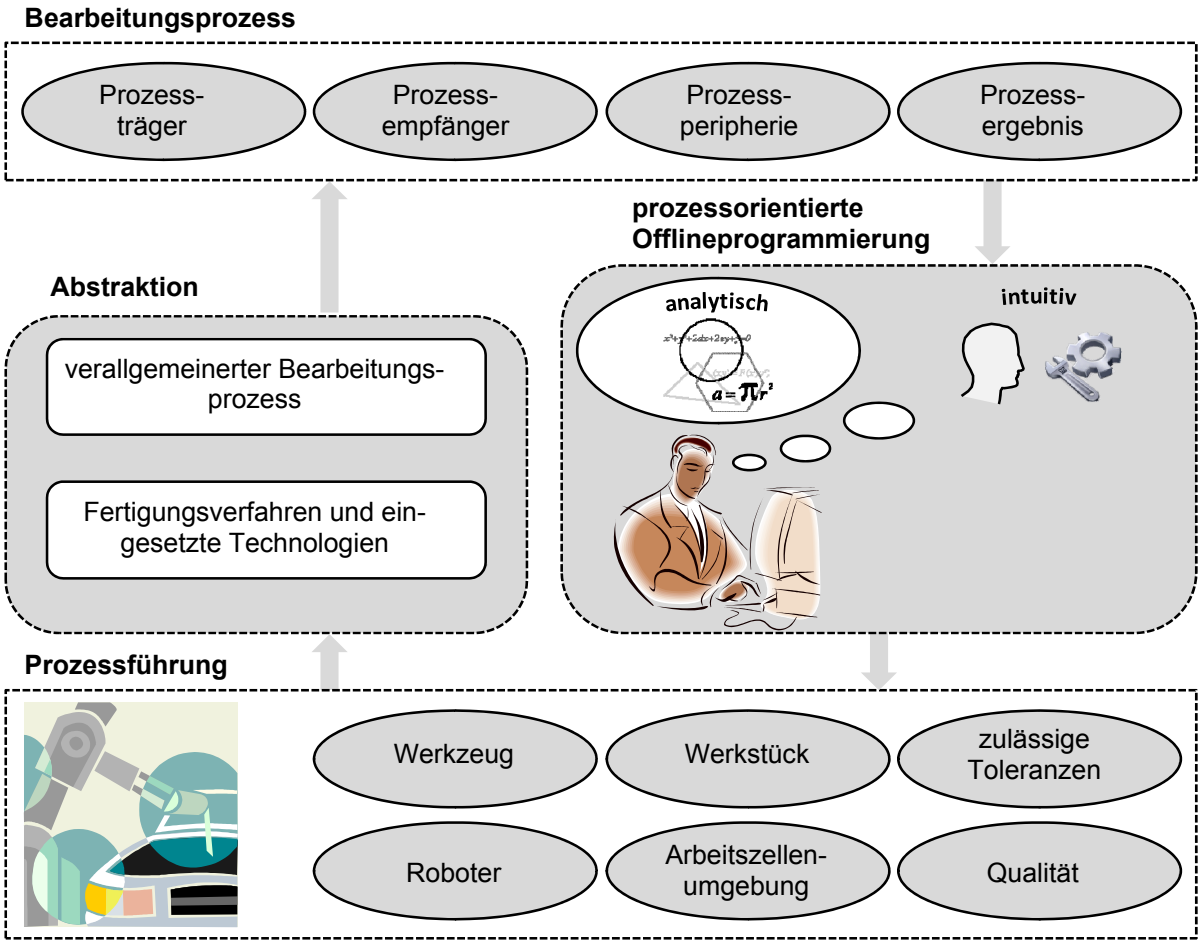


Bild 2.10: Prozessorientierte Offlineprogrammierung als Bindeglied zwischen beabsichtigtem Bearbeitungsprozess und Prozessführung [Rok-99]

HESSELER [Hes-95] entwickelte einen Ansatz zur Verbesserung der Bedienerfreundlichkeit von Offlineprogrammierverfahren. Um unerfahrenen Benutzern den Umgang mit dem System zu erleichtern und im Fehlerfall immer eine aktuelle und kontextspezifische Hilfestellung bei der Aufgabenbewältigung anbieten zu können,

wurde das Konzept der handlungsbasierten Unterstützung für Offlineprogrammiersysteme vorgestellt, siehe **Bild 2.11**. Durch den Einsatz vordefinierter Handlungspläne wurde die Möglichkeit vorgestellt, die es einerseits dem Anwendungsprogrammierer erlaubt, sein zugrundeliegendes konzeptuelles Systemmodell dem Benutzer in eingeschränkter Form zu präsentieren, die hierdurch andererseits die Bildung eines eigenen mentalen Modells unterstützt [Hes-94]. Der Schwerpunkt seiner Arbeit war die Unterstützung der Benutzer bei der Bewältigung von Denkfehlern, die auf der intellektuellen Regulationsebene bei der Planung der Vorgehensweise zur Bearbeitung der Arbeitsaufgabe mit Hilfe des Computers zu beachten und im betrachteten Fall der Offlineprogrammierung von ausschlaggebender Bedeutung sind [Hes-95].

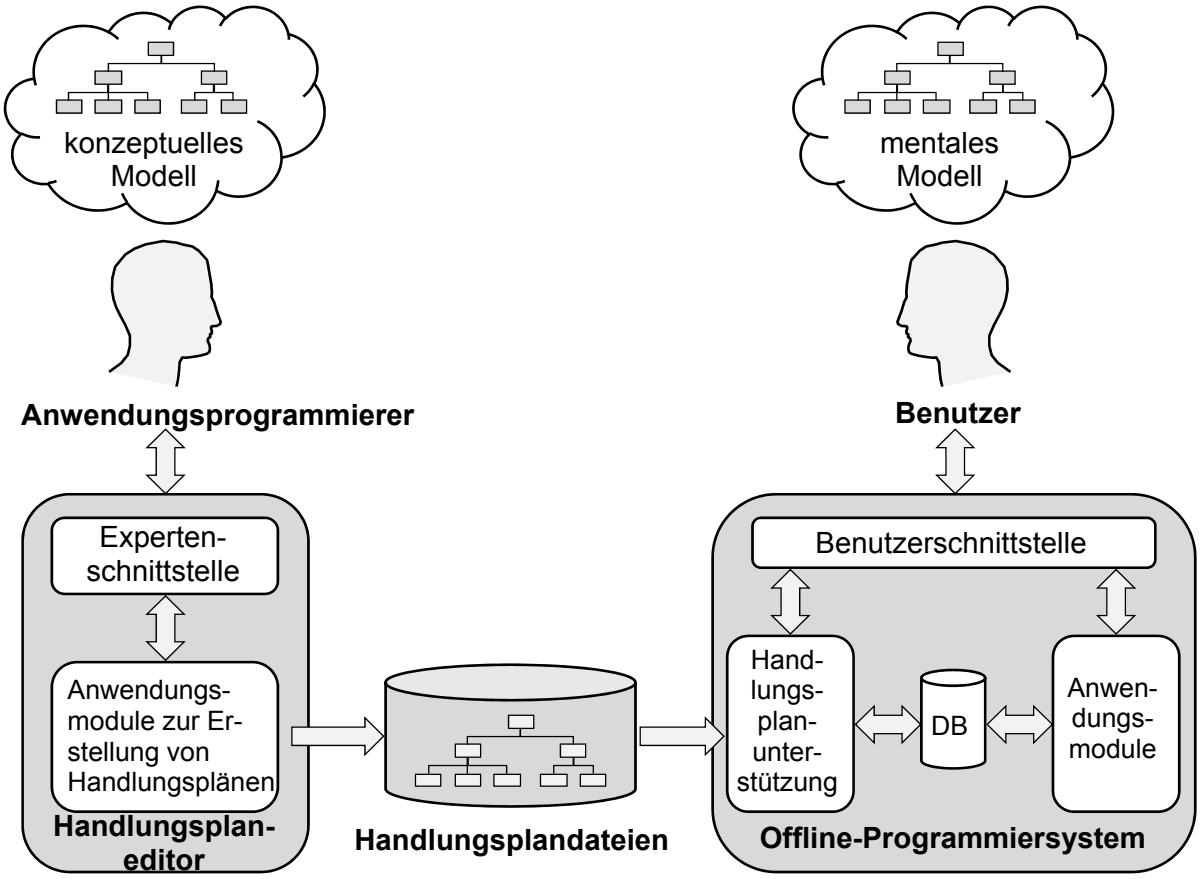


Bild 2.11: Konzept der Handlungsunterstützung bei der Offlineprogrammierung [Hes-95]

HECK [Hec-95] entwickelte ein auf neuartigen Algorithmen und Wissensdarstellungen basierendes, frei konfigurierbares System zur aufgabenorientierten Offlineprogrammierung von Industrierobotern. Über eine fensterorientierte, menü-

geführte integrierte Mensch-Maschine-Schnittstelle kann der Roboterprogrammierer interaktiv eine Arbeitszelle modellieren und Anwenderprogramme auf Aufgabenniveau erstellen. Die Struktur des aufgabenorientierten Roboterprogrammiersystems besteht aus vier Blöcken

- der Mensch-Maschine-Schnittstelle,
- dem Umweltmodell,
- dem Programmcodierer mit Schnittstellen zu externen Systemen und
- dem Planungs- und Simulationssystem, siehe **Bild 2.12** [Hec-95].

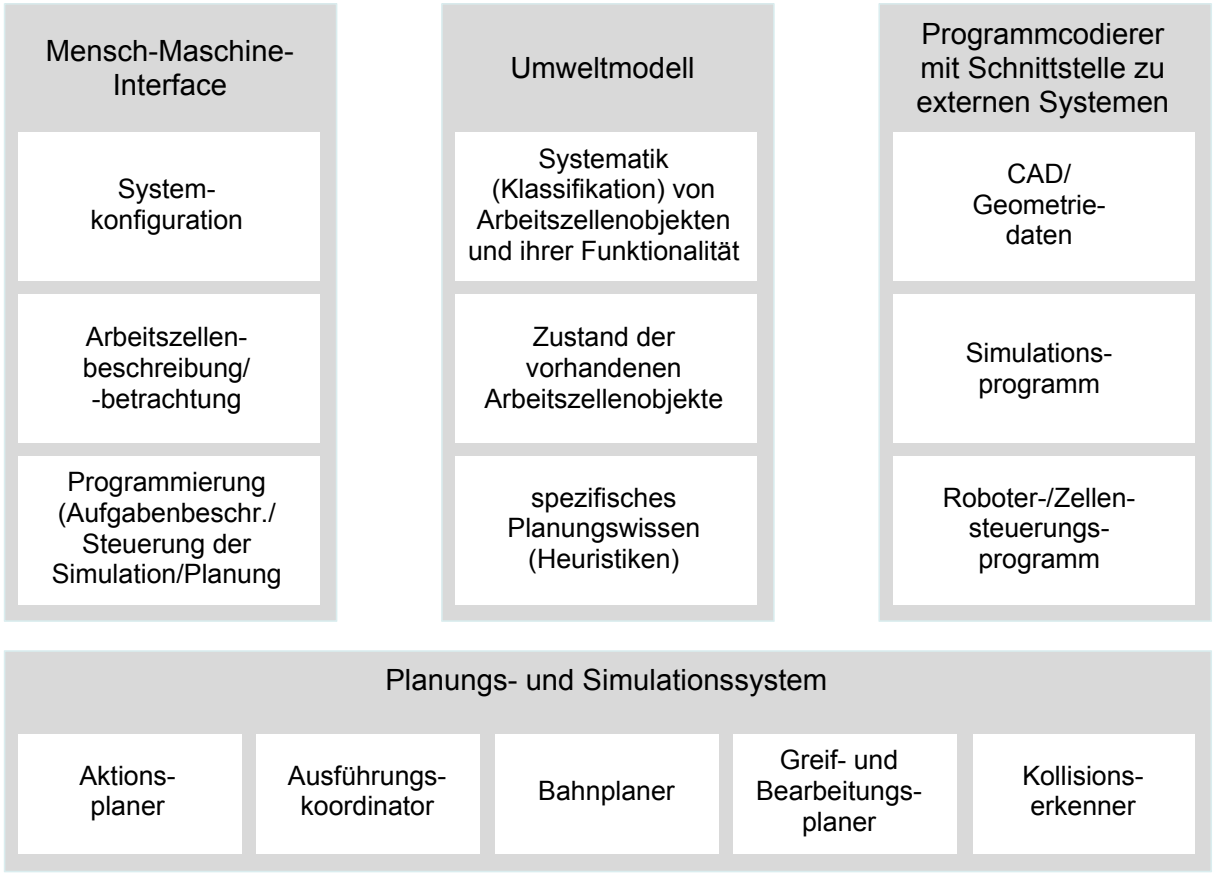


Bild 2.12: Komponenten des aufgabenorientierten Roboterprogrammiersystems [Hec-95]

Grundlage aller Planungsprozesse ist das dynamische Umweltmodell des Roboters und der Fertigungszelle, wobei sämtliche relevanten Objekte der Arbeitszelle hinsichtlich ihrer geometrisch-physikalischen Eigenschaft und Funktionalität in einer Hierarchie von Klassen dargestellt werden.

DAMMERTZ [Dam-96] präsentierte in seiner Arbeit ein Konzept für die Offlineprogrammierung von Industrierobotern, das im Rahmen eines Programmiersystems zwei getrennte Programmieroberflächen-Konzepte vereint, um die konträren Anforderungen innerhalb des gewählten Anwenderspektrums vom Roboterexperten bis zum Endanwender abdecken zu können, vgl. **Bild 2.13**. Dem erfahrenen Benutzer stehen alle Möglichkeiten einer Roboterhochsprache zur Verfügung, wie beispielsweise die Erstellung und Darstellung von Programmen oder die Eingabe von Befehlsparametern. Für einen Endanwender, der existierende Programme anpassen möchte, die ein Systemanbieter erstellt hat, wurde eine vereinfachte Programmieroberfläche vorgestellt.

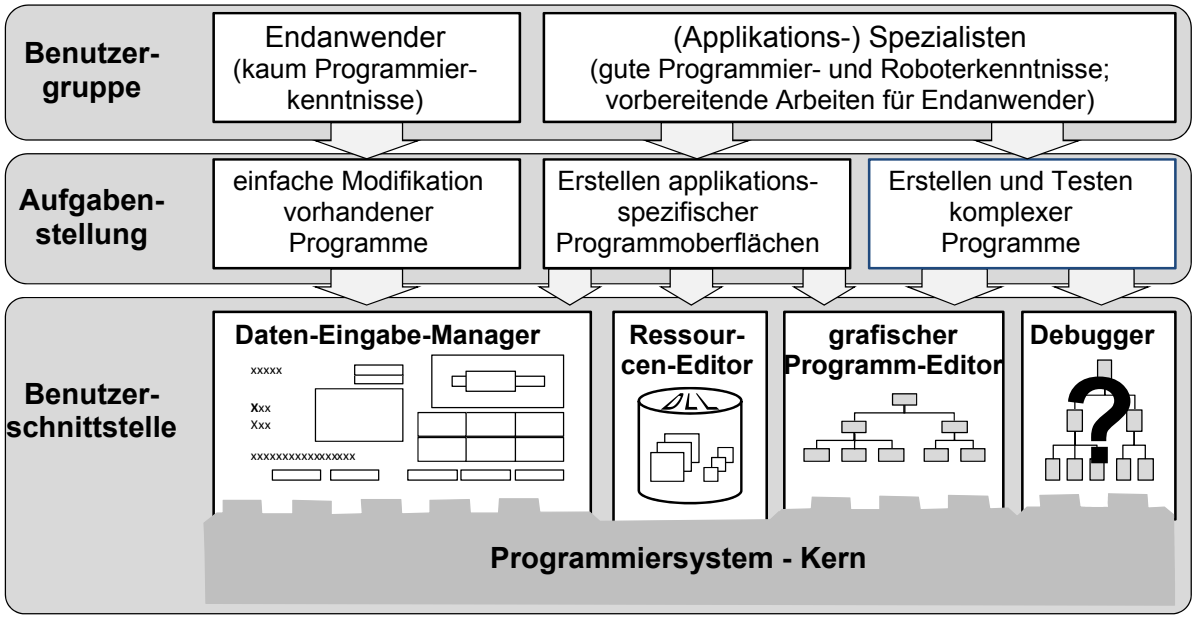


Bild 2.13: Benutzerschnittstellen für unterschiedliche Aufgabenstellungen [Dam-96]

Das vorgestellte Konzept und dessen Anforderungen wurden prototypisch in einem strukturorientierten, grafischen Roboterprogrammiersystem „*OPERA*“ (Open Programming Environment for Robot Applications) realisiert. Die Grundlage der Programmerstellung in „*OPERA*“ bildet der Programmablaufplan. Dieser hat gegenüber dem weitverbreiteten Struktogramm den Vorteil der besseren Einbindung von Grafik-Symbolen [Wec-94, Wec-97].

Das Offlineprogrammiersystem „*ProARC*“ wurde von PEPER [Pep-97] entwickelt und orientiert sich insbesondere an den Fähigkeiten des Facharbeiters (Schweißexperten), siehe **Bild 2.14**. Zielstellung war es, das Wissen des Benutzers über den Schweißprozess, beispielsweise Technologiedaten, Brennerstellung, Schweißreihen-

folge usw., unmittelbar in den Programmierprozess einzubringen, ihn jedoch von der Programmierung der Roboterbewegung zu entlasten [Pep-97]. PEPER entwickelte daher eine intuitive Programmiermethodik, mit der sich einzelne Nahtverläufe und Programmteile beliebig einfügen, verschieben und löschen lassen. Zudem ist es für erfahrene Benutzer über spezielle Programmieroberflächen möglich, Parameter des Schweißprozesses selbstständig zu optimieren. Unerfahrenen Benutzern unterbreitet man hingegen Parametervorschläge, wodurch die Anpassungsfähigkeit zwar eingeschränkt, das Programmieren an sich jedoch vereinfacht wird [Alm-06]. Für die Automatisierung des Schweißvorganges werden Schweißmakros genutzt, um auf dieser Basis die Roboterbewegungen zu generieren.

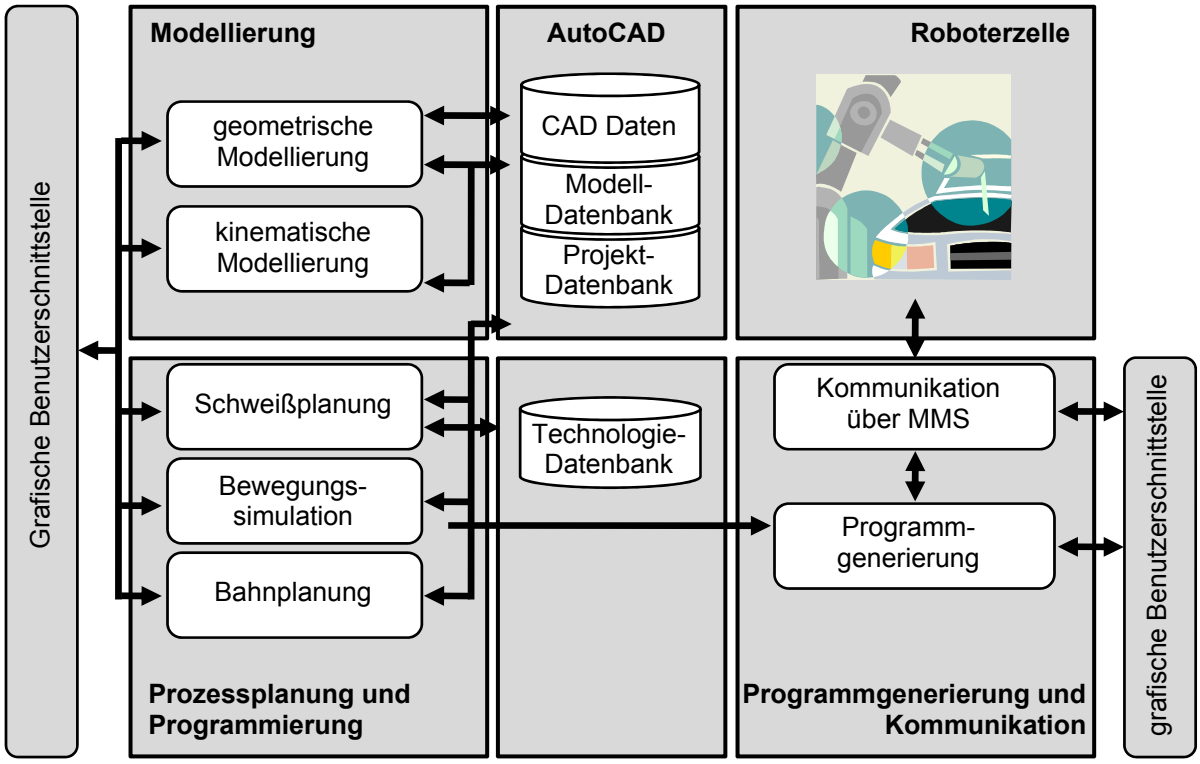


Bild 2.14: Funktionsorientierte Strukturierung des Programmiersystems „*ProARC*“ [Pep-97]

HOLLENBERG [Hol-95] entwickelte die CAD-basierte Offline-Programmierung von Lichtbogenschweißrobotern, für die nur geringe Programmierkenntnisse erforderlich sind. Eine wesentliche Neuerung dieses Ansatzes besteht darin, dass sich die CAD-Daten von Schweißkonstruktionen dazu nutzen lassen, Roboterprogramme für Schweißaufgaben zu generieren und eine geschlossene Verfahrenskette von der Konstruktion bis zur Fertigung zu realisieren. Zur Programmierung von Lichtbogen-

schweißrobotern für den wirtschaftlichen Einsatz in der Einzel- und Kleinserienfertigung wurde ein Verfahren entwickelt, das die Vorteile der CAD-Makroprogrammierung und der grafisch-interaktiven Programmierung zusammenfasst sowie an den spezifischen Erfordernissen des Handelsschiffbaus orientiert ist, siehe **Bild 2.15** [Hol-93].

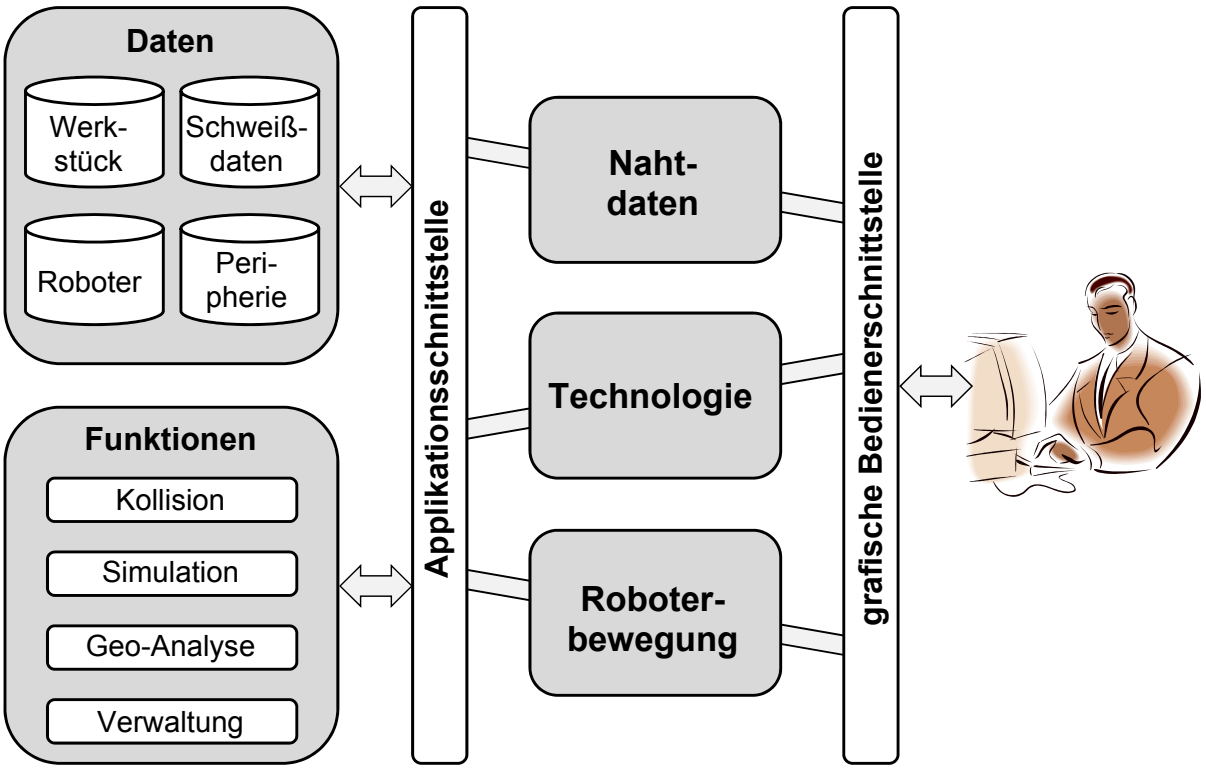


Bild 2.15: Systemarchitektur des Programmiersystems [Hol-95]

Die Programmierung erfolgt in drei Schritten. Im Zuge der Nahtdefinition erfolgt die Festlegung jener Bauteile, die durch Nähte verbunden werden, sowie der Nahtkonturen mit ihren Anfangs- und Endpunkten. Auf Basis der Nahtdaten wird vom System ein Brennerfahrweg vorgegeben, der neben dem Brennerorientierungsverlauf und den damit verbundenen Technologiedaten auch die zum Nahtanfangsuchen erforderlichen Messbahnen enthält. Für alle Bahnen müssen Roboterachsstellungen ermittelt werden, die zu einer kollisionsfreien Abfahrbarkeit ohne Überschreitung der Achsgrenzen oder Durchfahrten singulärer Stellungen führen [Hol-95].

BOTTAZI und FONESCA [Bot-05] verfolgen das Ziel, mit Hilfe von „*object oriented design pattern*“ (OODP) die Programmierzeiten von Industrierobotern entscheidend zu reduzieren. Wesentliche Funktionen und Strukturen werden durch Modelle so programmiert, dass diese Routinen in einem Baukastenprinzip angewendet werden

können, wodurch Roboterprogramme durch eine Kombination der Routinen entstehen. Häufig verwendete Module, die „*design patterns*“, werden definiert und in „factories“ zusammengefasst, wobei die Herausforderung ist, Dubletten zu vermeiden, die die Komplexität erhöhen und den Speicher belasten. Darüber hinaus wurde untersucht, wie die Objekte miteinander kommunizieren bzw. integriert werden können. Hierbei wurden standardisierte Schnittstellen entwickelt, so dass unterschiedliche Roboter die Variablen, Programmzeilen und Kommandos verstehen [Bot-05].

Die Firma KUKA Roboter GmbH [Kuk-08] verkauft für die gängigsten Roboter-Anwendungen vorgefertigte Softwarepakete zur Vereinfachung der Roboterprogrammierung und Reduzierung der Inbetriebnahmezeit. Hierbei werden verschiedenste Technologien wie beispielsweise das Laserschneiden oder das Kleben unterstützt, vgl. **Bild 2.16**. Für die unterschiedlichen Technologien werden Applikationen mit verschiedenem Umfang angeboten, von der Befehlserweiterung bis zur Unterstützung spezifischer Werkzeuge. Diese Softwarepakete beinhalten Programmmodule, die den Anwender bei der Programmierung von Roboterprogrammen unterstützen. Darüber hinaus beinhalten die Module Programmbausteine, um Werkzeuge und Sensoren einfach in die Programmierung zu integrieren [Kuk-08].

Bild 2.16: Applikation „*KUKA.LaserCut“* [Kuk-08]

2.2.3 Hybride Programmierverfahren

Hybridverfahren sind dadurch gekennzeichnet, dass die Erstellung eines lauffähigen Anwenderprogramms durch Offline- und Onlineverfahren erfolgt, wobei die Roboter in der Regel offline programmiert werden. Das heißt, es werden keine gültigen Steuerungsweisungen programmiert, sondern nur Frameworks, die für eine nachträgliche Verknüpfung mit exakten Steuerungsinformationen konzipiert sind. Mit Hilfe der Onlineverfahren werden lauffähige Anwenderprogramme erzeugt, wobei die unvollständigen Steuerungsanweisungen mit den notwendigen Informationen ergänzt werden.

Zu den Hybridverfahren gehören die aufgaben- bzw. produktorientierten impliziten Programmierverfahren. Implizite Roboterprogrammierung basiert auf spezifischem Lösungswissen, wobei es die Aufgabe des Anwenders ist, das Problem zu beschreiben, indem er die notwendigen Eingangsdaten zur Verfügung stellt [Lie-01].

LIEBENOW [Lie-01] entwickelte ein System zur Programmierung von Robotern für die Anwendung „Schutzgasschweißen“, das auf der Makroprogrammierung basiert, siehe **Bild 2.17**. Die im Rahmen seiner Arbeit vorgestellten Systemkomponenten für eine Makroprogrammierung ermöglichen es, mit einem aufgabenorientierten Hybridverfahren, Aufgabenstellungen im Stahl- oder Schiffbau zu lösen.

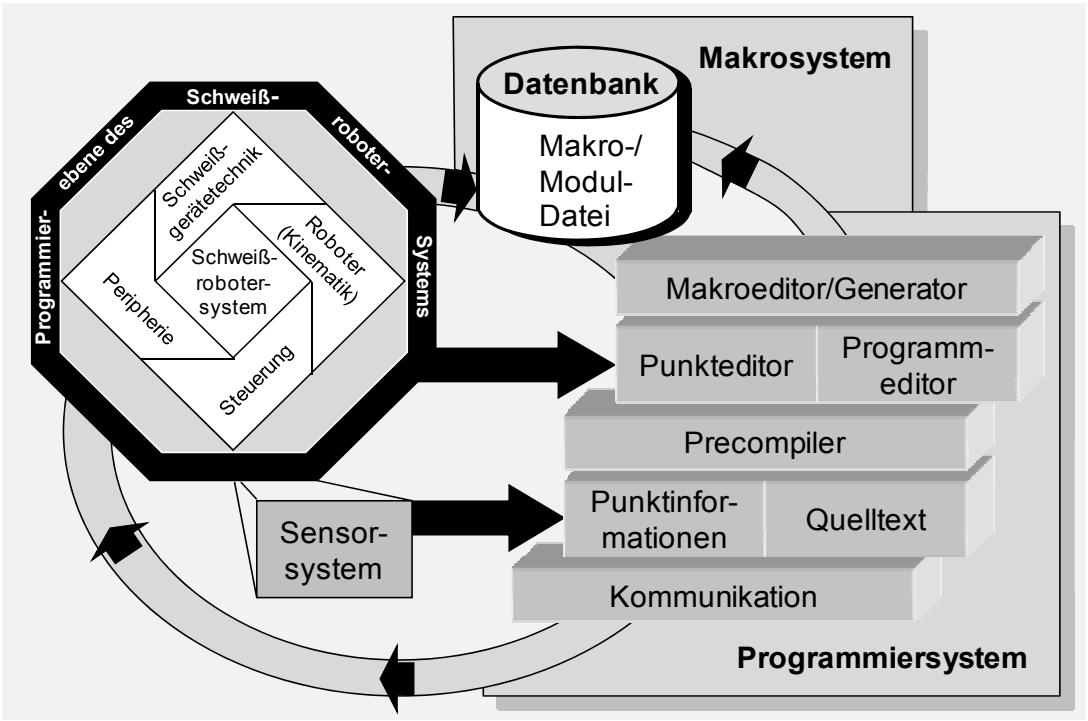


Bild 2-17: Systemarchitektur des Programmiersystems [Lie-01]

Der Schweißroboter ist insofern für die Programmerstellung erforderlich, als damit Konturpunkte aufgenommen werden. Aufgrund dieser Konturpunkte werden die Programmablauffolge und infolgedessen die Roboterprogramme generiert. Darüber hinaus hat das Programmiersystem eine Datenbankanbindung, um die Makros zu verwalten. LIEBENOW beschreibt Makros als strukturierte Bearbeitungssequenzen. Diese können offline und textuell erstellt und zur Erfüllung von weiteren gleichartigen Arbeitsaufgaben wiederholt aufgerufen werden. Durch eine Kombination dieser Makros lässt sich vor allem bei wiederkehrenden Operationen, die beim Schweißen häufig auftreten, die Programmierzeit verkürzen.

Am Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt e. V. (DLR) unter der Leitung von HIRZINGER [Hir-99] wurde ein Konzept zur aufgabenorientierten Fernprogrammierung von Industrierobotern entwickelt. Obwohl das Konzept ursprünglich für Raumfahrtanwendungen entwickelt wurde, ist es auch für terrestrische Anwendungen geeignet. Aufgabenorientiertes Fernprogrammieren basiert auf einem hierarchischen Schichtenmodell, das die Steuerung und Programmierung eines Robotersystems auf unterschiedlichen Abstraktionsebenen ermöglicht [Hir-01]. Dabei wird ein Roboter nicht mehr durch explizite Bewegungskommandos gesteuert, sondern durch implizite Aufgabenbeschreibungen. Hierbei wird unter einer impliziten Aufgabenbeschreibung die Spezifikation einer Roboteraufgabe bezüglich der Objekte seiner Arbeitsumgebung verstanden [Hir-99]. Diese implizite Beschreibungsebene bedeutet eine erhebliche Vereinfachung der Bedienung und versetzt auch Laien in die Lage, einen Roboter zu kommandieren. Das Konzept der aufgabenorientierten Programmierung beruht auf einem vierschichtigen hierarchischen Modell mit zwei expliziten (elementar beschreibbaren, manipulatorspezifischen) und zwei impliziten (aufgabenorientierten und objektbezogenen) Ebenen (2-in-2-Schichten-Modell), vgl. **Bild 2.18**.

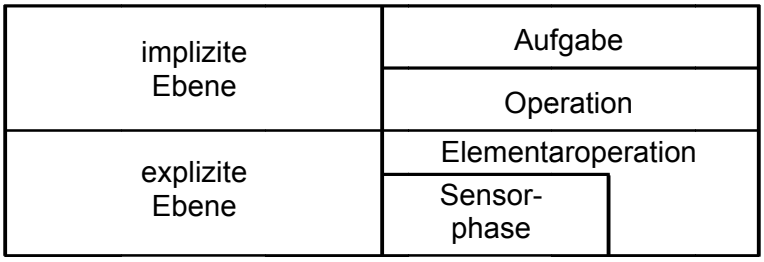


Bild 2.18: 2-in-2-Schichten-Modell für aufgabenorientierte Roboterprogrammierung [Hir-99]

Auf expliziter Ebene münden Aktuatorkommandos letztendlich in das Absetzen motorischer Bewegungsbefehle an den Roboter. Auf der impliziten Ebene wird ausgedrückt, was eine zielgerichtete Beschreibungsebene für die Bearbeitung komplexer Manipulationsaufgaben ist.

Das Konzept wurde unter der Zielsetzung entwickelt

- eine interaktive Online- und Offlineprogrammierbarkeit,
- ein hierarchisches Steuerungskonzept und
- die Bedienbarkeit für Robotiklaien

zu ermöglichen. Die Grundoperationen in einer Arbeitszelle werden von einem Roboterexperten definiert, und die intuitive Bedienung des Roboters erfolgt durch einen Laien [Hir-01].

2.3 Steuerung von Industrierobotern

Als Robotersteuerung bezeichnet man das Zusammenspiel von Hard- und Software solcher Robotersysteme, die über eine intrinsische, statische, nicht von außen manipulierbare Programmierung verfügen [Hau-06]. Als für Industrieroboter-steuerungen typische Grundaufgaben sind anzuführen:

- Koordination und Durchführung der Achsbewegungen und der Roboterperipherie,
- Verarbeitung von externen Signalen,
- Kommunikationsmöglichkeiten mit dem Anwender und
- Funktionsüberwachung von einzelnen Steuerungskomponenten [Kre-94].

Ein intelligentes Robotersystem setzt voraus, dass es frei programmierbar ist, also einerseits mit Algorithmen bzw. Methoden versorgt werden kann, um im praktischen Einsatz intelligentes Verhalten zu zeigen. Zum Anderen impliziert diese Forderung aber auch, dass das softwaretechnische System so geartet ist, dass das Robotersystem von sich aus auf wechselnde Anforderungen und Problemstellungen aus seiner Umwelt reagieren kann, ohne dass hierzu Änderungen am System vorgenommen werden müssen [Hau-06].

Die Robotersteuerung lässt sich prinzipiell in eine Ein-/Ausgabe- und Speicherebene gegenüber dem Nutzer, eine Verarbeitungsebene sowie eine Prozess-Ein-/Ausgabeebene gliedern, vgl. **Bild 2.19**. In der Ein-/Ausgabe- und Speicher-

ebene erfolgen das Bedienen und das Programmieren des Roboters. In der Verarbeitungsebene wird das gespeicherte Roboterprogramm beim Betrieb des Roboters ausgeführt. In der Prozess-Ein-/Ausgabeebene sind die Aktoren und Sensoren angesiedelt, z. B. die Antriebe des Roboters, welche die vorgegebenen Achslagen einstellen, und die Wegmesssysteme, die die Istwerte für die Regler bzw. für die übergeordnete Informationsverarbeitung erfassen [Wör-06].

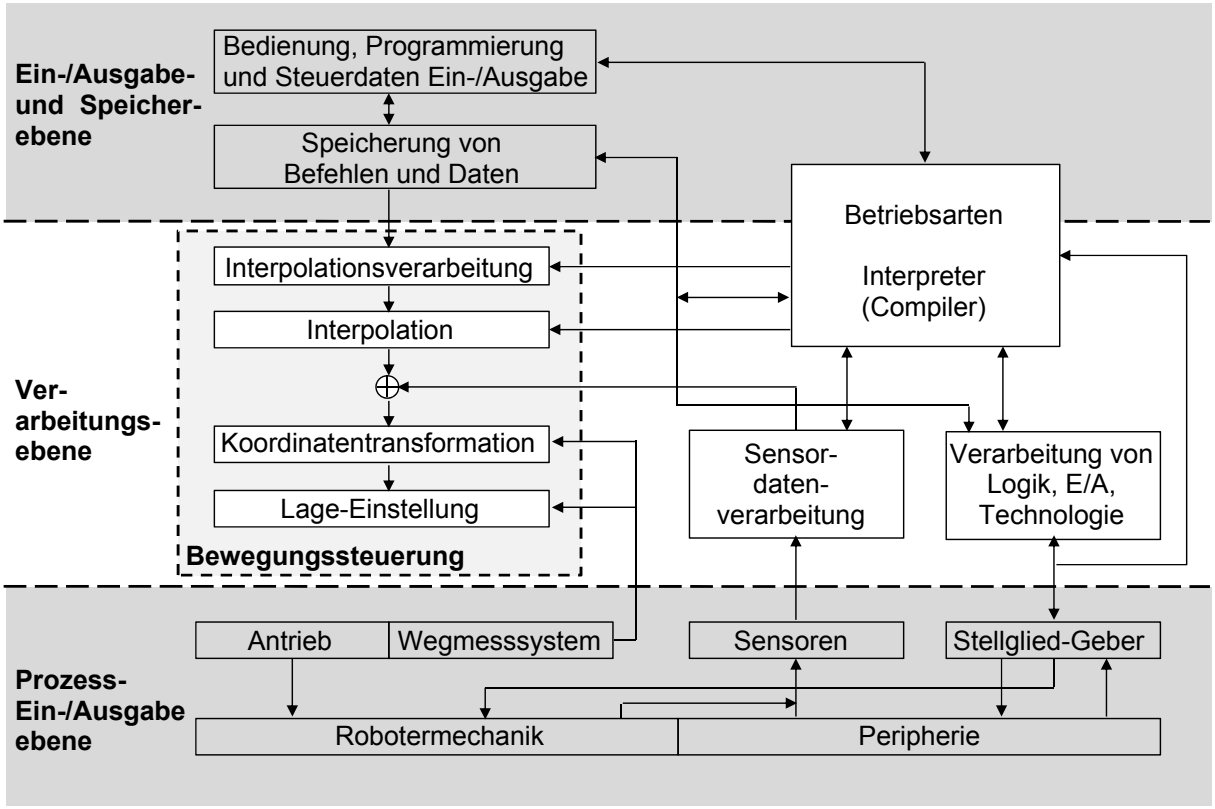


Bild 2.19: Informationsfluss und Funktionen einer Robotersteuerung [Wör-06]

Bewegungsbefehle gehören zu den wesentlichen Basisanweisungen für eine Industrierobotersteuerung. Man unterscheidet dabei folgende Bewegungsoperationen:

- PTP (Point to Point): Bewegung des Industrieroboters von der Ist- zur Sollposition des Tool-Center-Points (TCP) auf beliebiger Bahn im Raum.
- LIN (linear): lineare Bewegung des Tool-Center-Points auf der Verbindungsstrecke zwischen Ist- und Sollposition mit vorgegebener Orientierung des Tool-Center-Points.
- CIRC (circular): eindeutig durch Definition von drei Punkten im Arbeitsraum festgelegt Kreisbewegungsbahn des TCP eines Industrieroboters [Wec-06].

WEBER [Web-92] entwickelte eine Robotersteuerung „*FRC*“ (Flexibel Robot Control System), die über alle wesentlichen funktionellen Möglichkeiten industrieller Steuerungen verfügt sowie offen und einfach anpassbar ist. Die Steuerungsfunktionalität wird aus einer Bibliothek von Software-Modulen zusammengestellt, die in einer Hochsprache codiert sind und auf einem unterlagerten Echtzeitbetriebssystem arbeiten. Hierdurch ist die Steuerung von Roboter- und Steuerungsrechner-Hardware unabhängig und mit minimalem Aufwand an jede konkrete Hardware anpassbar [Web-92]. In der Standardstruktur besteht das „*FRC*“ aus den fünf Tasks: Interpreter, Bewegungssteuerung, Regler, Handbediengerät und Monitor. Der Interpreter bedient eine offene Zwischencode-Schnittstelle gemäß der durch IRDATA (Industrial Robot Data) festgelegten Syntax und Semantik [Fre-91].

HEIN [Hei-00] entwickelte für die Realisierung eines Unterstützersystems für die Chirurgie eine Systemarchitektur für eine Robotersteuerung, siehe **Bild 2.20**. Diese Steuerung berücksichtigt explizit die Einflussnahme des Chirurgen auf den Ablauf des Eingriffs und die Kooperation zwischen Robotersystem und Chirurg. Zur Vereinfachung der Bedienung wird der Ablauf eines Eingriffs in Form eines Ablaufplans und Zielgeometrien präoperativ definiert. Interoperativ folgt der Chirurg normalerweise diesem Plan [Lüt-04].

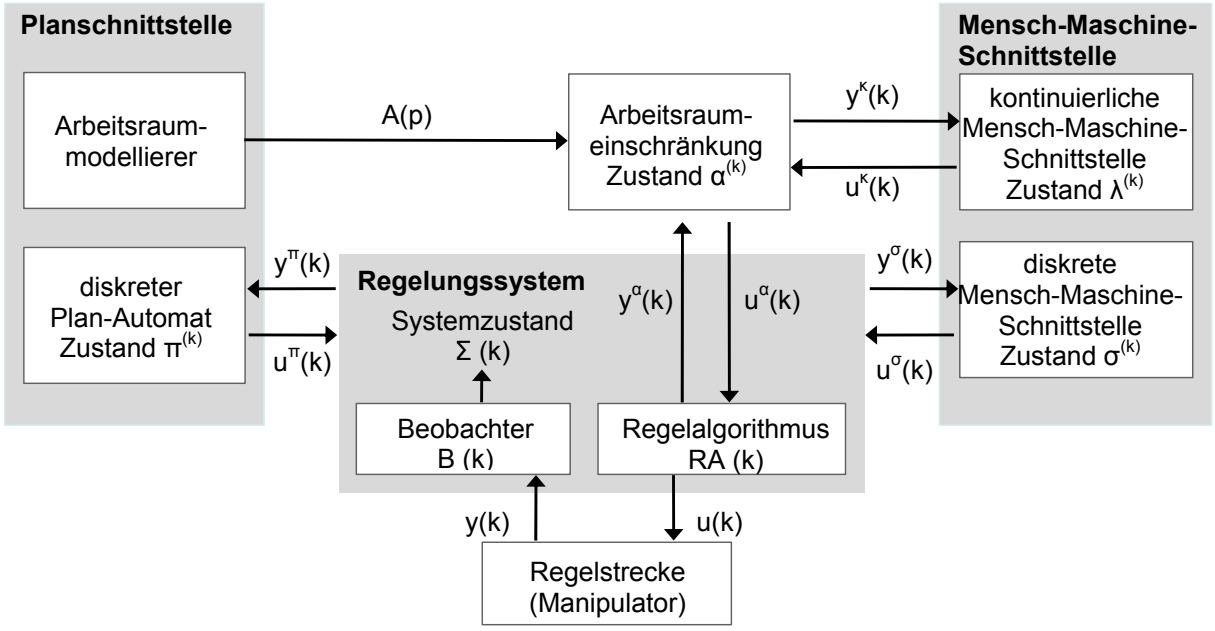


Bild 2.20: Steuerungsarchitektur eines interaktiven Robotersystems [Hei-00]

Als interaktive Führung des Roboters wird die Bedienung durch einfaches Greifen des am Manipulators befestigten Instruments sowie das Applizieren von Kraft in die

gewünschte Bewegungsrichtung verstanden. In Arbeit von HEIN wurde ein Modell für die Regelung des Roboters entwickelt, das die natürliche Dynamik eines normalen chirurgischen Instruments nachbildet. Auf diese Weise kann die Bedienung des Systems durch einen technisch nicht ausgebildeten Benutzer ermöglicht werden. Durch den entwickelten Regleralgorithmus ist das vorgestellte System das erste vollständig interaktiv bedienbare System für invasive Anwendungen [Hei-00].

VAN DER VALK [Val-95] stellt in seiner Arbeit eine leistungsfähige, modulare und flexibel einsetzbare Low-Cost-Robotersteuerung „*PCROB*“ vor, die über alle wesentlichen funktionellen Möglichkeiten industrieller Steuerungen verfügt, offene standardisierte Programmierschnittstellen bereitstellt und weitestgehend konfigurierbar sowie effizient anpassbar ist. Einen Schwerpunkt der vorgestellten Entwicklung bildet die Offlineprogrammierung der Steuerung, siehe **Bild 2.21**. Die Anbindung von „*PCROB*“ an das Robotersimulationssystem „*COSIMIR*“ (Cell Oriented Simulation of Industrial Robots) über eine Client-Server-Schnittstelle verdeutlich beispielhaft die Vorteile einer Integration von Steuerung und Simulation [Fre-93].

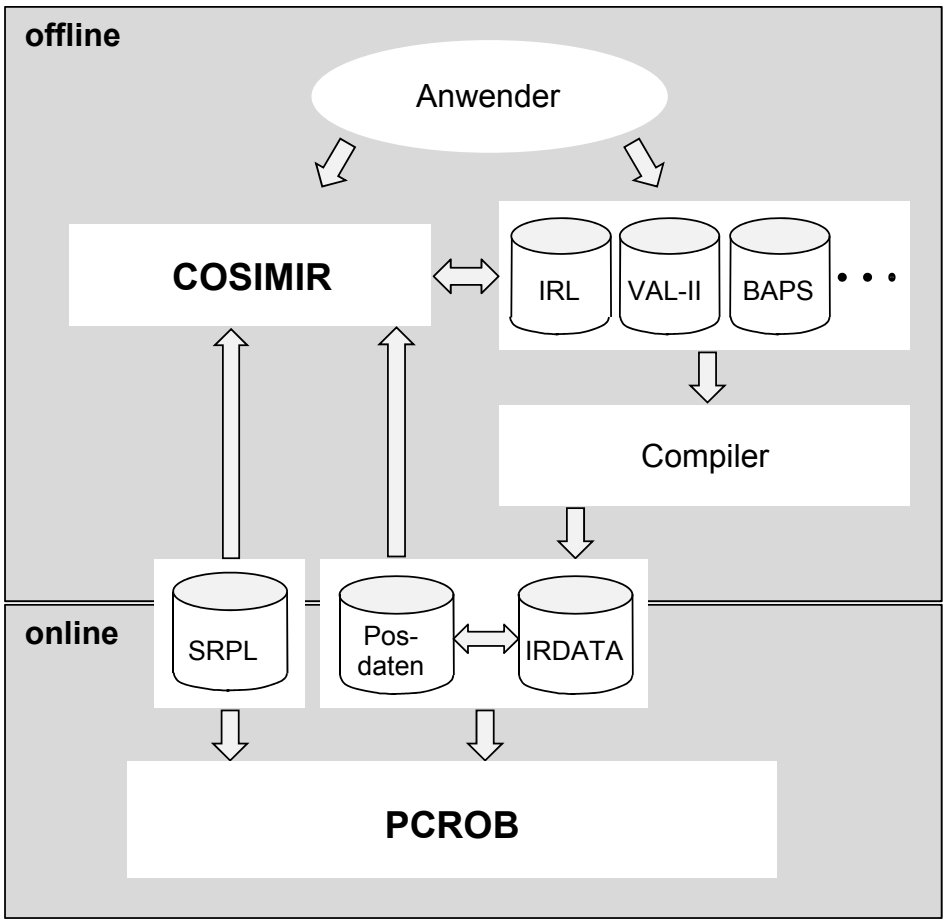


Bild 2.21: Offlineprogrammierung der „*PCROB*“ [Fre-93]

Die Konfigurierbarkeit und Anpassbarkeit der Steuerung auf den verschiedenen Systemebenen von der Offlineprogrammierung über die Onlineanwenderschnittstellen und die Bewegungssteuerung bis hin zur Schnittstellenebene zur Robotersystemhardware unterstützt die effiziente Anbindung unterschiedlicher Manipulationssysteme, wobei die Tragfähigkeit des Konzepts durch die Integration der „*PCROB*“ in die Robotersysteme NIKO N900/1, Mitsubishi RV-M1 und Manutec r2 gezeigt wurde [Val-95].

2.4 Demontage

Einen bedeutenden Bereich des Recyclings bildet die Demontage als sogenannte End-of-Life-Option, d. h. am Ende eines Produktnutzungsabschnittes. Sie bildet die Grundlage für eine Aufarbeitung von Produkten. Nach einer Reinigung, Prüfung und Sortierung werden Bauteile und Baugruppen aufgearbeitet und im Anschluss wieder montiert [Här-05]. Dabei steht die Demontage in Konkurrenz zu anderen Prozessen wie Shreddern, Pressen oder verfahrenstechnischen Lösungen. Gegenüber diesen Verfahren ermöglicht sie den Ausbau und Austausch verschlissener oder veralteter Komponenten zur Reparatur oder Erneuerung des Produktes, die Rückgewinnung funktionsfähiger Bauteile und Baugruppen zur physischen Verwendung sowie die sortenreine Separierung von Schadstoffen und wertvollen Werkstoffen zur stofflichen Verwertung [Sel-04]. Hierbei ist die Demontage von Gebrauchtgütern gekennzeichnet durch eine enorme Vielfalt an Produktvarianten und -zuständen, wobei die Herausforderung für Demontagesysteme ständig wechselnde Produktmerkmale und Demontageziele sind [Uhl-04a].

Der Demontageprozess gehört zum mechanischen Prozess des Trennens [DIN 8580]. Er lässt sich grundsätzlich in zerstörungsfreie, teilzerstörende und zerstörende Demontage unterteilen, vgl. **Bild 2.22**. Zerstörungsfreie Trennprozesse lösen die Verbindung an vorhandenen Fugen, ohne dass die Bauteile und Verbindungselemente beschädigt werden. Zerstörende Prozesse trennen z. B. unlösbare Verbindungen und schaffen neue Trennfugen, wobei die Bauteile in der Regel beschädigt werden. Zwischen diesen beiden Ausprägungen von Trennprozessen existiert ein Übergangsbereich, in dem die Bauteile keine Beeinträchtigung erfahren, während das Verbindungselement zerstört werden kann [VDI 2343-3].

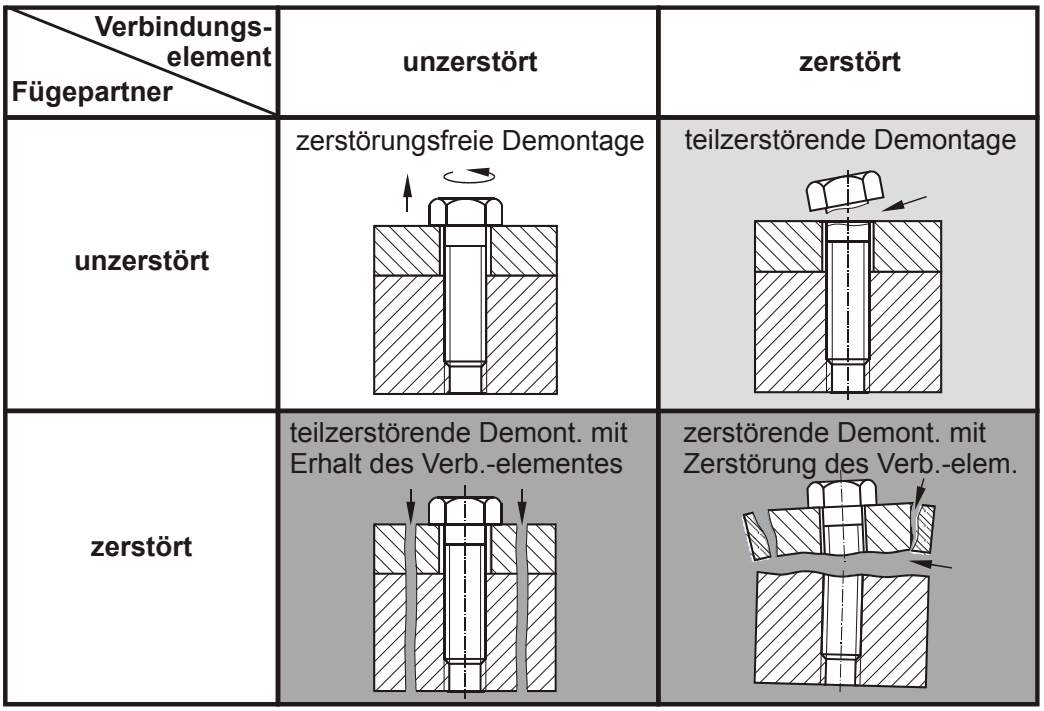


Bild 2.22: Zerstörungsgrad der Demontageprozesse [Här-05]

In Abhängigkeit von der Quelle der genutzten Energie sowie der Art der Führung, Steuerung und Überwachung der Werkzeuge kann zwischen manueller, mechanisierter und automatisierter Arbeitsweise unterschieden werden [Spu-86]. Bei der manuellen Arbeitsweise wird im Folgenden zusätzlich danach unterschieden, ob die Demontage ohne oder mit einem Werkzeug erfolgt, da zwischen beiden Varianten erhebliche Zeitunterschiede zu erwarten sind, siehe **Bild 2.23** [Här-05].

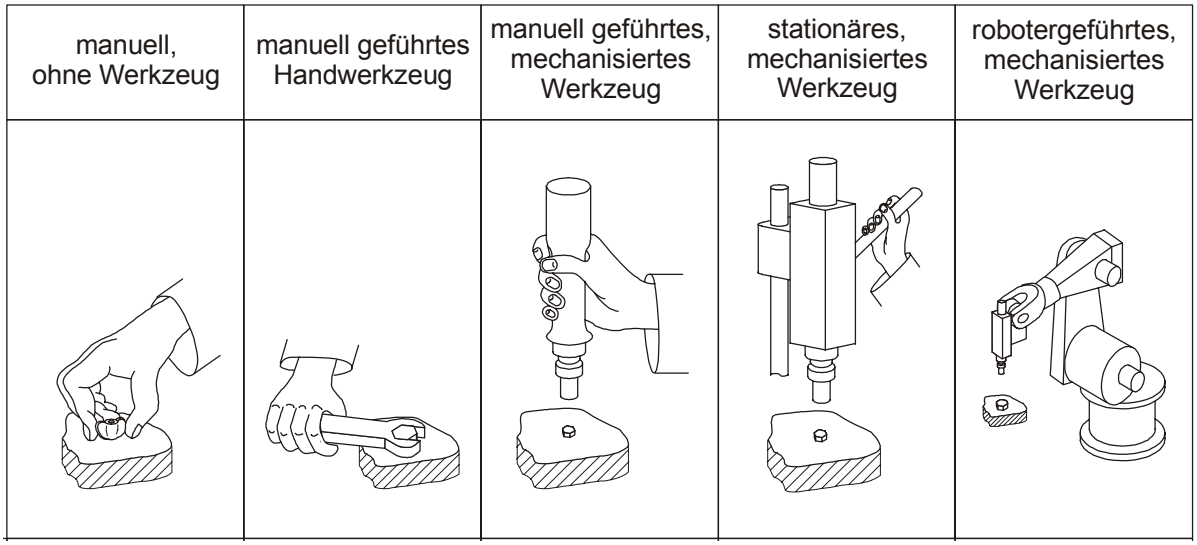


Bild 2.23: Automatisierungsgrade von Demontagesystemen [Här-05]

Demontage ist nicht die einfache Umkehr der Montage. Die bei der Demontage herrschenden Randbedingungen sind im Vergleich zur Montage wesentlich ungünstiger. SCHOLZ-REITER [Sch-00] benennt sogenannte direkte und indirekte Unsicherheiten der Demontage. Zu den direkten Unsicherheiten zählen Unsicherheitsfaktoren:

- des Produktzustandes,
- der Prozessunsicherheiten (wie Werkzeugbruch und Fehlschlagen des Prozesses) und
- die stochastische Risiken.

Zu den indirekte Unsicherheiten zählt SCHOLZ-REITER die sich verändernden:

- gesetzlichen Rahmenbedingungen,
- ökonomische Rahmenbedingungen, beispielsweise durch neue Nutzungskonzepte, und
- technologischen Rahmenbedingungen [Sch-00].

Daraus resultiert die Forderung nach einem Höchstmaß an Flexibilität für Demontagesysteme [Här-05]. In Abhängigkeit vom betrachteten Bezugsobjekt werden unterschiedliche Arten der Flexibilität gefordert. Ziele sind in Anlehnung an HENTSCHEL [Hen-96]:

1. *Produktvariantenflexibilität*, d. h. die Fähigkeit eines Systems, unterschiedliche Produkttypen und -varianten in beliebiger Reihenfolge demontieren zu können,
2. *Produktzustandsflexibilität*, d. h. die Fähigkeit eines Systems, Produkte unterschiedlicher Anlieferungszustände demontieren zu können, sowie
3. *Produktmengenflexibilität*, d. h. die Fähigkeit eines Systems, unterschiedlich anfallende Produktmengen wirtschaftlich demontieren zu können.

3 Handlungsbedarf

3.1 Herausforderungen für kleine und mittlere Unternehmen

Kleine und mittlere Unternehmen (KMU) sind Unternehmen, die definierte Grenzen hinsichtlich Beschäftigtenzahl, Umsatzerlös oder Bilanzsumme nicht überschreiten und somit begrenzten Zugang zu Ressourcen haben, siehe **Kapitel 2.1**. Darüber hinaus werden kleine Stückzahlen eines variantenreichen Produktspektrums in diesem Unternehmen gefertigt, wodurch sich das Verhältnis von Produktionszeit und Programmieraufwand verschlechtert [Neu-97, Wec-94]. Die Stärke von KMU im Vergleich zu Großbetrieben ist die Existenz kleinerer Produktionseinheiten, die eine hohe Variantenvielfalt ermöglichen [Got-01]. Hierbei sind typische Losgrößen 1 bis 10, wobei kurze Rüstzeiten gefordert werden [Rei-07]. Kleine Stückzahlen verlangen eine deutlich höhere Flexibilität der Produktionssysteme im Vergleich zu den Anwendungen in der Massenproduktion [Ver-08a].

Industrieroboter besitzen aufgrund ihrer hohen Leistungsfähigkeit vielseitige Anwendungsgebiete innerhalb flexibel automatisierter Fertigungs- und Montageanlagen. Das Aufgabenspektrum reicht dabei von einfachen Handhabungsaufgaben und Bestückungsvorgängen bis hin zu Schweißapplikationen und komplexen Fügeoperationen [Zab-93].

Um mit geringem Zeit- und Kostenaufwand an unterschiedliche Aufgabenstellungen in der Produktion angepasst zu werden, müssen Industrierobotersysteme ein hohes Maß an Flexibilität aufweisen und sich schnell umprogrammieren lassen [Lau-91, Val-95, Ver-08b]. Darüber hinaus muss die Anpassung der Roboterprogramme an die Produktion von Fertigungsfamilien unverzüglich und intuitiv realisierbar sein. Weiterhin müssen Fertigungsparameter schnell angepasst werden, da diese für eine kontinuierliche Optimierung bei wechselnden Prozessparametern genutzt werden [Rei-07, Rei-09]. Die Nutzungsdauer von Anlagen ist überschaubar und kurzfristige Umrüstungen für neue Produktvarianten müssen durch geeignete Fertigungsstrukturen unterstützt werden [Bus-96]. Mit der Flexibilität der Fertigungsstruktur und mit kurzfristigen Umrüstungen ändert sich auch der Einsatz von Roboteranwendungen in regelmäßigen Abständen. Damit ist nicht nur die Umprogrammierung, sondern auch die Änderung der Arbeitsumgebung des Roboters

durch Umstellung kompletter Fertigungsabläufe für neue Produktvarianten gemeint [Got-01].

Um eine Produktionssteigerung im KMU-Umfeld zu erreichen, müssen Robotersysteme eingesetzt werden, die vor Ort ohne Umständ durch den Werker bedient werden [IPA-06]. Damit der Roboter „als Werkzeug für den Werker“ akzeptiert sowie die Programmierarbeit durch den Werker mit seinen prozessspezifischen Kenntnissen und Erfahrungen durchgeführt werden, müssen die Bedien- und Programmierbarkeit deutlich vereinfacht und nutzergerecht gestaltet werden [Möb-96, Heß-09]. Im Idealfall sollte die Erstellung von Anwenderprogrammen durch einen Facharbeiter mit eingeschränkten Kenntnissen auf dem Gebiet der Robotik durchgeführt werden, da die Aneignung von Industrieroboterfachwissen für die Belegschaft von KMU keine Selbstverständlichkeit ist. Bei den vorhandenen Mitarbeitern in KMU ist zwar das erforderliche Fachwissen zur Erstellung von Anwenderprogrammen vorhanden, beispielweise die Bedienung von Maschinen, aber in KMU existiert keine fachbezogene Abteilung mit hochspezialisierten Mitarbeitern wie in Großunternehmen. Dies führt dazu, dass Mitarbeiter in KMU über ein wenig fundiertes Roboterfachwissen verfügen [Got-01]. Weiterhin sollte die Programmierung von der eigentlichen Programmiersprache unabhängig erfolgen, da diese von Hersteller zu Hersteller unterschiedlich ist und bei einem Systemwechsel zusätzlichen Umschulungsaufwand erfordert [Wec-94]. Des Weiteren sollten die Aktionen des Bedieners direkt auf Plausibilität geprüft werden, z. B. Konsistenzüberprüfung während der Programmerstellung, um den Bediener zu unterstützen [Bre-04, Mat-04]. Der Anwender benötigt somit ein Werkzeug, das ihm zum einen zur Erstellung eines Roboterprogramms eine komfortable Bedieneroberfläche bietet und ihn zum anderen vom steuerungsspezifischen Code weitestgehend entbindet, indem es eine Übertragung des generierten Programms auf verschiedene Steuerungssysteme ermöglicht [Möb-96]. Des Weiteren sind aufwendige Mitarbeiterschulungen, wie sie in Großunternehmen üblich sind, in der Regel in KMU nicht möglich. Daher müssen Geräte und Anlagen ein Maximum an Benutzerfreundlichkeit aufweisen, um den Schulungsaufwand auf ein Minimum zu reduzieren [Bus-96].

Aufgrund der Produktspektren von KMU sind sie auf eine heterogene Zusammensetzung ihrer Betriebsmittel mit Komponenten unterschiedlicher Hersteller angewiesen [Möb-96]. Dies bedeutet, dass einerseits die Integration von Industrie-

robotern verschiedener Hersteller gewährleistet werden, andererseits die verschiedenen Betriebsmittel mit Hilfe einer einheitlichen Programmierschnittstelle bedient werden muss. Darüber hinaus muss der Einsatz des Programmiersystems in neue Fertigungsumgebungen mit minimalem Aufwand realisiert werden können [Bre-04a, Rei-07]. Des Weiteren muss die Erweiterbarkeit eines Systems sowohl durch den Endkunden als auch durch den Systemanbieter durchführbar sein [Wec-94].

Um die Effektivität von Automatisierungsanlagen zu erhöhen, müssen zukünftig Roboterprogrammiersysteme in eine Gesamtsteuerungsarchitektur integriert werden [Rei-07]. Das bedeutet, dass eine Differenzierung der Steuerungsfunktionen von einem Programmiersystem unterstützt wird und damit Steuerungsfunktionen eigenständig ausgeführt werden. Roboterprogrammiersysteme müssen beispielsweise befähigt werden, die Ablauf- und Kommunikationssteuerung flexibel an spezifische Situationen einer Produktionsanlage anzupassen. Zusammenfassend lassen sich folgende Anforderungen an ein Programmier- und Steuerungssystem nennen, das bestmöglich für den Einsatz in KMU ausgelegt ist:

- flexible Anpassung an unterschiedliche Aufgabenstellungen,
- geringe Qualifikationsanforderungen an die Bediener,
- Unterstützung der Bediener,
- einheitliche (geräte- und steuerungsübergreifende) Bedienung und Programmierung sowie
- Einbindung in die Gesamtsteuerungsarchitektur.

3.2 Defizite aktueller Programmierverfahren hinsichtlich des Einsatzes bei kleinen und mittleren Unternehmen

Zusätzlich zu den systemimmanenten Eigenschaften der verschiedenen Programmierverfahren ergeben sich aufgrund des Einsatzes bei kleinen und mittleren Unternehmen weitere Anforderungen. Diese zusätzlichen Anforderungen führen zu weiteren Defiziten der aktuell entwickelten und verfügbaren Programmierverfahren. Wichtige Vorteile der Onlineprogrammierverfahren sind eine relativ einfache Erlernbarkeit, die Möglichkeit des genauen Anfahrens von Punkten und die Tatsache, dass der Programmierer die Positionen und Bahnen eines Industrieroboters in der realen Umwelt beobachten kann [Möb-96]. Ein wesentlicher Nachteil

der Onlineprogrammierverfahren ist, dass während der Programmierung das Robotersystem nicht verfügbar ist. Dagegen erfolgt die Offlineprogrammierung an einem robotersystemunabhängigen Computer. Die Differenzen zwischen Modell und Realität bei der Offlineprogrammierung per grafischer Simulation sind aufgrund der Realitätskomplexität bisher nicht vollständig überwindbar und kennzeichnen damit den größten Nachteil [Kle-92]. Diese Toleranzen zwischen dem Modell und der Realität erfordern eine nachträgliche Anpassung der Roboterprogramme [Kug-99].

Keines der bekannten Programmiersysteme erfüllt umfassend die in Kapitel 3.1 genannten Anforderungen von kleinen und mittleren Unternehmen. Darüber hinaus stand die Entwicklung eines hybriden Programmier- und Steuerungssystems, das speziell die Anforderungen von KMU erfüllt, bislang nicht im Mittelpunkt der Forschungsaktivitäten.

Eine flexible Anpassung der Roboterprogramme wird nur teilweise von den bisher entwickelten Programmierverfahren unterstützt. Die Anpassung der Roboterprogramme innerhalb einer Fertigungsfamilie war kein Schwerpunkt bisheriger Forschungsaktivitäten. Einen Ansatz für die Programmgenerierung innerhalb eines definierten Produktspektrums stellt LÜDEMANN-RAVIT [Lüd-05] in seinem Konzept für Programmierung mit Hilfe von CAR-Systemen vor. Ein zentraler Bestandteil seines Systemkonzepts ist der Generierungsplan, der es Personen ohne umfangreiche Kenntnisse der Roboterprogrammierung ermöglicht, neue Roboterprogramme für ein Produktspektrum zu erzeugen.

Viele der vorgestellten Entwicklungen verfolgt eine Reduktion der Qualifikationsanforderungen, um die Erstellung von Roboterprogrammen durch Werker zu ermöglichen. Dadurch soll der Endanwender während der Programmierung unterstützt, ihm die Bedienung ermöglicht und die Programmierzeit reduziert werden. Hierbei werden unterschiedliche Konzepte verfolgt. Beispielsweise wird im Projekt „*IROPROG*“ der Augmented-Reality-Einsatz zur intuitiven Mensch-Maschine-Interaktion verfolgt, um die Programmierung von Robotersystemen zu vereinfachen und den Programmieraufwand zu verringern [Den-05, Wör-04]. HESSLER [Hes-95] entwickelte ein Konzept zur handlungsbasierten Unterstützung für Offlineprogrammiersysteme, um die Bedienung zu vereinfachen und den Anwender bei der Bewältigung von Denkfehlern zu unterstützen. Durch den Einsatz vordefinierter Handlungspläne wird es

möglich, dass der Anwendungsprogrammierer das Systemmodell dem Benutzer in eingeschränkter Form präsentiert, so wird die Benutzerfreundlichkeit verbessert.

Da sich die Programmiersprachen von Hersteller zu Hersteller unterscheiden, sollten die Programmierverfahren eine geräte- und steuerungsübergreifende Bedienung und Programmierung ermöglichen, um zusätzlichen Schulungsaufwand bei einem Systemwechsel zu vermeiden. Diese Eigenschaft ist bei den Offlineprogrammierverfahren systemimmanent, da die Anwenderprogramme mit Hilfe von Postprozessoren erzeugt werden. Darüber hinaus kann bei hybriden Programmierverfahren eine steuerungsunabhängige Programmierung ermöglicht werden, indem die Grundoperationen einer Robotersteuerung herstellerneutral ausgeführt werden.

Programmiersysteme wurden bisher noch nicht in die Gesamtsteuerungsarchitektur einer Produktionsanlage integriert, sie werden nur zur Erstellung von Roboterprogrammen genutzt. Durch die Verknüpfung von Programmier- und Steuerungsfunktionalitäten kann die Effizienz von Robotern und somit der gesamten Automatisierungsanlagen erhöht werden. Aufgrund dieser Verknüpfung wird die Flexibilität gesteigert und somit ein wirtschaftlich sinnvoller Einsatz bei kleinen Losgrößen ermöglicht. Sowohl die zunehmende Anlagenkomplexität als auch die Bemühung um Steigerung der Flexibilität führen zu einem Übergang von der reinen Bewegungsprogrammierung zu komplexeren programmlogischen Strukturen. Darüber hinaus führt der Einsatz von Robotern in verketteten Systemen zu einer deutlichen Steigerung der Anlagenkomplexität sowie des Aufwands für die Integration in die Fertigungsanlage und die Kopplung mit seiner Peripherie [Wec-94]. Zusammenfassend sind die Charakteristiken der Programmierverfahren in **Tabelle 3.1** und **Tabelle 3.2** dargestellt. Hierbei wurden die verfügbaren Programmiersysteme hinsichtlich der Eigenschaften bewertet, die aufgrund der Nutzung bei kleinen und mittleren Unternehmen entstehen.

Tabelle 3.1: Charakterisierung und Bewertung der Programmierverfahren (Teil 1)

	Charakteristika / **Projekt / Verantwortliche**	Programmanpassung	Parameteranpassung	geringe Qualifikations-anforderungen	einheitliche Bedienung und Programmierung	Einbindung in die Gesamtsteuerungsarchitektur
Online-programmier-verfahren	Unscharfe Roboter-programmierung; *RWTH Aachen*	+	+	-	+	0
	Programmieren durch Vor-machen; *Universität Karlsruhe*	-	-	+	+	0
	Werkstattorientierte Programmierung (Zabel) *RWTH Aachen*	-	-	+	+	0
Offline-programmier-verfahren	IROPROG *Forschungsverbund*	-	-	+	+	0
	Robotergestützte Zellen-kalibrierung als Basis einer featurebasierten Montage-planung *Universität des Saarlandes*	-	-	-	+	0
	ViRoP *Universität Bayreuth*	0	-	+	+	0
	PORTHOS *RWTH Aachen*	+	0	+	+	0
	Automatisierung der Planung und Programmierung von industriellen Roboter-applikationen (Lüdemann-Ravit) *TU Dortmund*	+	+	-	+	0
	SMErobot *Forschungsverbund*	0	-	+	+	0
	Prozessorientierte Offline-programmierung (Rokossa) *TU Dortmund*	0	-	+	+	0
Legende	+ vorhanden	0 teilweise vorhanden		- nicht vorhanden		

Tabelle 3.2: Charakterisierung und Bewertung der Programmierverfahren (Teil 2)

Projekt / Verantwortliche \ Charakteristika		Programmanpassung	Parameteranpassung	geringe Qualifikations-anforderungen	einheitliche Bedienung und Programmierung	Einbindung in die Gesamtsteuerungsarchitektur
Offline-programmier-verfahren	Verbesserung der Bediener-freundlichkeit von Offlineprogrammierverfahren (Hesseler) *TU Dortmund*	0	-	+	+	0
	Aufgabenorientierte Offline-programmierung (Heck) *Fernuniversität in Hagen*	0	-	+	+	0
	Programmiersystem OPERA (Dammertz) *RWTH Aachen*	+	-	+	+	0
	Programmiersystem ProArc (Peper) *RWTH Aachen*	-	+	+	+	0
	CAD-basierte Offline-programmierung (Hollenberg) *RWTH Aachen*	0	-	+	+	0
Hybride Programmier-verfahren	Makroprogrammierung (Liebenow) *RWTH Aachen*	+	-	+	+	0
	Aufgabenorientierte Fern-programmierung *Deutsches Zentrum für Luft- und Raumfahrt e. V.*	0		+-	+	0
Legende	+ vorhanden	0 teilweise vorhanden		- nicht vorhanden		

3.3 Anforderungen an technologieorientierte Programmier- und Steuerungssysteme

Die Anforderungen an technologieorientierte Programmier- und Steuerungssysteme resultieren aus der Analyse der Defizite vorhandener Programmiersysteme im zuvor beschriebenen Kapitel. Darüber hinaus werden die Anforderungen von kleinen und mittleren Unternehmen zu Anforderungen an das Programmier- und Steuerungs-

system transformiert. Hierbei werden die Anforderungen in Grund-, Programmier- und Steuerungsfunktionen eingeordnet. Eine Zusammenfassung der Anforderungen ist in **Tabelle 3.3** zu finden.

Tabelle 3.3: Anforderungsliste

TU Berlin, IWF FG Werkzeugmaschinen und Fertigungstechnik		Anforderungsliste Technologieorientiertes Programmier- und Steuerungssystem	Blatt: 1 Seite: 1
Änderung	**F/W**	**Anforderung**	**Verantw.**
		Grundfunktionen	Entwickler Programmier- und Steuerungssystem
	F	Bedienung durch einen Werker/Facharbeiter	
	W	interaktive Unterstützung des Bedieners	
	F	schnelle Neuprogrammierung innerhalb einer Fertigungsfamilie ≤ 3 AT	
	F	Integration von Industrierobotern unterschiedlicher Hersteller	
	W	schnelle Anpassung der Roboterprogramme an veränderte Fertigungsstrukturen	
	F	Einbindung in die Gesamtsteuerung	
	F	offene Systemarchitektur	
	W	funktionale Erweiterung durch einen Endkunden	
		Programmierfunktionen	
	F	aufgabenorientierte Programmierung	
	W	weitgehend automatisierte Generierung eines roboter-unabhängigen Anwenderprogramms	
	F	herstellerunabhängige Programmierung und Steuerung	
	F	kurze Stillstandzeiten während der Programmierung	
	W	geringer Testaufwand	
		Steuerungsfunktionen	
	F	schnelle Anpassung der Prozessparameter ≤ 5 min	
	F	schnelle Anpassung der Anwenderprogramme ≤ 10 min	
	W	weitgehend automatisierte Ausführung von Steuerungs-funktionen	
		1. Ausgabe Oktober 2008	**F**orderung/**W**unsch

Das Ziel der vorliegenden Arbeit ist die Entwicklung eines Programmier- und Steuerungssystems, das die flexible Programmierung und Steuerung von Industrie-robotern ermöglicht, um Industrieroboter ökonomisch sinnvoll in kleinen und mittleren

Unternehmen einsetzen zu können. Das Programmier- und Steuerungssystem soll KMU in die Lage versetzen, Roboter

- effizient bei kleinen Losgrößen einzusetzen,
- flexibel für unterschiedliche Aufgaben zu programmieren,
- unter einfacher Anpassung der Parameter zu steuern sowie
- durch einen Werker/Facharbeiter zu programmieren und zu steuern.

Die Grundfunktionen bestimmen allgemeine und grundsätzliche Anforderungen an technologieorientierte Programmier- und Steuerungssysteme. Eine zentrale Grundfunktion ist die Programmierung und Steuerung des Systems durch einen Werker bzw. Facharbeiter, die durch die besondere Personalsituation von KMU bestimmt wird. Weitere wichtige Grundfunktionen sind der Einsatz des Systems unabhängig von einer Robotersteuerung und die Integration von heterogenen Automatisierungskomponenten. Des Weiteren ist die Einbindung des Systems in die Gesamtsteuerungsarchitektur der Produktionsanlage eine entscheidende Basisfunktion, um die Gesamteffizienz der Produktionsanlage zu erhöhen.

Die Aufgabenorientierung des Systems ist eine zentrale Programmier- und Steuerungsfunktion, die es dem Nutzer ermöglicht, das System intuitiv zu bedienen. Darüber hinaus unterstützen technologieorientierte Programmier- und Steuerungssysteme die automatisierte Generierung von Anwenderprogrammen, so dass die Programmierung auf Basis des technologischen Wissens des Anwenders erfolgt. Weiterhin wird die Bedienung des Systems unabhängig von der Robotersteuerung erfolgen.

Die Steuerungsfunktionen sind Voraussetzungen, um Industrieroboter ökonomisch sinnvoll bei KMU einzusetzen. Diese Funktionen ermöglichen eine flexible Produktion bei kleinen Stückzahlen eines variantenreichen Produktspektrums. Eine schnelle Anpassung der Prozessparameter und Anwenderprogramme ermöglicht die Reaktion auf veränderte Rahmenbedingungen.

4 Zielsetzung und Vorgehensweise

Ziel dieser Arbeit ist die Entwicklung eines technologieorientierten Programmier- und Steuerungssystems für Industrieroboter. Hierbei wird eine Systemlösung erarbeitet, um Industrieroboter aufgrund der gegebenen Anforderungen von KMU bedarfsgerecht zu programmieren und zu steuern. Weiterhin werden Lösungsansätze dargestellt, die über das Konzept des technologieorientierten Programmier- und Steuerungssystems hinausgehen. Zu diesen Lösungen gehören beispielsweise Nutzungsstrategien in Kooperationsverbünden oder mögliche Erweiterungen des Systems für einen effektiveren Einsatz bei KMU. Das Programmier- und Steuerungssystem basiert auf der Verwendung von kleinsten, generischen Programmelementen, die erst in ihrer Kombination und Verknüpfung mit den entsprechenden Steuerungsinformationen ein Anwenderprogramm ergeben.

In der vorliegenden Arbeit wird das Konzept entwickelt sowie prototypisch im Pilot-Demontagesystem der TU Berlin realisiert und erprobt, wobei einerseits die Leistungsfähigkeit und andererseits die Bedienerfreundlichkeit des Systems nachgewiesen werden. Das in der Arbeit zugrunde liegende Vorgehen ist schematisch in **Bild 4.1** dargestellt.

Im Kapitel 2 „Stand der Erkenntnisse“ wurden die zum Verständnis der Arbeit erforderlichen Begriffe geklärt und definiert sowie relevante Programmier- und Steuerungskonzepte vorgestellt. Für die zukünftige Entwicklung des technologieorientierten Programmier- und Steuerungssystems wurden bisherige Lösungskonzepte für die Programmierung von Industrieroboter analysiert. Hierbei stehen die Ansätze für eine hybride Programmierung im Mittelpunkt der Diskussion, also Verfahren, die zur Programmierung eines Roboters offline und online Werkzeuge einsetzen und somit eine Kombination der Offline- und Onlineprogrammierverfahren darstellen. Weiterhin werden Steuerungskonzepte für Industrieroboter präsentiert. Abschließend erfolgt eine Vorstellung des Themas „Demontage“ sowie deren Herausforderungen und Lösungen. Da bei der Automatisierung der Demontage hohe Anforderungen an die Flexibilität gestellt werden, wird das technologieorientierte Programmier- und Steuerungssystem exemplarisch im Pilot-Demontagesystem der TU Berlin realisiert.

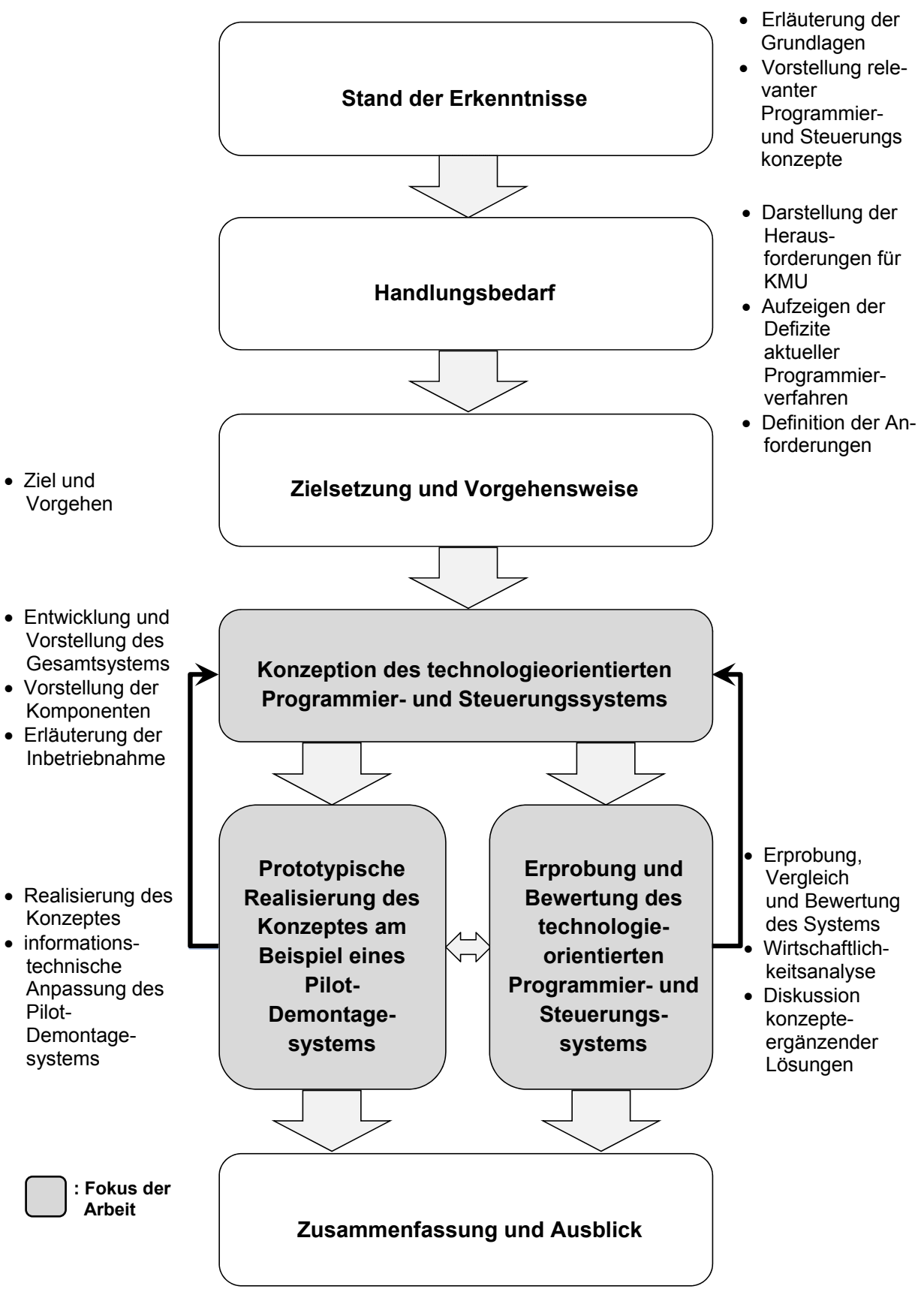


Bild 4.1: Gliederung der Arbeit

Das Kapitel 3 „Handlungsbedarf“ zeigt die Herausforderungen für KMU hinsichtlich des ökonomisch sinnvollen Einsatzes von Industrierobotern auf. Daraus abgeleitet werden die Defizite aktueller Programmierverfahren dargestellt. Hierzu zählen insbesondere Defizite wie Flexibilität bei der Programmerstellung oder Bedienung durch gering qualifizierte Endanwender. Weiterhin werden Anforderungen an zu entwickelnde technologieorientierte Programmier- und Steuerungssysteme definiert. Diese Anforderungen bilden die Grundlage für die Entwicklung des Konzepts im Kapitel 5.

Im Kapitel 4 „Zielsetzung und Vorgehensweise“ wird das Ziel dieser Arbeit zusammengefasst. Darüber hinaus wird das Vorgehen zur Erreichung der Zielsetzung vorgestellt sowie kurz beschrieben.

Das Kapitel 5 „Konzeption des technologieorientierten Programmier- und Steuerungssystems“ präsentiert das entwickelte Gesamtsystem und deren Komponenten. Das Konzept besteht aus den vier Komponenten Datenbanksystem, Speicherprogrammierbare Steuerung, Robotersteuerung und Programmierleitstand, wobei die Komponenten teilweise in das Gesamtsystem integriert oder neu entwickelt werden. Des Weiteren wird die Inbetriebnahme des Systems erläutert.

Im Kapitel 6 „Prototypische Realisierung des Konzeptes am Beispiel eines Pilot-Demontagesystems“ wird beispielhaft die Realisierung des technologieorientierten Programmier- und Steuerungssystems vorgestellt. Hierzu werden das Pilot-Demontagesystem der TU Berlin vorgestellt und die informationstechnische Anpassung präsentiert. Die Speicherprogrammierbare Steuerung und die Robotersteuerung werden modifiziert sowie die Konzepte für das Datenbanksystem und den Programmierleitstand prototypisch umgesetzt.

Innerhalb des Kapitels 7 „Erprobung und Bewertung des Programmier- und Steuerungssystems“ werden die Erprobung und Bewertung des Systems vorgenommen. Hierfür wird die Demontage einer Waschmaschine vor und nach der Inbetriebnahme des Programmier- und Steuerungssystems in das Pilot-Demontagesystem miteinander verglichen und bewertet. Des Weiteren wird eine Wirtschaftlichkeitsabschätzung durchgeführt, um den ökonomisch sinnvollen Einsatz von Industrierobotern bei KMU aufzuzeigen. Darüber hinaus werden Lösungen für den Einsatz des technologieorientierten Programmier- und Steuerungssystems vorgestellt, die über die grundsätzliche Entwicklung des Systems hinaus gehen. Zu

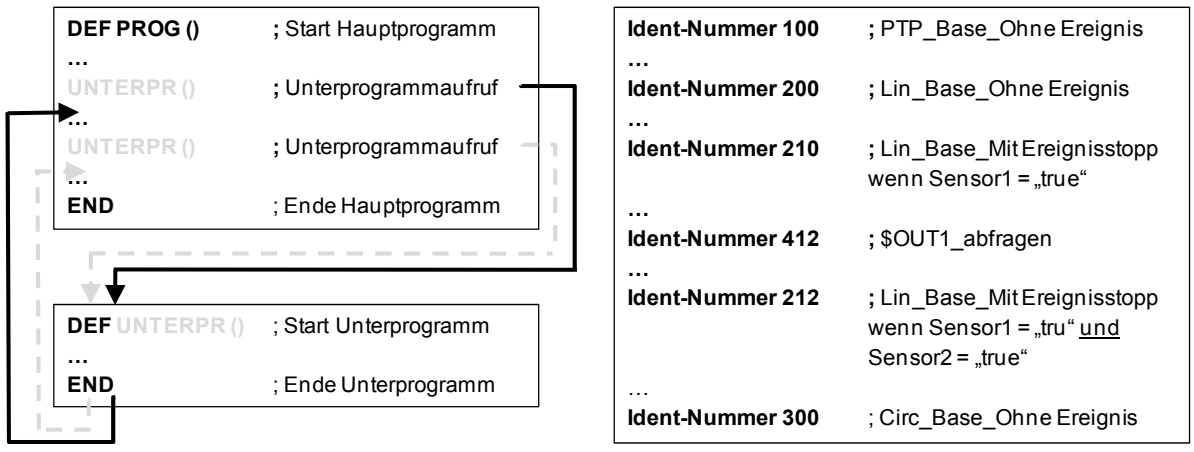
diesen Lösungen gehören beispielsweise Unternehmensnetzwerke oder mögliche Softwareerweiterungsmodule des Systems für einen effektiveren Einsatz bei KMU.

Im letzten Kapitel 8 „Zusammenfassung und Ausblick“ werden die Ergebnisse dieser Arbeit zusammengefasst, es erfolgt eine Gegenüberstellung des Konzeptes mit der prototypischen Realisierung. Weiterhin wird ein Ausblick auf zukünftige Arbeiten gegeben.

5 Konzeption technologieorientierter Programmier- und Steuerungssysteme

5.1 Struktur des Gesamtsystems

Für die Ausführung der Anwenderprogramme von Industrierobotern wird der Ansatz genutzt, bestehende Programme in kleinste generische Einheiten, die sogenannten Elementaranweisungen (ELA), zu unterteilen. Die ELA ergeben erst in ihrer Kombination sowie mit der Verknüpfung entsprechender Steuerungsinformationen ein lauffähiges Anwenderprogramm, siehe **Bild 5.1** [Fri-09]. Dadurch ist es möglich, die Elementaranweisungen mehrfach zu verwenden. Einerseits werden unterschiedliche Anwenderprogramme mit Hilfe einer definierten Anzahl von ELA erzeugt und andererseits wird eine ELA durch die Verknüpfung mit unterschiedlichen Steuerungsfunktionen anpassungsfähig ausgeführt. Aufgrund der Verwendung der Elementaranweisungen entsteht eine Modularisierung und Flexibilisierung der Roboterprogramme, weshalb sie nach dem Baukastenprinzip kombiniert werden [Uhl-08a].



a) Anwenderprogramme ohne ELA (heute) b) Anwenderprogramme mit ELA (zukünftig)

Bild 5.1: Anwenderprogramme

Aufgrund der erarbeiteten Anforderungsliste wird das Gesamtkonzept des technologieorientierten Programmier- und Steuerungssystems aus vier Komponenten zusammengesetzt:

- der Robotersteuerung,
- der Prozesssteuerung,

- dem Datenbanksystem und
- dem Programmierleitstand, vgl. **Bild 5.2**.

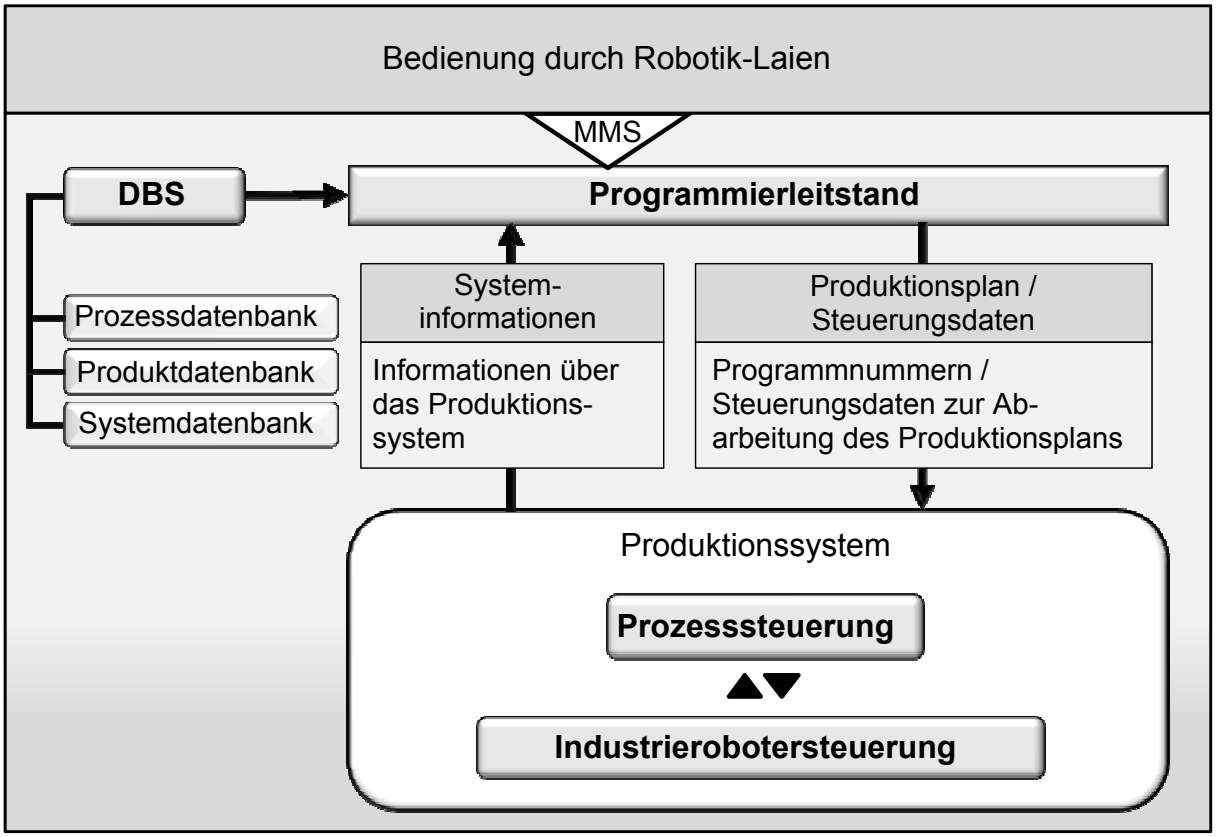


Bild 5.2: Gesamtkonzept des technologieorientierten Programmier- und Steuerungssystems

Die *Robotersteuerung* wird für die Erstellung der Bewegungsprogramme und Überwachung des Industrieroboters genutzt, wobei die Steuerung nur einmalig mit Hilfe der Elementaranweisungen programmiert wird. Das bedeutet, dass eine definierte Anzahl von kleinen, generischen Programmen offline programmiert wird. Durch diese Offlineprogrammierung können die Anwenderprogramme online generiert werden, indem die Elementaranweisungen mit den entsprechenden Steuerungsanweisungen kombiniert werden.

Die *Prozesssteuerung* erfüllt im Konzept des technologieorientierten Programmier- und Steuerungssystems zwei Aufgaben. Einerseits werden von der Prozesssteuerung die benötigten Steuerungsinformationen den Robotern zur Verfügung gestellt, andererseits werden dem Programmierleitstand aufbereitete Informationen über das Produktionssystem übermittelt. Die Prozesssteuerung ist somit die Schnitt-

stelle zwischen Industrierobotern und Produktionssystem sowie dem Programmierleitstand.

Das *Datenbanksystem* hat die Funktion, die notwendigen Daten für das Programmier- und Steuerungssystem elektronisch zu verwalten. Hierbei werden Daten der folgenden Bereiche gespeichert:

- Produkt,
- Roboter und
- System.

Auf Basis dieser Daten werden die Steuerungsinformationen und die Reihenfolge der Elementaranweisungen in Form eines Produktionsplans generiert. Weiterhin werden alle Daten des Programmier- und Steuerungssystems mit Hilfe des Datenbanksystems verwaltet und dargestellt. Die Trennung von Roboterprogrammen und Steuerungsinformationen erhöht die Flexibilität des Gesamtsystems. Beispielsweise werden bei einem Produktwechsel innerhalb einer Fertigungsfamilie nur die Produktdaten ausgetauscht [Uhl-08b].

Aufgrund der bereitgestellten Informationen kann der *Programmierleitstand* eine flexible Steuerung der Industrieroboter im Kontext des Produktionssystems vornehmen. Darüber hinaus können mit dem Programmierleitstand weitere Automatisierungskomponenten wie beispielweise ein Transportsystem mit Hilfe der Prozesssteuerung gesteuert werden. Der Programmierleitstand beinhaltet eine Mensch-Maschine-Schnittstelle, die es dem Endanwender ermöglicht, das System ohne fortgeschrittene Programmierkenntnisse zu bedienen. Weiterhin ist der Programmierleitstand ein zentrales Element des technologieorientierten Programmier- und Steuerungssystems, das über die Einbindung des Datenbanksystems einen funktionsfähigen Produktionsplan als Anweisungsliste erstellt.

5.2 Systemkomponenten

5.2.1 Allgemeines

In den nachfolgenden vier Kapiteln werden die Grundlagen der Systemkomponenten identifiziert sowie die Komponenten für den Einsatz im konzipierten technologieorientierten Programmier- und Steuerungssystem angepasst bzw. neu entwickelt. Das heißt, der exakte Funktionsumfang der Komponenten wird definiert, wobei

einerseits der Umfang reduziert bzw. erweitert wird sowie andererseits Neuentwicklungen präsentiert werden. In der **Tabelle 5.1** sind die Modifikationen der vier Komponenten des zu entwickelnden Systems zusammengefasst.

Tabelle 5.1: Modifikation der Komponenten

Komponente / Modifikation	**Roboter-steuerung**	**Prozess-steuerung**	**Datenbank-system**	**Programmier-leitstand**
Reduzierung des Funktions-umfangs	Programmierauf-wand wird auf ein Minimum reduziert			
Erweiterung des Funktions-umfangs		Verarbeitung des Produktionsplans		
Entwicklungen auf Basis von Standard-technologien			Konzeptionierung der Datenbanken inklusive Daten-bankentwurf	
eigene Entwicklung				Konzeptionierung der Programmier- und Steuerungs-logik sowie einer grafischen Be-nutzeroberfläche

Darüber hinaus werden die Datenbanken des technologieorientieren Programmier- und Steuerungssystems auf Basis der Standardtechnologie „Datenbanksystem“ konzipiert, um die Informationsverarbeitung zu erleichtern.

5.2.2 Robotersteuerung

Die Robotersteuerung wird mit Hilfe der Elementaranweisungen programmiert, wobei diese Frameworks (Programmstrukturen) online mit Steuerungsinformationen zu lauffähigen Anwenderprogrammen verknüpft werden. Die Verwendung der Elementaranweisungen ermöglicht routinierte Entscheidungen.

Bei routinierten Entscheidungen sind die möglichen Alternativen bei jeder Entscheidungswiederholung gleich, endlich sowie bekannt und es wird zwischen ihnen routinemäßig und automatisch gewählt. Dabei erfolgt im Moment der Entscheidungs-

findung ein bloßer Abgleich, der sogenannte Matchingprozess, zwischen den vorgespeicherten Situationen und allen Entscheidungswegen. Wichtig für die Entscheidung ist der Ähnlichkeitsgrad mit bereits bekannten Daten [Jun-05].

Elementaranweisungen sind kleinste, generische Roboterprogramme, die einfachste Roboteranweisungen repräsentieren, wie z. B. das Verfahren des Roboters von Punkt A nach Punkt B oder die Aktivierung eines digitalen Roboterausganges, vgl. **Anhang 1 und Anhang 2**. Als Gemeinsamkeit zeigen alle Elementaranweisungen eine Fehlerquote, die gegen null geht und demnach einen Routinefaktor der 100 % entgegenstrebt. Als Folge der geringen Fehlerquote können sämtliche Produktionsschritte in eine eindeutige Baumstruktur überführt werden, die dann in Algorithmen im Zuge der Programmierung übersetzt wird.

Für die Roboterprogrammierung werden folgende Anweisungsarten definiert [in Anlehnung an Hes-98, Koc-07]:

- Ablaufanweisungen:
  Diese beschreiben den Ablauf eines Programms in Form einfacher Befehle.
- Kommunikationsanweisungen:
  Diese geben Hinweise und Aufforderungen an das Fachpersonal und leiten einen Informationsaustausch mit angeschlossenen Geräten ein.
- Bewegungsanweisungen:
  Diese beschreiben die anzufahrenden Positionen, die Art der Bewegung und enthalten Anweisungen hinsichtlich der Bewegungen des Effektors.
- Kontrollanweisungen:
  Diese beeinflussen den Ablauf des auszuführenden Bewegungsprogramms.
- Diagnoseanweisungen:
  Sie beschreiben das Verhalten des Roboters bei Unterbrechungen und geben Hinweise zur Störungsbehebung.

Die Roboterprogrammierung mit Hilfe des technologieorientierten Programmier- und Steuerungssystems erfordert nur Bewegungs-, Kontroll- und Diagnoseanweisungen. Die Funktionen der Ablauf- und Kommunikationsanweisungen werden von dem Programmierleitstand übernommen, siehe **Kapitel 5.2.4**.

Die Elementar-Bewegungsanweisungen basieren auf den drei Bewegungsbefehlen einer Robotersteuerung:

- PTP-Anweisung:
  Die Positionierung des Robotersystems erfolgt hier auf dem schnellsten Weg zwischen zwei Punkten.
- LIN-Anweisung:
  Bei einer linearen Bewegung werden die Roboterachsen so aufeinander abgestimmt, dass der Werkzeug- bzw. Werkstückbezugspunkt entlang einer Geraden zum Zielpunkt bewegt wird.
- CIRC-Anweisung, vgl. **Bild 5.3**:
  Hier bewegt sich der Bezugspunkt des Werkzeugs bzw. Werkstücks auf einem Kreisbogen zum Zielpunkt.

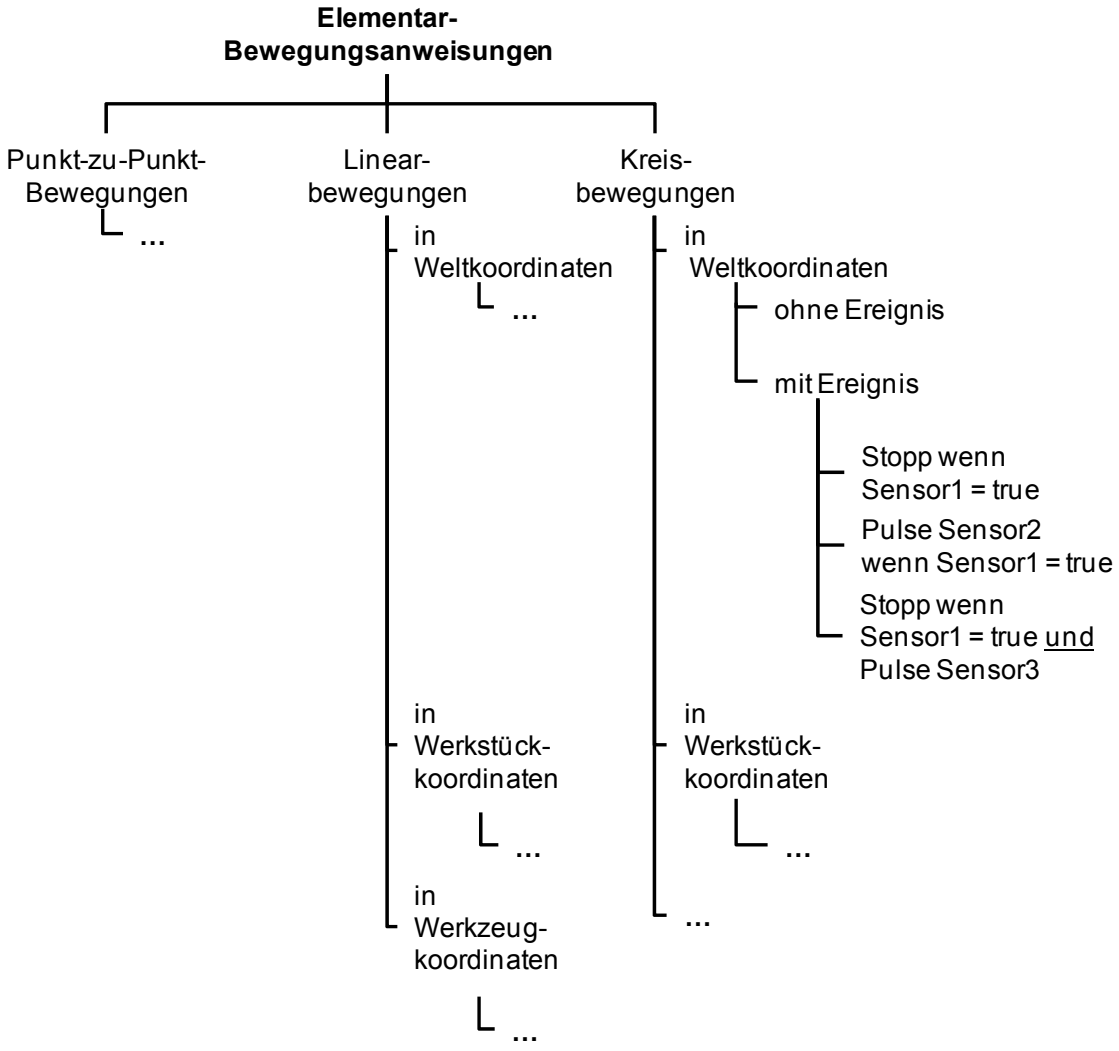


Bild 5.3: Elementar-Bewegungsanweisungen des TOPS

Weiterhin können diese Bewegungsanweisungen in verschiedenen Bezugskoordinatensystemen ausgeführt werden, wobei sich in der Praxis drei rechtwinklige Koordinatensysteme herauskristallisiert haben, um einen Industrieroboter zu steuern. Das Welt-Koordinatensystem befindet sich ortsfest im Ursprung des Roboterfußes; das Werkstück-Koordinatensystem hat seinen Ursprung am zu bearbeitenden Werkstück; beim Werkzeug-Koordinatensystem liegt der Ursprung im Werkzeug.

Während einer Bewegungsanweisung können zeitgleich Sensoren überwacht und/oder gesteuert werden, um sich flexibel an neue Anforderungen und Umgebungsbedingungen anzupassen. Die Kommunikation zwischen Robotersteuerung und Umwelt wird mit Hilfe der Sensoren realisiert, beispielsweise soll ein abstandsmessender Sensor die Roboterbewegung bei Erreichen einer bestimmten Distanz stoppen.

Kontrollanweisungen sind Kontrollstrukturen, welche die Abfolge der Anweisungen bestimmen. Aufgrund der vereinfachten Robotersteuerungsprogrammierung werden nur die Kontrollanweisungen „WAIT“ (Verweilzeit) und „OUT“ (Setzen von Ausgängen) erforderlich. Die darüber hinaus erforderlichen Kontrollanweisungen, wie beispielsweise bedingte Anweisungen („If-Then-Anweisung“) oder Sprunganweisungen („Go-To-Anweisung“), werden von dem Programmierleitstand übernommen, siehe **Bild 5.4.**

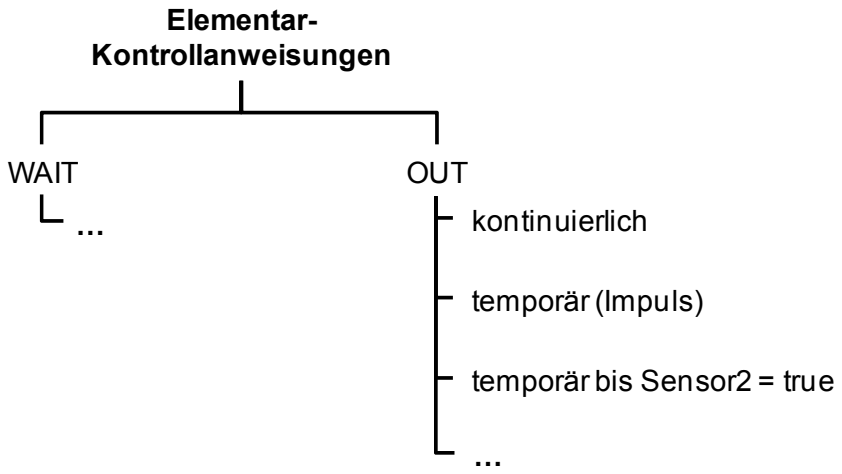


Bild 5.4: Elementar-Kontrollanweisungen

Die Diagnose-Elementaranweisungen werden benötigt, um Statusmeldungen der Robotersteuerung zu ermitteln, vgl. **Bild 5.5**. Hierbei ist eine Vielzahl von Anweisungen denkbar, wobei nur die Abfrage der Ein- und Ausgänge elementar ist.

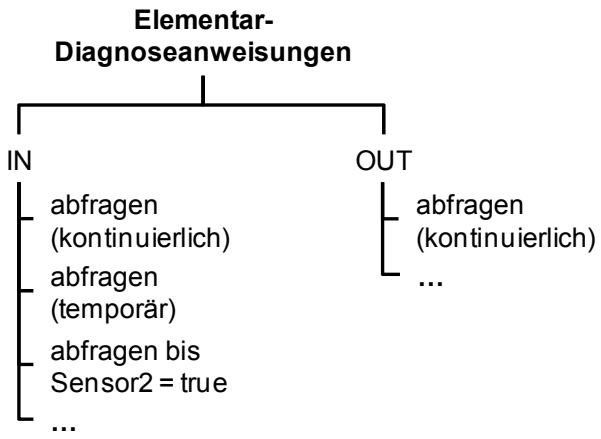


Bild 5.5: Elementar-Diagnoseanweisungen

Um die Trennung zwischen Funktionen und spezifischen Roboteranweisungen zu erzielen, werden alle Elementaranweisungen mit Hilfe von Identifikationsnummern (Ident-Nummer) als eindeutigem Schlüssel aufgerufen, siehe **Bild 5.6**. Dabei wird die Unabhängigkeit des technologieorientierten Programmier- und Steuerungssystems von der Robotersteuerung sichergestellt. Darüber hinaus wird die Integration unterschiedlicher Robotersteuerungen im TOPS ermöglicht. Dafür müssen jeweils die Elementaranweisungen auf den Steuerungen programmiert werden.

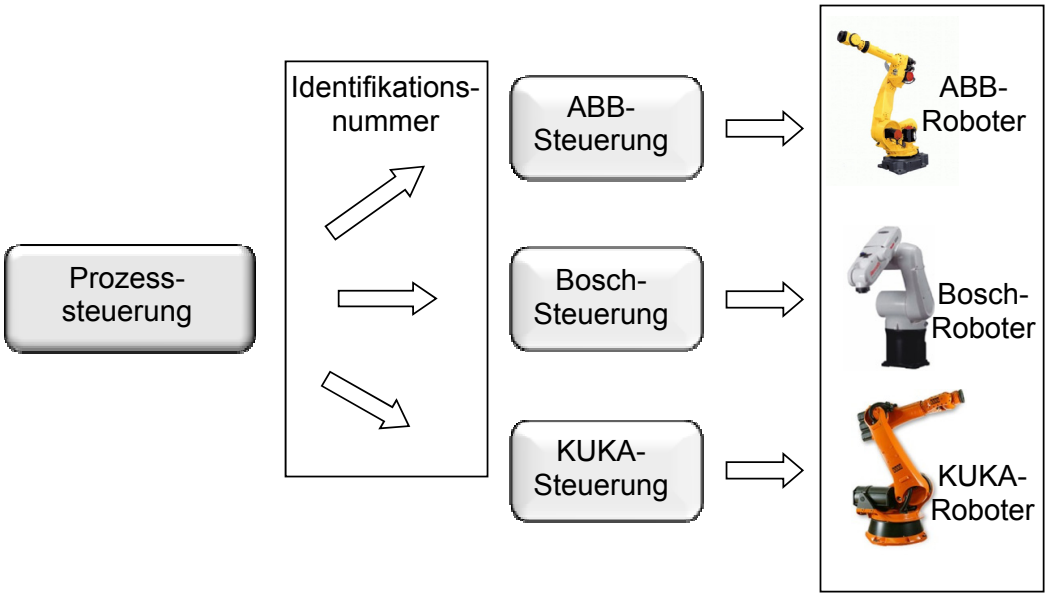


Bild 5.6: Verwendung der Identifikationsnummer

5.2.3 Prozesssteuerung

In der Regel werden Industrieroboter oder Werkzeugmaschinen über die Prozesssteuerungen gesteuert und überwacht. Wichtige Aufgabe der Prozesssteuerung ist es, den Prozess bzw. die Anlagen so zu steuern, dass ein größtmöglicher autonomer Ablauf gewährleistet ist. Dabei sollen die notwendigen Funktions- und Sicherheitsanforderungen erfüllt werden [Wör-06].

Um die geforderten Steuerungsfunktionalitäten in das technologieorientierte Programmier- und Steuerungssystem zu integrieren, wird die Prozesssteuerung in das Konzept einbezogen. Hierbei obliegen der Prozesssteuerung hinsichtlich des Gesamtsystems zwei Aufgaben:

1. Verarbeitung des Produktionsplans (Steuerung der Automatisierungskomponenten) und
2. Übermittlung von Steuerungsinformationen an den Programmierleitstand.

Darüber hinaus werden der Prozesssteuerung keine weiteren Funktionalitäten übertragen oder entzogen. Sie lenkt und regelt die automatischen Funktionsabläufe der Fertigungsanlage, Transportmittel und Maschinen. Der Programmierleitstand übermittelt an die Prozesssteuerung den kompletten Produktionsplan (Kombination von Elementaranweisungen) mit den dazugehörigen Steuerungsinformationen. Dagegen überträgt die Prozesssteuerung die Elementarüberweisungen sequenziell an die Roboter und das Produktionssystem. Die Synchronisation der Teilnehmer wird über Synchronisationsmarken (Rendezvous-Punkte) vorgenommen, da bei bestimmten Fertigungsaufgaben mitunter der Gleichlauf zwischen mehreren Teilnehmern sichergestellt werden muss und die Bearbeitung vom Fortschritt des jeweiligen Teilnehmers abhängig ist. Durch eine Synchronisationsmarke wird ein Zeitpunkt definiert, an dem sich mehrere Teilnehmer synchronisieren.

Sollen beispielsweise durch eine Synchronisationsmarke zwei Komponenten synchronisiert werden, müssen in den Produktionsplänen der zwei Teilnehmer an der gewünschten Stelle die identischen Marken stehen. Um eine Synchronisationsmarke zu identifizieren, muss diese im Produktionsplan mit einer definierten Identifikationsnummer gekennzeichnet werden. Darüber hinaus wird diese Identifikationsnummer mit einer Synchronisationsnummer kombiniert, da es bei Synchronisationsmarken während eines Prozesses nicht zu Verwechslungen kommen darf. Weiterhin wird ein

weiterer Parameter (Synchronisationsteilnehmer) verwendet, damit erkennbar ist, wie viele Teilnehmer an dieser Synchronisation beteiligt sind.

Um eine bestmögliche Programmierung der Industrieroboter durchzuführen, muss die Prozesssteuerung aufbereitete Informationen über das Produktionssystem an den Programmierleitstand übermitteln, wobei die übergeordnete Steuerung des Produktionssystems durch den Programmierleitstand (PLS) ermöglicht wird. Weiterhin stellt die Prozesssteuerung Informationen bereit, die es dem PLS ermöglichen, einen Produktionsplan zu generieren, der den Ausgangszustand des Produktionssystems berücksichtigt. Darüber hinaus wird das PLS befähigt, beim Eintreten einer Fehlersituation die Situation zu analysieren und entsprechende Korrekturmaßnahmen zu initiieren. Hierzu kann der PLS den aktuellen Zustand über die Ausführung des Produktionsplans sowie den aktuellen Zustand des gesamten Produktionssystems abfragen. Diese Informationen sowie die Fehlermeldung werden interpretiert und mit dem Sollzustand verglichen.

5.2.4 Datenbanksystem

Da die Verwaltung von Informationen und der Zugriff auf Informationen eine wichtige Rolle im technologieorientierten Programmier- und Steuerungssystem darstellen, kann auf den Einsatz eines Datenbanksystems für die Informationsverarbeitung nicht verzichtet werden. Ein Datenbanksystem ist ein akzeptiertes und eingeführtes Hilfsmittel zur effizienten, rechnergestützten Organisation, Erzeugung, Manipulation und Verwaltung großer Datensammlungen [Kud-07]. Ein Datenbanksystem (DBS) besteht aus einem Datenbankmanagementsystem (DBMS) und einer gewissen Anzahl von Datenbanken (DB). Eine Datenbank ist eine Sammlung von Daten, die Informationen repräsentiert und miteinander in Beziehung setzt [Kem-04, Vos-00].

Hierbei ist die Informationsmodellierung die grundlegende Technik der Transformation des gesamten Produktionssystems inklusive aller Automatisierungskomponenten in eine formale Repräsentation, die dann zum Datenbankentwurf benutzt wird. Die beschreibenden Modelle beinhalten dann die Angabe der Objekte und die Beziehungen zwischen den Objekten des Produktionssystems, wobei als Beschreibungsformalismen das Entity-Relationship-Modell oder die Unified Modeling Language (UML) verwendet werden [Kud-07].

Das DBS enthält Informationen über das Produkt, die Produktionssequenz und das Produktionssystem, die in folgenden Datenbanken gespeichert werden:

- in der Produktdatenbank,
- der Prozessdatenbank und
- der Systemdatenbank.

In der *Produktdatenbank* (ProdDB) sind Informationen über den konstruktiven Aufbau und die technischen Eigenschaften des Produkts dokumentiert, das Produkt wird mit seinen Baugruppen und Bauteilen beschrieben. Darüber hinaus gehören zu den Produktinformationen Steuerungsinformationen für die Industrieroboter, die produktabhängig sind. Dazu gehören beispielsweise die geometrische Beschreibung des Produkts oder die Objektbasis eines Bauteils, die die Lage im Raum beschreibt.

Die Beschreibung der Industrieroboter wird in der *Prozessdatenbank* (ProzDB) vorgenommen. Hierbei werden einerseits die Grundeigenschaften eines Industrieroboters, wie z. B. der Arbeitsraum, und andererseits systemimmanente Eigenschaften definiert. Zu systemimmanenten Eigenschaften gehören beispielsweise:

- welche Roboter mit welchen Werkzeugen arbeiten oder
- welche Roboter an welchen Arbeitsstationen arbeiten.

Darüber hinaus enthält die Prozessdatenbank Informationen zu den einzelnen Produktionssequenzen bzw. den Elementaranweisungen, die für eine Realisierung der Produktionsschritte benötigt werden. Das heißt, es werden die Fähigkeiten des Produktionssystems beschrieben, Produktionsschritte in Abhängigkeit der definierten Komponenten der Produktdatenbank auszuführen. Im Speziellen werden die Abläufe der Produktionsschritte gespeichert und durch deren Abbild z. B. die Verknüpfung mit einem Werkzeug definiert.

In der *Systemdatenbank* (SysDB) wird das Produktionssystem abgebildet. Das heißt, es werden alle Informationen, die das Produktionssystem beschreiben, in einer Datenbank gespeichert. Es werden mit Hilfe der Datenbankstrukturen die Komponenten des Produktionssystems und deren Eigenschaften wie beispielsweise die Industrieroboter, Arbeitsstationen oder das Transportsystem beschrieben sowie mit Parametereinträgen wie z. B. vorhandenen Werkzeugen, Arbeitsräumen und Koordinaten verknüpft. Darüber hinaus werden in der Systemdatenbank die Ausgangssituationen für die Produktionsschritte, die in der Systemdatenbank definiert sind,

charakterisiert. Für die Datenpflege ist es notwendig, eine Schnittstelle zwischen dem DBS und dem Bediener des Programmier- und Steuerungssystems bereitzustellen. Der Anwender muss hierbei über die Darstellung der Komponenten des DBS und deren Daten verfügen. Durch diese Kommunikationsschnittstelle wird eine komplette Administration ermöglicht, so dass eigenständige Einträge hinzugefügt, editiert und gelöscht werden können.

5.2.5 Programmierleitstand

Die Verwendung grafischer Benutzeroberflächen für die Mensch-Computer-Interaktion ist in den letzten zwei Jahrzehnten für nahezu alle Computerbenutzer zu einer Selbstverständlichkeit geworden. Daher bildet eine grafische Benutzeroberfläche eine wichtige Komponente des Programmierleitstands und somit einen zentralen Bestandteil des technologieorientierten Programmier- und Steuerungssystems. Um eine flexible Einsetzbarkeit und Erweiterbarkeit des Programmierleitstands zu gewährleisten, wird eine modularisierte und funktionsorientierte Aufteilung der Programmeigenschaften auf verschiedene Komponenten vorgesehen, wobei die Komponenten in die drei Bereiche

- Kernkomponenten,
- System-Plug-ins und
- Erweiterungs-Plug-ins

untergliedert sind, siehe **Bild 5.7**.

Des Weiteren wird die bereichsübergreifende Kommunikation mit Hilfe des Plug-in-Managers realisiert. Weiterhin stellt der Plug-in-Manager durch seine Integration in den Programmierleitstand eine erweiterte Verwendungsmöglichkeit unterschiedlicher Komponenten zur Verfügung. Darüber hinaus ermöglicht der Plug-in-Manager das Konzept des Programmierleitstands, zukünftig bei weiterführenden Anwendungsfällen zu benutzen. Damit lassen sich für eine spätere Verwendung einfach und unkompliziert neue Programmkomponenten hinzufügen, um den Funktionsumfang beliebig erweitern zu können und somit die Software für neue Produkte, Prozesse oder Systeme anwendbar zu gestalten.

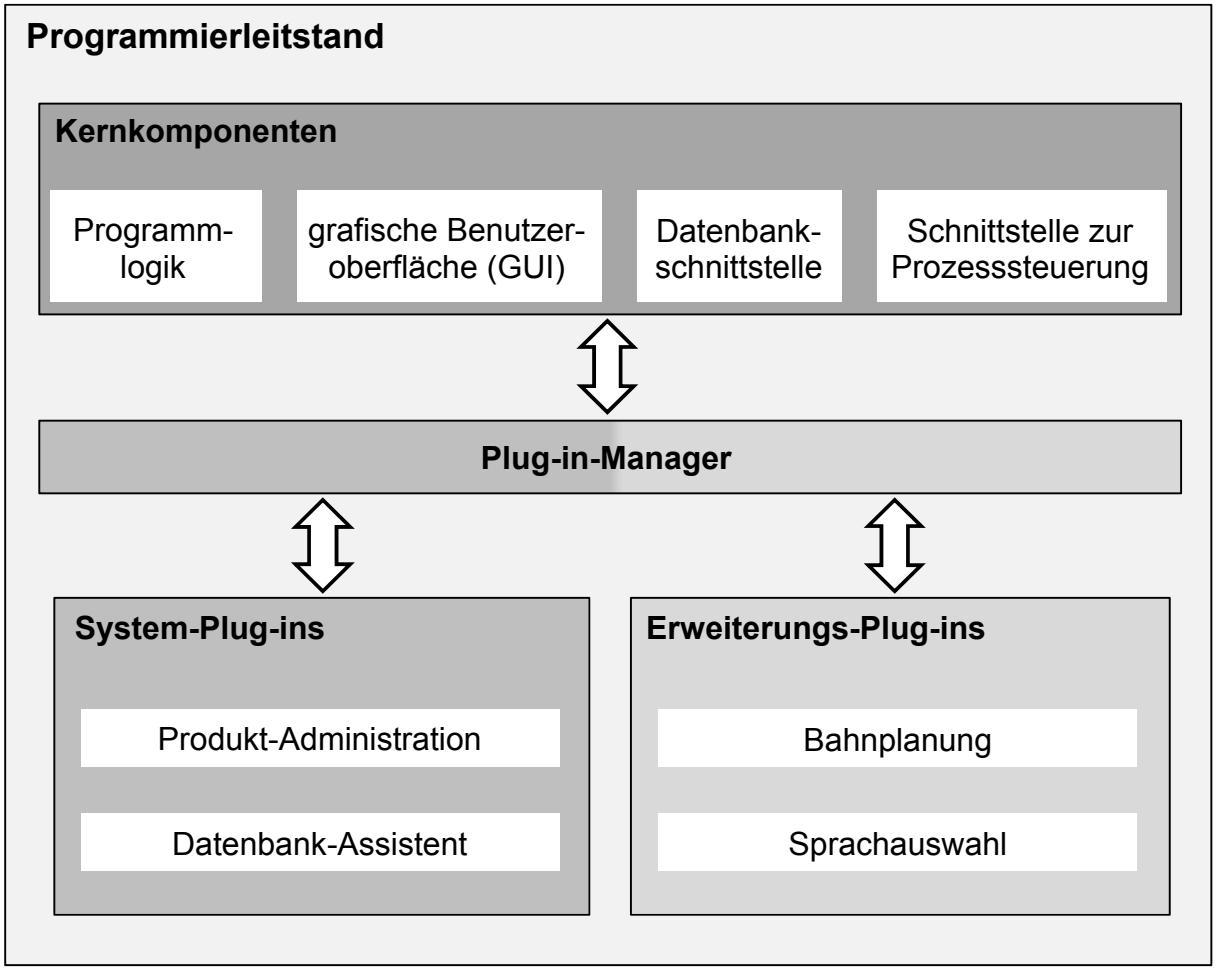


Bild 5.7: Konzept des Programmierleitstands

Der Plug-in-Manager repräsentiert das Plug-in-System im Konzept des Programmierleitstands. Die Basis für die Konzeptionierung eines solchen Systems ist die Abstraktion und Vereinheitlichung der Funktionalität potenzieller Plug-ins, wobei es entscheidend ist, die Schnittstelle zwischen einem Plug-in und den Kernkomponenten abzubilden. Für eine benutzerfreundliche Integration vorhandener Plug-ins ist eine intuitive Umsetzung der Plug-in-Schnittstelle unabdingbar. Um dies zu erreichen, wird analog zum Standard der selbstkonfigurierenden Hardware (Plug-and-Play-Prinzip) ein Prinzip genutzt, das automatisch alle innerhalb eines definierten Verzeichnisses vorhandenen Plug-ins erkennt und in die Applikation integriert. Plug-ins müssen dafür in selbstständigen Programmpaketen unter strenger Einhaltung der für das System maßgebenden Konventionen organisiert werden. Die Definition dieser Konventionen wird dabei über eine Basis realisiert, welche die Grundlage für jedes Plug-in bildet. Die Aufgabe der Plug-in-Basis ist die gezielte Abgrenzung und Eingrenzung der für die Kernkomponenten relevanten Plug-in-

Informationen. Hierbei ermöglicht die Definition dieser Kommunikationsgrundlage zwischen einem Plug-in und dem Programmierleitstand eine Verarbeitung des Erweiterungsmoduls seitens des Programms sowie die erfolgreiche Integration.

Im Bereich „Kernkomponenten“ werden die zentralen Komponenten des Programmierleitstands zusammengefasst, die für eine Integration des Programmierleitstands in das Gesamtkonzept des Programmier- und Steuerungssystems zwingend notwendig sind. Dieser Bereich des Programmierleitstands umfasst die Komponenten

- Programmlogik,
- grafische Benutzeroberfläche,
- Datenbankschnittstelle und
- Schnittstelle zur Prozesssteuerung.

Die Komponente *Programmlogik* definiert den Funktionsumfang des Programmierleitstands. Deren Aufgabe ist es, alle logischen Operationen durchzuführen. Dazu gehören die Verarbeitung und Prüfung von Benutzereingaben, das Durchführen von Berechnungen sowie das Senden und Empfangen von Daten anderer Komponenten des Produktionssystems. Des Weiteren stellt die Programmlogik den Befehlsumfang bereit, d. h., die Grundoperationen des Produktionssystems werden als Bausteine für die Verwendung durch den Benutzer bereitgestellt. Darüber hinaus können durch den Anwender zusätzliche Operationen definiert werden, um den Basisbefehlsumfang individuell zu erweitern. Für einen sicheren und stabilen Programmablauf ist es zwingend erforderlich, die Kommunikation mit Hilfe der Mensch-Maschine-Schnittstelle zu den verschiedensten Zeitpunkten innerhalb des Programmablaufs auf ihre Gültigkeit und Konsistenz zu prüfen. Eine besondere Rolle spielt dabei die Validierung der vom Benutzer getätigten Eingaben bezüglich bestimmter Formatvorgaben, Bereichsgrenzen oder weiterer Zwangsbedingungen. Bei Verletzung dieser Bedingungen ist der Benutzer durch die grafische Benutzeroberfläche (GUI) auf die Fehleingabe hinzuweisen. Nach erfolgreicher Datenvalidierung folgt die Weiterverarbeitung der geprüften Daten. Als Weiterverarbeitung kommen hier neben der Datensicherung in dem Datenbanksystem auch alle sonstigen Programmabläufe in Frage, die diese Daten verwenden. Dazu gehören unter anderem auch die Aufbereitung und das Versenden der Daten an die Prozesssteuerung. Darüber hinaus werden für die benutzerfreundliche Darstellung der Produktionsabläufe die erforder-

lichen bzw. gewünschten Daten aus dem Datenbanksystem geladen und durch Überführung in grafische Oberflächenkomponenten für den Anwender aufbereitet. Die Aufgabe der *grafischen Benutzeroberfläche* ist es, die von der Programmlogik zur Verfügung gestellten Informationen für den Benutzer optisch aufzubereiten und darzustellen, sowie umgekehrt die Eingaben des Benutzers an die Programmlogik zurückzugeben. Für die Realisierung der grafischen Benutzeroberfläche soll die Strategie einer engen Verknüpfung der Oberfläche mit der Programmlogik verfolgt werden, um eine oberflächen- und benutzernahe Datenbehandlung zu erreichen. Ein Datenbanksystem verwaltet alle Informationen des technologieorientierten Programmier- und Steuerungssystems, wobei für die Kommunikation mit dem Datenbanksystem im Programmierleitstand eine *Datenbankschnittstelle* vorgesehen ist. Durch die Trennung des Zugriffs auf diese Ressourcen vom Rest der Anwendung lassen sich hier problemlos und unabhängig zu jedem späteren Zeitpunkt Änderungen vornehmen. Es ist auf diese Weise auch ein vollständiger Austausch der Datengrundlage (z. B. Wechsel des Datenbanksystems) ohne Komplikationen möglich. Die *Schnittstelle zur Prozesssteuerung* ermöglicht die Kommunikation mit der Steuerung eines Produktionssystems. Über diese Schnittstelle können sowohl Systemzustände erfragt als auch der Zustand des Gesamtsystems in einen anderen Zustand überführt werden. Für den Anwender werden die Systemgrößen in einer grafischen Systemgesamtansicht und in tabellarischer Form mit Hilfe der grafischen Benutzeroberfläche dargestellt, wobei damit die Überwachung des Systems zu jedem Zeitpunkt vor und während des Produktionsablaufes ermöglicht wird. Damit stellt sie die zentrale Übertragungsfunktion für die von den Benutzern gestalteten Prozessabläufe dar.

Komponenten, die eine Erweiterung des Komforts der Anwendung darstellen, sind im Bereich „System-Plug-ins“ zusammengefasst. Die Komponente *Produkt-Administration* dient der Darstellung und Konfiguration des für die Anwendung relevanten Produkts. Hier können neue Produkte sowie die für diese notwendigen Produktionsabläufe definiert werden. In einem Flussdiagramm werden die einzelnen Produktionsschritte in ihrer zeitlichen Abfolge dargestellt und für die Modifikation seitens des Benutzers aufbereitet. Die Modifikationen bieten die Möglichkeit, vorhandene Konfigurationen in allen Prozessparametern zu variieren, um diese für die sofortige Produktion zu verwenden oder den vollständigen Produktionsablauf für die

spätere Verwendung in der Datenbank zu speichern. Des Weiteren wird eine Administration des Produktionssystems und aller darin enthaltener Komponenten wie beispielsweise

- die Arbeitsstationen,
- die Roboter,
- die Werkzeuge oder
- die Transportmittel

ermöglicht. Der *Datenbank-Assistent* stellt eine Erweiterung des Komforts der Komponente Datenbankschnittstelle dar. Durch diese Erweiterungen können die Einstellungen der Verbindung zum Datenbankmanagementsystem angepasst werden. Darüber hinaus ist eine Initialisierungs- sowie eine Back-up- und Restore-Funktion vorgesehen, die vom Benutzer gewählte Datenbank komfortabel verwalten kann.

Durch eine zusätzliche Integration von optionalen Plug-ins kann der Programmierleitstand in seinem Funktionsumfang beliebig erweitert werden, wobei diese Komponenten im Bereich „Erweiterungs-Plug-ins“ verwaltet werden. Eine Möglichkeit einer solchen Erweiterung stellt die Integration einer *Bahn- oder Greifplanung* für die gezielte Anpassung der Roboterbewegung dar, siehe **Kapitel 7.4**. Die Komponente *Sprachauswahl* ermöglicht z. B. ein sofortiges Umschalten der Applikation auf eine andere installierte Sprache, so dass der Anwender seine favorisierte Sprache benutzen kann. Darüber hinaus lassen sich hier neue Sprachen über gegebene Sprachdateien in die Anwendung integrieren.

5.2.6 Schnittstellen

Durch die verschiedenartigen Hard- und Softwarekomponenten, die verwendet werden, muss der Informationsaustausch mit Hilfe von Schnittstellen realisiert werden. Eine Schnittstelle kann als Teil eines Systems definiert werden, der dem Austausch von Informationen mit anderen Systemen dient [Pri-04]. Bei den informationstechnischen Komponenten, die in diesem Konzept benutzt werden, liegen die Informationen in unterschiedlichen Datenformaten vor. Damit ein Informationsaustausch ermöglicht werden kann, werden vier Informationsflüsse betrachtet. Das entspricht drei internen Schnittstellen zwischen den Komponenten des Systems, die in **Bild 5.8** dargestellt sind, und einer externen Benutzerschnittstelle zur Bedienung des Programmierleitstands.

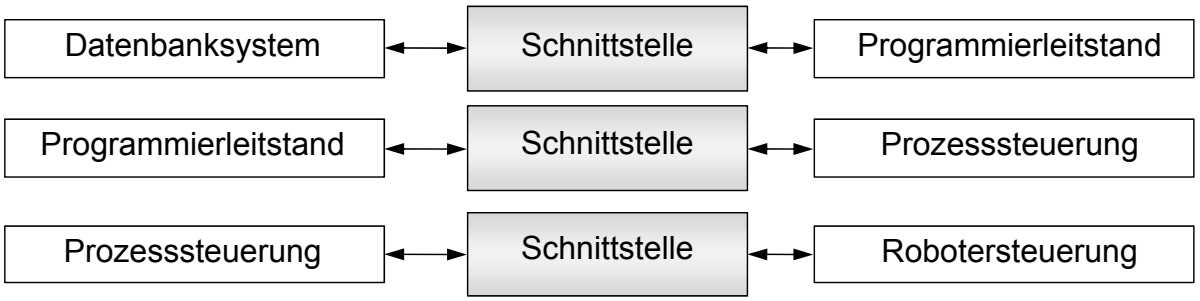


Bild 5.8: Interne Schnittstellen des technologieorientierten Programmier- und Steuerungssystems

5.2.6.1 Interne Schnittstellen

Die Schnittstelle zwischen dem Programmierleitstand und dem Datenbanksystem ist eine Datenbankschnittstelle, welche die Kommunikation zwischen einer Softwareapplikation und einem Datenbanksystem ermöglicht. Durch eine definierte Datenbankschnittstelle können die Speicherung und der Zugriff auf die Datensätze ohne das Wissen der zur Verwaltung und Speicherung verwendeten Datenstrukturen in der Datenbank erfolgen, wobei datenunabhängige Schnittstellen wie beispielsweise ODBC (Open Database Connectivity) oder JDBC (Java Database Connectivity) und datenbankspezifische Schnittstellen wie beispielsweise OCI (Oracle Call Interface) existieren [Kud-07].

In der industriellen Kommunikation haben sich auf der Zellenebene aufwärts das Ethernet und TCP/IP als Standard für Computernetzwerke etabliert [Wec-06]. Für die Schnittstelle zwischen Programmierleitstand und Prozesssteuerung können verschiedene Lösungen für die Kommunikation über Ethernet eingesetzt werden, die einen Datenaustausch von verschiedenen Komponenten mit Hilfe standardisierter Applikationen ermöglichen [Zac-00]. Hierbei haben sich OPC (Openness, Productivity, Collaboration) und XML (Extensible Markup Language) als Standard für den vertikalen Datenaustausch in der Automatisierungstechnik durchgesetzt [Sei-08]. Ziel dieser Standards ist es, herstellerübergreifende Systemschnittstellen zur Verfügung zu stellen, die gezielt für die industrielle Steuerungs- und Automatisierungstechnik erarbeitet wurden [Ens-07].

Für die Kommunikation zwischen Roboter- und Prozesssteuerung werden Feldbussysteme eingesetzt. Feldbussysteme sind seriell arbeitende Bussysteme, die der

automatisierten Datenübertragung im Feldbereich dienen und dabei folgende Vorteile besitzen:

- Verringerung des Verkabelungsaufwands,
- einfache Erweiterbarkeit,
- verbesserte Störsicherheit und
- schnelle Fehlerlokalisierung bei Störfällen [Arz-02].

Innerhalb des Anwendungsbereiches von Feldbussystemen kann eine Einteilung nach Einsatzgebieten getroffen werden, die unterschiedliche Anforderungen bezüglich Datenvolumen, Echtzeiteigenschaften und akzeptablem Kostenniveau stellen [Pri-05]. Unter anderem haben sich die Feldbussysteme Device Net, Ethernet, Foundation Fieldbus, Modbus und Profibus weltweit etabliert [Sin-01]. Das international gängigste Feldbussystem ist Profibus mit besonders starker Dominanz in Europa. Hierbei handelt es sich um ein offenes digitales Kommunikationssystem, das für schnelle, zeitkritische und komplexe Kommunikationsaufgaben geeignet ist [Pri-05].

5.2.6.2 Externe Benutzerschnittstelle

Die Interaktion von Mensch und Maschine hat sich in den letzten Jahren aufgrund des technischen Fortschritts stark gewandelt. Die Bedienung und Konfiguration von Maschinen ist zu einer alltäglichen Aufgabe geworden. Grundlage für die Bedienung einer Maschine durch den Menschen ist stets eine Benutzerschnittstelle, welche die Steuerung der Maschine ermöglicht.

Unabhängig von den auszuführenden Interaktionen müssen grundlegende Gestaltungsregeln beachtet werden, um die gewünschte Akzeptanz beim Benutzer zu erreichen. Diese Gestaltungsregeln werden im Folgenden genauer vorgestellt, da sie bei der Entwicklung und Realisierung des Programmierleitstands angewendet werden [in Anlehnung an Her-05, Kam-00, Züh-04]. Es werden dabei softwareorientierte Betrachtungsweisen im Vordergrund stehen, da der PLS eine softwarebasierte Benutzerschnittstelle ist.

Geeignete Aufgaben- und Funktionsteilung

Insbesondere im Bereich der softwarebasierten Mensch-Maschine-Schnittstelle steht im Vordergrund, dem Benutzer durch geeignete Aufgaben- und Funktionsteilung die

Arbeit zu erleichtern und ansprechend zu gestalten. Der Programmierleitstand muss wiederholende, monotone oder für den Menschen zu aufwendige Aufgaben automatisieren, um dem Benutzer ein interessantes und effizientes Arbeiten zu ermöglichen. Darüber hinaus sollte der Benutzer dessen ungeachtet stets die Kontrolle über die Ergebnisse besitzen und immer ausreichend Eingriffs- und Kontrollmöglichkeiten in den Produktionsablauf haben.

Verwendung von Standards

Um beim Benutzer einer Software und somit einer Maschine größtmögliche Akzeptanz zu erreichen, ist es erforderlich, auf Standards zurückzugreifen. Es werden sowohl Standardisierungen aufgrund von Normen als auch inoffizielle Standards, die durch einen hohen Verbreitungsgrad ihre Gültigkeit haben, berücksichtigt, beispielsweise die Farbe „Rot“ für Notausschalter oder das Icon „Diskette“ als Symbol für die Funktion „Speichern“. Dadurch wird dem Benutzer ein schneller Ein- bzw. Umstieg auf die neue Benutzerschnittstelle ermöglicht.

Verwendung optischer Hilfsmittel

Es ist im Allgemeinen bekannt, dass grafische Informationen von Menschen wesentlich schneller verarbeitet werden, als dies bei Texten oder Zahlen der Fall ist. Aus diesem Grund lassen sich bei der Gestaltung von Schnittstellen zwischen Mensch und Maschine einfache Hilfsmittel effektiv nutzen, um dem Benutzer das Arbeiten erheblich zu erleichtern, so z. B. die Verwendung von analogen Anzeigen. Digitale Anzeigen bieten den Vorteil, bei großen Messbereichen größere Anzeigegenauigkeiten darzustellen. Diese stellen oft nicht die bestmögliche Wahrnehmung durch den Benutzer dar, da dieser für die Interpretation der angezeigten Werte Kenntnisse über Grenzbereiche haben muss. Bei der Verwendung analoger Anzeigen hingegen ist für den Benutzer anhand eines Zeigerausschlags und entsprechender farblicher Kennzeichnung ohne weiteres Wissen sofort ersichtlich, ob die angezeigten Werte kritisch sind. Ein genauer Messwert ist dafür oft nicht erforderlich und kann bei erforderlicher hoher Genauigkeit gegebenenfalls über eine zusätzliche digitale Anzeige präsentiert werden.

Reduzierung auf das Notwendigste

Um eine schnelle Bedienung einer Benutzerschnittstelle zu gewährleisten, ist es neben der Verwendung von Standards und von optischen Hilfsmitteln ebenfalls erforderlich, die für den Benutzer sichtbaren Informationen auf ein übersichtliches Maß zu reduzieren. Auf das Wesentliche reduzierte Informationen können die Zeit, die notwendig ist, um sich innerhalb der Mensch-Maschine-Schnittstelle zu orientieren, auf ein Minimum reduzieren. Innerhalb von Softwareanwendungen werden z. B. ausklappbare Menüs verwendet, die somit übersichtlich sind. Um unnötig langes Suchen zu vermeiden, sollte die Menütiefe auf einen Richtwert von zwei Ebenen unterhalb des eigentlichen Menüpunktes der Menüleiste begrenzt werden.

Überprüfungsmöglichkeiten

Bei der Auslegung von Mensch-Maschine-Schnittstellen ist es stets notwendig, eine hohe Fehlertoleranz bezüglich der vom Benutzer getätigten Eingaben vorzusehen, um so fehlerhafte Eingaben frühzeitig zu erkennen und zu korrigieren. Darüber hinaus sollte bei durchgeführten Prüfungen stets darauf geachtet werden, dass dem Benutzer genügend Freiräume gelassen werden und sich dieser nicht von der Maschine bevormundet fühlt.

5.3 Inbetriebnahme des technologieorientierten Programmier- und Steuerungssystems

Die Inbetriebnahme ist die erstmalig bestimmungsgemäße Verwendung einer Maschine in der Anwendergemeinschaft und damit die erste Benutzung durch einen Endanwender [EG-2006]. In diesem Kapitel wird in Anlehnung an diese Richtlinie die Inbetriebnahme des Programmier- und Steuerungssystems beschrieben. Der Leitfaden für die Inbetriebnahme basiert auf der Ausgangssituation, bei der das technologieorientierte Programmier- und Steuerungssystem bei einem Anwender nicht vorhanden ist. Hierbei werden die notwendigen Schritte und Voraussetzungen aufgezeigt, wobei die Abfolge der Schritte und der Tätigkeiten nach den entsprechenden Prioritäten im Folgenden dargestellt wird.

Technologieorientierte Programmier- und Steuerungssysteme sind dafür konzipiert, die Produktion von Fertigungsfamilien mit Hilfe von Industrierobotern zu unterstützen. Aus diesem Grund wird der Programmierleitstand (PLS) generisch entwickelt. Die

Anpassung des Programmier- und Steuerungssystems an ein festgelegtes Produkt wird durch das Datenbanksystem (DBS) ermöglicht. Die Entwicklung eines Programmierleitstands beinhaltet die Programmier- und Steuerungsfunktionalitäten aller Produkte einer Fertigungsfamilie. Das bedeutet, dass vor der Realisierung des PLS alle Arbeitsgänge einer Fertigungsfamilie zusammengefasst sowie mit Hilfe des PLS programmiert und gesteuert werden.

Durch das Datenbanksystem wird das Programmier- und Steuerungssystem an ein Produkt angepasst, da in der Produkt- und Prozessdatenbank die Programmier- und Steuerungsfunktionalitäten für ein festgelegtes Produkt gespeichert sind. Die Anpassung des Systems wird durch den Austausch der Datenbanken realisiert, das heißt, im einfachsten Fall wird die Datenbank eines Produkts durch die Datenbank eines anderen Produkts ersetzt.

Nachdem die Komponenten Programmierleitstand und Datenbanksystem entwickelt und installiert wurden, wird die Anpassung der Prozess- und der Robotersteuerung vorgenommen. Um die Steuerung des Produktionssystems durch das technologieorientierte Programmier- und Steuerungssystem zu ermöglichen, muss die Prozesssteuerung an das neue System angepasst werden. Hierbei steht die Erstellung der Kommunikationsstruktur innerhalb des neuen Programmier- und Steuerungssystems im Vordergrund. Alle Komponenten müssen miteinander verbunden und mit der Prozesssteuerung direkt oder indirekt verknüpft sein, um die Übertragung der Parameter, den Austausch der Steuerungsinformation und die Mitteilung über den Zustand der Komponenten zu ermöglichen. Die Eingabe-/Ausgabe-Funktionen des Programmier- und Steuerungssystems müssen an die Prozesssteuerung angepasst werden. Zwei Ergebnisse resultieren aus dieser Aufgabenstellung:

- Die Prozesssteuerung kann den Produktionsplan des PLS mit den Komponenten des Robotersystems ausführen.
- Zustands- und Steuerungsinformationen können von jeder Automatisierungskomponente zurück an die Prozesssteuerung und somit an das PLS gesendet werden.

Darüber hinaus erfolgt eine Erweiterung der Kommunikation zwischen dem Roboter und der Prozesssteuerung, damit einerseits die Übertragung der Steuerungsinformationen ermöglicht wird sowie andererseits der Roboter seine Status-

meldungen an die SPS und somit an den PLS übermitteln kann. Die Elementaranweisungen (ELA) müssen für jeden Industrieroboter des Produktionssystems erstellt werden, um die Programmierung zu realisieren. Jede ELA ist mit einer eindeutigen Identifikationsnummer zu versehen. Die Ident-Nummer ist der entscheidende Parameter für die Zuordnung eines Roboterprogrammaufrufs und des auszuführenden Roboterprogramms.

Der nächste Schritt bildet die Testphase, hier werden die gesamten Produktionsprozesse getestet. Es wird vor allem darauf geachtet, dass es zu keinen Kollisionen zwischen den Robotern und der Umwelt kommt sowie auf die Einhaltung des Produktionsplans. Verläuft der Testlauf erfolgreich, so wird das TOPS freigegeben. Bei Fehlern muss in den jeweiligen Komponenten nachgebessert werden.

Befindet sich das TOPS in einem einwandfreien Zustand, so kann nun die Anlage zu Schulungs- und Einweisungszwecken verwendet werden. Hierbei werden die zukünftigen Bediener des Produktionssystems geschult und eingewiesen. Das TOPS ermöglicht, das Bediener mit unterschiedlichem Qualifikationsniveau mit dem System arbeiten können.

6 Prototypische Realisierung des Konzepts am Beispiel eines Pilot-Demontagesystems

6.1 Pilot-Demontagesystem

Im Rahmen des durch die Deutsche Forschungsgemeinschaft geförderten Sonderforschungsbereiches 281 „Demontagefabriken zur Rückgewinnung von Ressourcen in Produkt- und Materialkreisläufen“ wurde am Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) der TU Berlin unter der Leitung von Prof. Uhlmann prototypisch ein hybrides Demontagesystem realisiert und erprobt [Uhl-04a, Uhl-04b]. An dem konzipierten System wurde ursprünglich die zerstörende, teilzerstörende und zerstörungsfreie Demontage von Waschmaschinen durchgeführt. Später wurden ebenfalls Kfz-Motoren in das Produktspektrum aufgenommen, um die Flexibilität der Anlage zu demonstrieren.

Ein ökonomisch arbeitendes Demontagesystem kann nur als hybrides Demontagesystem realisiert werden [Sel-00a]. Die Kombination von manuellen, teilautomatisierten und automatisierten Prozessen wird favorisiert, da auf die kognitiven Fähigkeiten des Menschen nicht verzichtet werden kann. Langfristiges Ziel dieser Forschungsarbeiten war es, zu überprüfen, inwieweit bei den hohen Flexibilitätsanforderungen der Demontage automatisierte und manuelle Arbeitsstationen zuverlässig und wirtschaftlich zu einem flexiblen Demontagesystem kombiniert werden können [Sel-01].

In der modularen Anlage werden automatisierte und manuelle Arbeitsstationen über ein flexibles Transportsystem automatisiert miteinander verkettet, siehe **Bild 6.1**. Von einem zentral angeordneten Drehtisch gehen sternförmig sechs Transportbänder ab. Damit ist es möglich, im Sinne einer hybriden Arbeitsweise direkt zwischen dem automatisierten und dem manuellen Arbeitsbereich und damit auch zwischen einzelnen Arbeitsschritten zu wechseln. Der untere automatisierte Arbeitsbereich bildet ein in sich geschlossenes Dreieck, so dass dieser Abschnitt als Bypass für die waagerecht verlaufende, zentrale Transportlinie genutzt werden kann [Kei-04].

Wegen der Komplexität der Handhabungsaufgaben, der Bewegungsflexibilität und der weiten Verbreitung in der Handhabungstechnik werden zwei Knickarmroboter vom Typ KUKA KR 150/200 (KUKA I&II) sowie ein Roboter KUKA 160/60 (KUKA III)

eingesetzt. Der mittlere Roboter KUKA II ist zusätzlich auf einer Linearachse mit einem Verfahrweg von vier Metern positionierbar. Damit wird einerseits der für die Demontage verfügbare Arbeitsraum vergrößert, andererseits ein Arbeiten an mehreren Arbeitsstationen ermöglicht. Zusätzlich zu den drei Knickarmrobotern wurde das im Sonderforschungsbereich 281 entwickelte mehrachsige Handlingsystem „Dodekapod“ integriert. Der Dodekapod unterstützt dabei die manuelle Demontage durch Anordnung des Demontageobjekts in einer ergonomisch günstigen Position [Spu-00]. In der nachfolgenden Tabelle sind die verfügbaren Werkzeuge den Robotern zugeordnet, vgl. **Tabelle 6.1**.

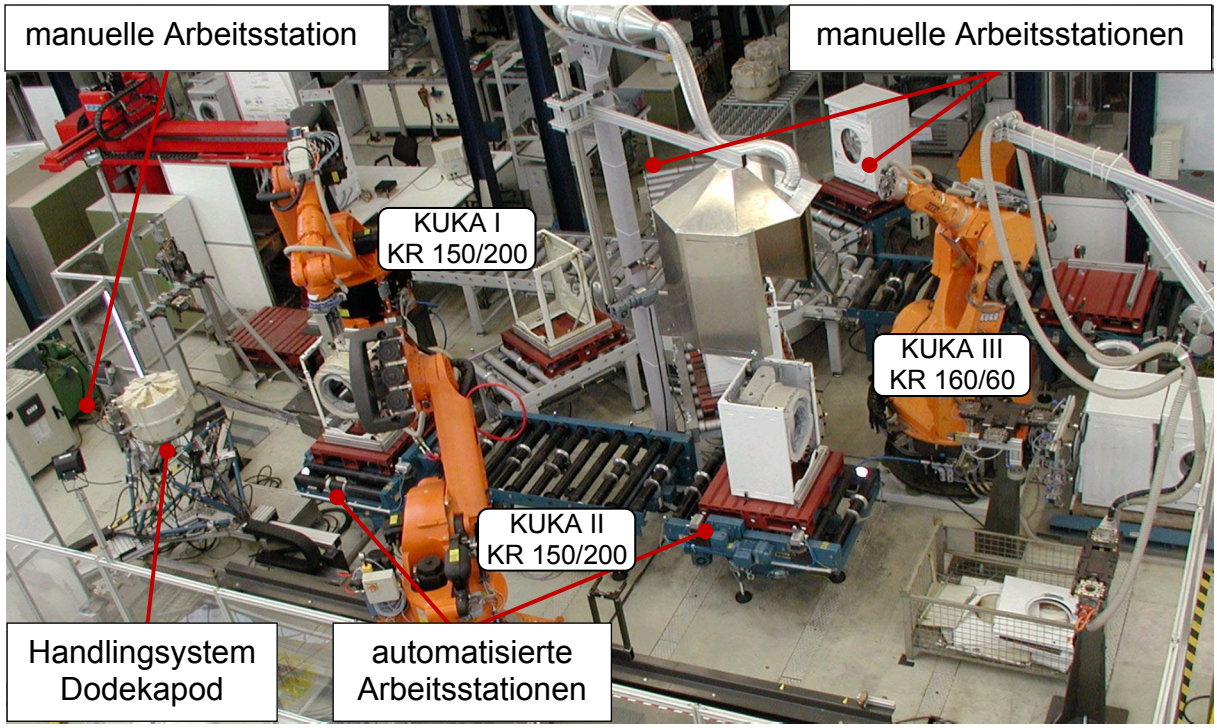


Bild 6.1: Pilot-Demontagesystem des IWF der TU Berlin

Um alle vorgestellten Komponenten miteinander zu verbinden und Steuerungsaufgaben zu ermöglichen, ist das System durch diverse Kommunikationseinrichtungen vernetzt. Die Steuerung der Aktoren, z. B. der Motoren und Pneumatikzylinder an den Förderbändern, sowie der Empfang von Informationen diverser Sensoren, z. B. der Lichtschranken eines Förderbandes, erfolgt über den ASi-Bus (Aktor-Sensor-Interface), der an die SPS des Transportsystems (Simatic S7-300) angeschlossen ist. Über den PROFIBUS werden die Transportsteuerung, die Roboter, der Dodekapod und die Zellenhauptsteuerung (Simatic S7-400) miteinander verbunden. Weiterhin wird die PC-basierte Visualisierungsanwendung WinCC (Windows Control

Center) von Siemens Simatic zur Visualisierung der Abläufe bis hin zu komplexen Prozessüberwachungen und Prozesssteuerungen eingesetzt. Die Sicherheitssteuerung ist ebenfalls an den PROFIBUS angeschlossen. Alle PC-unterstützten Komponenten, also die Roboter und der Bedienerarbeitsplatz, sind über Ethernet verknüpft.

Tabelle 6.1: Verfügbare Werkzeuge im Pilot-Demontagesystem [Reb-03, Här-05, Kei-04, Ste-01, Sel-00b, Sel-02, Sel-03]

Roboter	Werkzeug	Aufgabe
KUKA I	Bohr-Schraubgreifer	Handhabung von dünnen Kunststoff- oder Metallteilen
	Schlagentschraubwerkzeug	Demontage von Schrauben
	3D-Bildvearbeitungsystem	Erkennen von Demontageobjekten
KUKA II	Zerschleifwerkzeug	Trennen von Kunststoffen und metallischen Werkstoffen
	Hydraulikschere	Zerschneiden von Kabeln und Schläuchen
	Stößelgreifer	Handhabung kleiner Demontageobjekte
	Plasmazerstrahlwerkzeug	Trennen flächiger, dünnwandiger metallischer Werkstücke
KUKA III	Sauggreifer groß	Handhabung großer, flächiger Demontageobjekte
	Sauggreifer klein	Handhabung kleiner, flächiger Demontageobjekte

Das gewählte Layout ermöglicht eine Reihe unterschiedlicher Ablaufvarianten. Diese reichen von einem voll automatisierten Durchlauf über eine ausschließlich manuelle Bearbeitung bis hin zur Einzelplatzdemontage am hybriden Dodekapod-Arbeitsplatz. Der Ablauf der Demontage ist primär vom Produktspektrum, dem Demontageziel und dem möglichen Automatisierungsgrad abhängig [Kei-04]. Im Folgenden wird der Demontageablauf für eine Waschmaschine beschrieben.

Die zu demontierende Frontlader-Waschmaschine wird zuerst an der automatisierten Arbeitsstation 2 mit Hilfe des „Sauggreifers groß“ aufgenommen, vgl. **Bild 6.2.** Ist die Waschmaschine durch ein sensorgeführtes Roboterwerkzeug auf der Systempalette korrekt aufgesetzt worden, wird sie mit Hilfe eines Spindeltriebs und Spanndorns fixiert. Danach wird die Waschmaschine an die manuelle Arbeitsstation 12 oder 13 transportiert. Zur Vermeidung möglicher Störungen während der nachfolgenden

automatisierten Demontageabläufe werden manuell alle von außen zugänglichen Teile (wie z. B. die Abdeckplatte und die Frontblende) sowie biegeschlaffe Komponenten im Innenraum wie z. B. die Verkabelung entfernt, siehe **Bild 6.3**. Im nächsten Arbeitsschritt wird eine Öffnung zu den im inneren Raum befindlichen Bauteilen geschaffen. Dazu wird die Waschmaschine an die Arbeitsstation 7 transportiert. An dieser Arbeitsstation, die hauptsächlich als zerstörende automatisierte Arbeitsstation entworfen ist, werden die Seitenwände der Waschmaschine mit Hilfe eines Plasmazerstrahlwerkzeugs herausgetrennt. Auch die Frontwand und der Fensterkomplex werden in diesem Arbeitsschritt mittels eines Zerschleifwerkzeugs gelöst. Die demontierten Teile werden automatisiert mit dem robotergeführten „Sauggreifer klein“ gehalten.

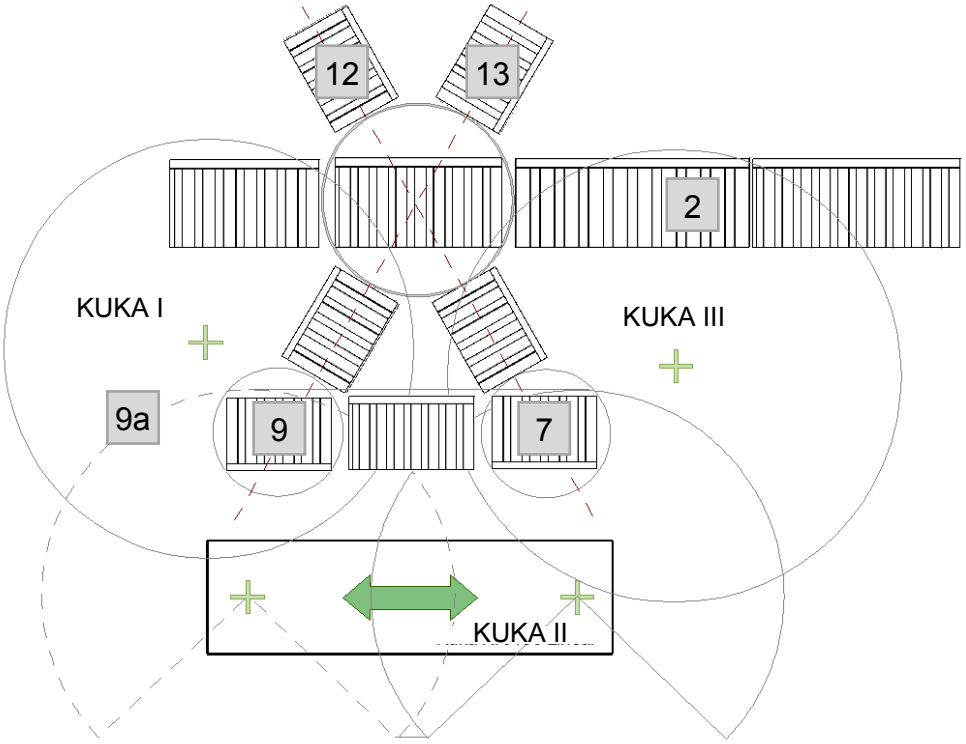


Bild 6.2: Schema des Pilot-Demontagesystems

Die Waschmaschine wird anschließend an die Arbeitsstation 9 übergeben. Ziel dieses Arbeitsschrittes ist es, das Schwingsystem vom Rahmen der Waschmaschine zu trennen. Dazu werden mittels einer Hydraulikschere die Federn und Dämpfer von einem prozessführenden Roboter (KUKA II) zerschnitten. Das Schwingsystem wird dabei von einem anderen Roboter mit Hilfe eines Bohr-Schraubgreifers gehalten. Dieser entwickelte Schraubgreifer erzeugt zur Krafteinleitung seine Wirkfläche selbst

und ist daher geeignet, dünnwandige Metall- und Kunststoffteile unbekannter Geometrie zu handhaben. Das Schwingsystem wird an den Dodekapod (Arbeitsstation 9a) übergeben und aus einer Kombination von manuellen Verrichtungen, automatisiertem Schlagentschraubwerkzeug, Stößelgreifer und anschließendem Entnehmen der Teile demontiert.

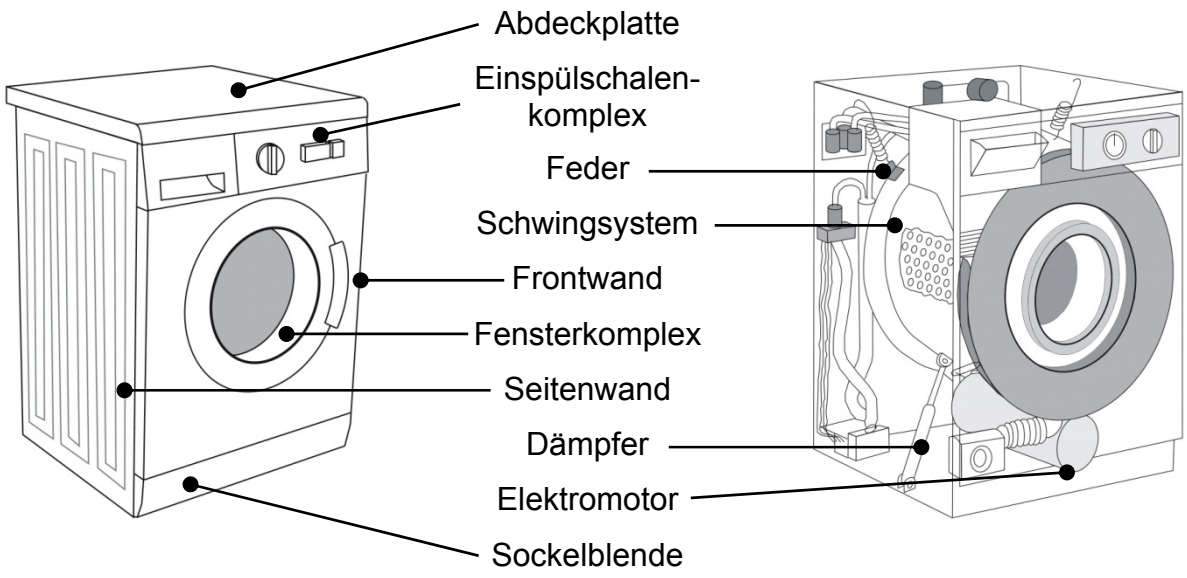


Bild 6.3: Baugruppen einer Frontlader-Waschmaschine (ohne biegeschlaffe Komponenten)

Die prototypische Realisierung des Konzeptes definiert, dass für die Kommunikation zwischen dem Pilot-Demontagesystem und dem Programmierleitstand die Schnittstelle OPC (Openess, Productivity, Collaboration) zum Einsatz kommt, vgl. **Bild 6.4**. OPC definiert eine offene Schnittstelle, über die PC-basierte Softwarekomponenten Daten austauschen können. Sie basiert auf den Windows-Technologien OLE (Objekt Linking Embedding), COM (Component Objekt Model) und DCOM (Distributed COM). OLE ist hierbei eine spezielle Methode zur gemeinsamen Nutzung von Informationen in Microsoft-Anwendungsprogrammen. Die Wahl für OPC ist begründet in den Möglichkeiten, sowohl große statische Datenmengen als auch Prozessdaten hierüber austauschen zu können. Eine vereinfachte Nutzung in Netzwerken ergänzt die Vorteile von OPC [Iwa-05].

Die Übermittlung eines Demontageplans erfolgt über die Zellenhauptsteuerung. Die Zellenhauptsteuerung ist damit Zwischenspeicher des Demontageplans für alle Teilnehmer und ermöglicht die Verteilung der gespeicherten Parameterdaten an die

Zellenkomponenten. Dieses wird über den Profibus realisiert, da dieser durch sein deterministisches Verhalten sowohl zur Übertragung von Steuerungsdaten als auch für umfangreiche Parameterdaten genutzt werden kann.

Im Rahmen dieser Arbeit wird nur die Anwendung des Konzeptes anhand von zwei Zellenkomponenten realisiert. Hierfür wurden Roboter und das Transportsystem verwendet.

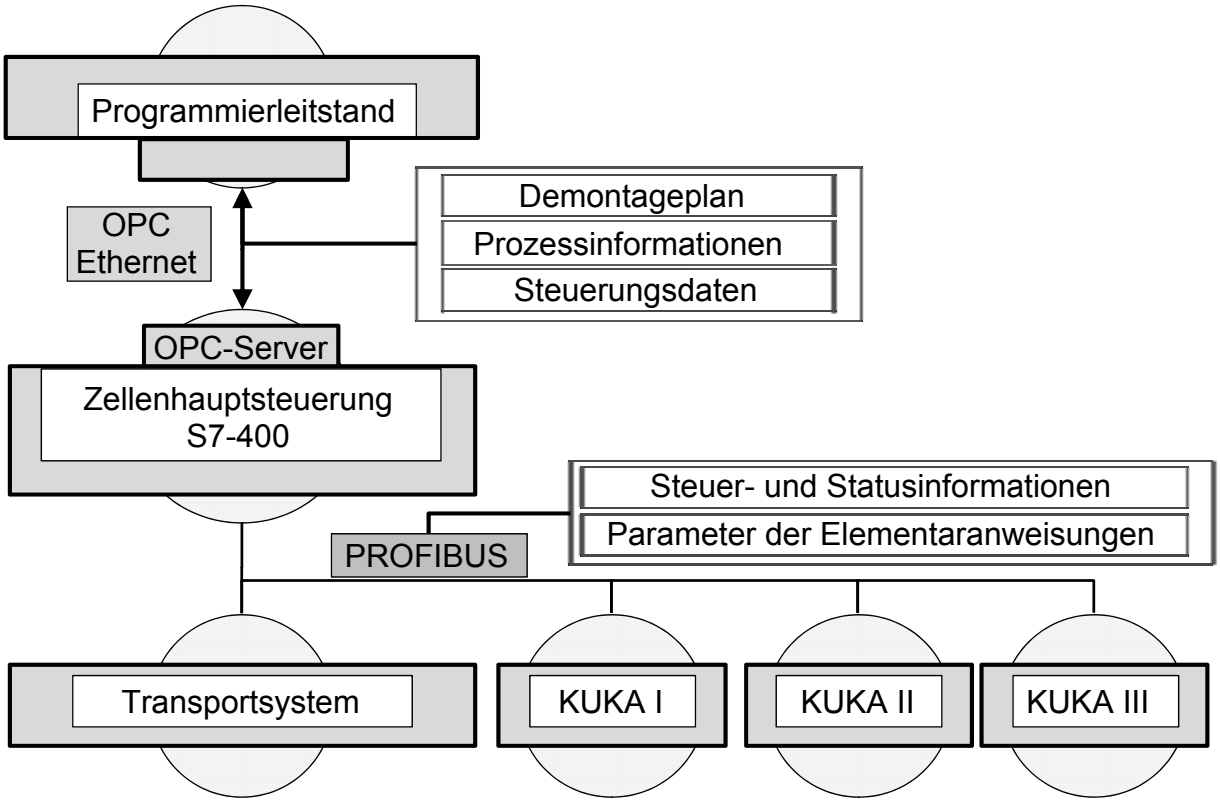


Bild 6.4: Lösungskonzept für das technologieorientierte Programmier- und Steuerungssystem

6.2 Adaption existierender Komponenten

6.2.1 Robotersteuerung

Die Grundlage für die Erweiterung der bestehenden Roboterprogrammierung war die Analyse der Demontageprozesse, die für die Demontage einer Waschmaschine benötigt werden. Hierzu wurden die in Kapitel 5.1 vorgestellten Demontageprozesse des Pilot-Demontagesystems hinsichtlich der Gliederung in Elementaranweisungen analysiert, siehe **Tabelle 6.2**. Die Analyse ergab, dass folgende sieben Elementaranweisungen (ELA) benötigt werden:

- PTP_Base_Ohne Ereignis (Ident-Nummer 100),
  Diese ELA ist eine einfache Punkt-zu-Punkt-Bewegung im Werkstück-Koordinatensystem.
- Lin_Base_Ohne Ereignis (Ident-Nummer 120),
  Diese ELA ist eine einfache Positionierung des Robotersystems auf dem kürzesten Weg zwischen zwei Punkten im Arbeitsraum, einer Geraden.
- Lin_Base_Mit Ereignisstopp wenn Sensor1=„true“ (Ident-Nummer 121),
  Diese ELA ist eine Linear-Bewegung im Werkstück-(Base)-Koordinaten-system, die abgebrochen wird, wenn ein überwachter Sensor den Zustand „wahr“ einnimmt.
- Lin_Base_Mit Ereignisstopp wenn Sensor1=„true“ und Sensor2=„true“ (Ident-Nummer 122),
  Diese ELA ist eine Linear-Bewegung im Werkstück-(Base)-Koordinaten-system, die abgebrochen wird, wenn die beiden überwachten Sensoren den Zustand „wahr“ einnehmen.
- Circ_Base_Ohne Ereignis (Ident-Nummer 140),
  Diese ELA ist eine einfache Kreisbogen-Bewegung zum Endpunkt im Werk-stück-Koordinatensystem. Die Bahn wird durch Start-, Hilfs- und Endpunkt beschrieben.
- $OUT_kontinuierlich (Ident-Nummer 400) und
  Diese ELA weist kontinuierlich einem digitalen Ausgang den Wert „wahr“ zu.
- $OUT_abfragen (Ident-Nummer 412).
  Diese ELA fragt den Zustand eines digitalen Ausgangs ab.

Durch die Mehrfachverwendung dieser ELA unter der Verwendung spezifischer Steuerungsinformationen werden alle benötigten Prozesse abgebildet, beispiels-weise wurde die ELA „PTP_Base_Ohne Ereignis“ 48 Mal genutzt, siehe **Anhang 3**.

Tabelle 6.2: Aufteilung des Arbeitsschrittes „Demontage Seitenwand“ in Elementaranweisungen

	Demontage Seitenwand		
	Prozessschritt	**Wiederholungen**	**Ident-Nummer**
KUKA II	Fahren nach Save-Point 2	4	100
	Plasmazerstrahlwerkzeug aufnehmen	1	Makro
	Synchronisationsmarke Transportsystem		
	Synchronisationsmarke KUKA III		
	Fahren nach „Startpunkt“	1	100
	Plasmazerstrahlwerkzeug einschalten	1	400
	Kontrolle, ob Plasmazerstrahlwerkzeug eingeschaltet	1	412
	Suchstrecke mit Plasmazerstrahlwerkzeug fahren, bis Zündung erfolgt	1	123
	Schnittfahrt	4	120
	Fahren nach „Startpunkt“	1	120
	Plasmazerstrahlwerkzeug ausschalten	1	400
	Kontrolle, ob Plasmazerstrahlwerkzeug ausgeschaltet	1	412
	Synchronisationsmarke Transportsystem		
	Synchronisationsmarke KUKA III		
	Fahren nach Save-Point 2	1	100
	Plasmazerstrahlwerkzeug ablegen	1	Makro
	Fahren nach Save-Point 2	1	100
KUKA III	Fahren nach Save-Point 2	4	100
	Sauggreifer klein aufnehmen	1	Makro
	Fahren nach „Startpunkt“	1	100
	Sauggreifer klein einschalten	1	400
	Kontrolle ob Sauggreifer klein eingeschaltet	1	412
	Fahren der Suchstrecke bis Saugpunkt erreicht	1	120
	Synchronisationsmarke KUKA II		
	Abfahren mit angesaugter Seitenwand	2	120
	Fahren nach Materialsammelstelle für Seitenwände	7	100
	Sauggreifer klein ausschalten	1	400
	Kontrolle, ob Sauggreifer klein ausgeschaltet	1	412
	Fahren nach Save-Point 2	1	100
	Sauggreifer klein ablegen	1	Makro

Aus Gründen der vereinfachten Darstellung wurden die Elementaranweisungen für das Aufnehmen und Ablegen eines Werkzeuges zusammengefasst und als ein Makro dargestellt. Aufgrund der verwendeten standardisierten Werkzeugwechselsysteme werden für das Aufnehmen und Ablegen eines Werkzeuges die gleichen Elementaranweisungen in derselben Reihenfolge benötigt [Sel-05].

Bei der Analyse der Demontageprozesse wurde festgestellt, dass eine ELA diverse Parameter zur Konfiguration erhalten muss. Für die Übertragung eines vollständigen Parametersatzes (25 Parameter) sind 87 Byte erforderlich, siehe **Tabelle 6.3**.

Tabelle 6.3: Parameter einer Elementaranweisungen

Parameter	Einheit	Typ und Anzahl	Anzahl	Menge
Ident-Nummer		INT	1	2 Byte
Base	mm/grad	REAL	6	24 Byte
Point_1	mm/grad	REAL	6	24 Byte
Point_2	mm/grad	REAL	6	24 Byte
Werkzeugnummer		BYTE	1	1 Byte
PTP_Geschwindigkeit	%	INT	1	2 Byte
LIN_Geschwindigkeit	m/s	REAL	1	4 Byte
Sensor_1		INT	1	2 Byte
Sensor_2		INT	1	2 Byte
Sensor_3		INT	1	2 Byte

Für einen Überblick über die Größenordnung von den zu übertragenden Daten dient folgendes Beispiel, das nur einzelne Demontageprozesse einer Demontage berücksichtigt: Für eine Demontage der rechten und linken Seitenwand einer Waschmaschine mit dem Plasmazerstrahlwerkzeug (2x19 ELA) mit darauffolgender Demontage des Schwingsystems (27 ELA) ist eine Elementaranweisungsliste von 65 Zeileneinträgen mit einer Datengröße von 5655 Byte (65x87Byte) nötig, sofern alle Parameter beschrieben werden. Dieses betrifft allerdings nur die Anweisungen für einen Roboter. Unter Berücksichtigung der weiteren Roboter muss der Faktor 3 herangezogen werden, da diese ebenfalls eine Anzahl von ELA dieser Größenordnung erfordern. Für die Komponente „Dodekapod“, die ebenfalls in den Demontageplan integriert wird, müssen Angaben im Demontageplan enthalten sein, die die auszuführende Aktion eindeutig beschreiben. Deren Größe ist jedoch aufgrund des kleineren Funktionsumfanges weitaus geringer und soll vorerst vernachlässigt

werden. Zusammenfassend wird es sich bei der Übermittlung eines gesamten Demontageplans um circa 20 - 100 Kilo Byte handeln.

Daraufhin wurde für die Übertragung der Parameter von der Zellensteuerung zum Industrieroboter die bestehende Cell-Standard-Kommunikation über den Profibus erweitert, vgl. **Bild 6.5**. Da für die Übertragung der Parameter die digitalen Eingänge der Robotersteuerungen benutzt werden, sind 704 digitale Eingänge erforderlich.

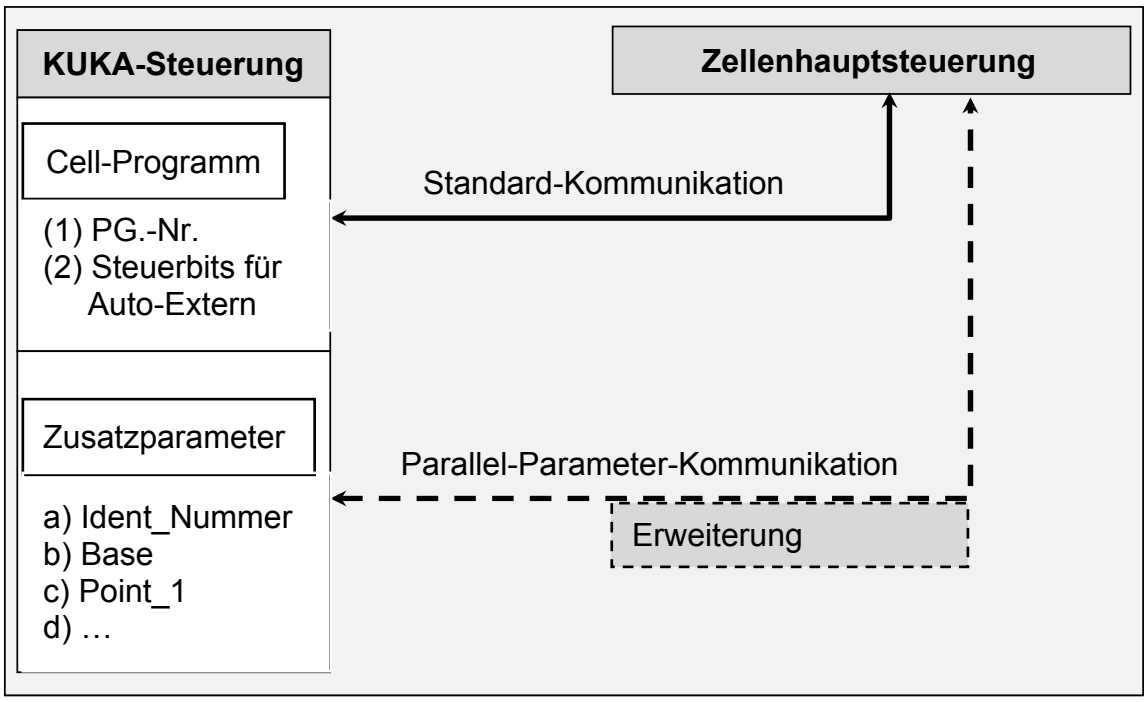


Bild 6.5: Erweiterung der Kommunikation zwischen Industrieroboter und Zellenhauptsteuerung

Der Parameter Ident-Nummer ist der entscheidende Parameter für die Zuordnung einer Elementaranweisung, die auf dem Roboter hinterlegt ist. Die Parameter Base, Point_1, Point_2 sind Angaben in kartesischen Koordinaten für den Arbeitsraum und dienen dem Roboter als Basis und Eckpunkte für Bewegungen. Bei ihnen liegt der Wertebereich zwischen +9999,9999 bis -9999,9999 mit einer Genauigkeit von vier Nachkommastellen, da die Steuerung der KUKA-Roboter bis maximal vier Stellen nach dem Komma genau berechnen kann. Des Weiteren geben die Parameter Werkzeugnummer, PTP_Geschwindigkeit und LIN_Geschwindigkeit Eigenschaften einer Fahrt wie Verfahrgeschwindigkeit und Werkzeugbestückung an. Die Parameter Sensor_1 bis Sensor_3 dienen der Behandlung von Interrupts während der Ausführung einer ELA, wie sie durch einen Sensor am Roboter erzeugt wird.

Damit die Parameter durch die Robotersteuerung verarbeitet werden können, mussten die Robotersteuerung angepasst und Funktionen neu programmiert werden. Die Konfigurationsdatei „$CONFIG.DAT“ und das Programm „CELL.SRC“ wurden angepasst. Die Konfigurationsdatei „$CONFIG.DAT“ ist eine von KUKA vordefinierte Datenliste, die aber keine vordefinierten Systemvariablen enthält. In solchen Systemvariablen werden Variablen, Strukturen, Kanäle und Signale definiert, die längere Zeit gültig und für viele Programme von übergeordneter Bedeutung sind. In dieser Datei wurden globalen Variablen deklariert, um eine programm- und funktionsübergreifende Verarbeitung der Parameter zu ermöglichen. Das Programm zur Steuerung von Robotern über eine zentrale SPS- die „CELL.SRC“, die entsprechend einer Programmnummer ein Bauteileprogramm wählt- wurde um die Programme „DTTP.SRC“ und „ELA.SRC“ erweitert. Das Programm „DTTP.SRC“ liest die digitalen Eingänge aus und transferiert sie in Integer- oder Realzahlen, wobei die Werte in globalen Variablen gespeichert werden, vgl. **Bild 6.6**.

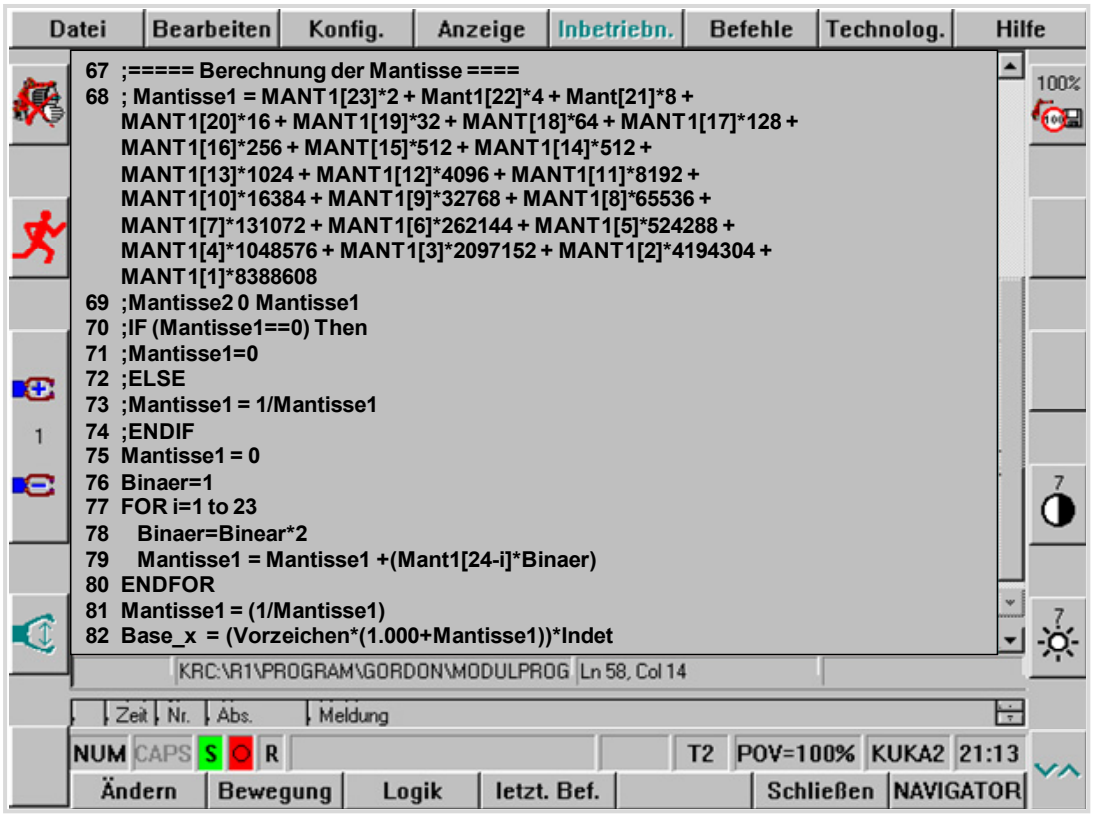


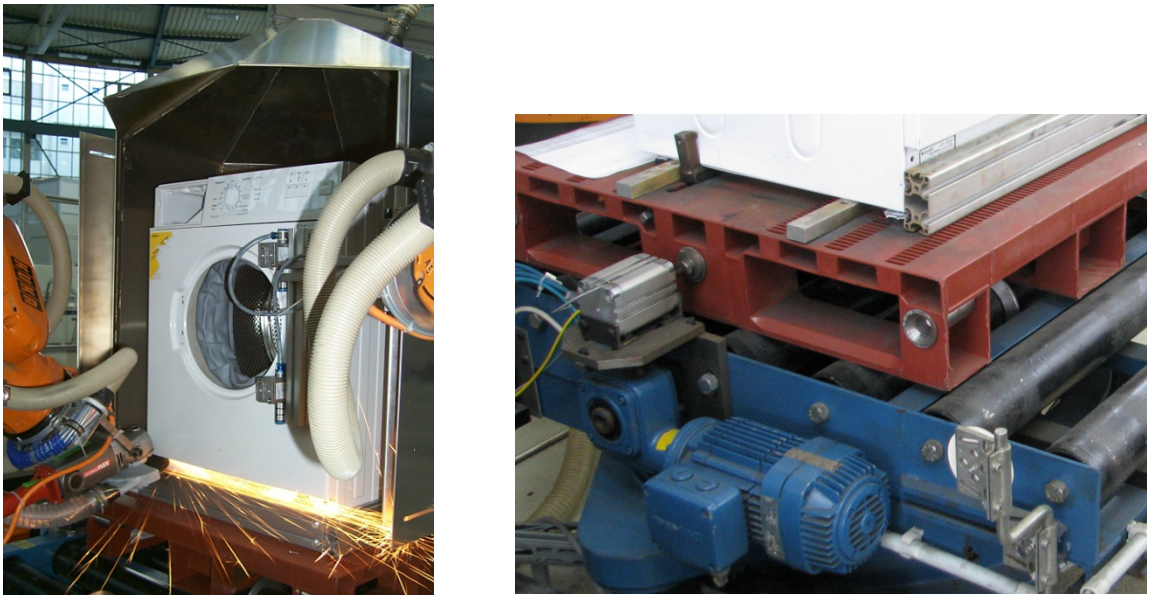
Bild 6.6: Auszug aus dem KUKA-Programm “DTTP.SRC“

Die Funktion „ELA.SRC“ enthält als Case-Struktur alle benötigten Elementaranweisungen, die aufgrund der transferierten Daten ausgeführt werden. Hierbei wird die Identifizierung der Elementaranweisungen aufgrund der Ident-Nummer durch-

geführt. Zur Validierung der Übertragung wurden alle Datensätze auf der S7-Steuerung mit beispielhaften Gleitpunktzahlen versehen und auf die Eingänge der Robotersteuerung geschrieben. Der erfolgreiche Empfang der Daten und die nötigen Formatwandlungen wurden von den Robotern mit der Rückgabe eines definierten Ausgangsbits quittiert. Hierbei wurde bei den gemessenen Übertragungen und bei geringer Busauslastung eine durchschnittliche Gesamtübertragungszeit von 528 ms gemessen.

6.2.2 Speicherprogrammierbare Steuerung

Damit im realisierten technologieorientierten Programmier- und Steuerungssystem (TOPS) für alle Teilnehmer Anweisungslisten in Form eines Demontageplanes verwendet werden können, werden die Absaughaube und Pneumatikventile des Pilot-Demontagesystems als eigenständige Komponenten betrachtet, vgl. **Bild 6.7**. Für die Integration aller Einzelkomponenten in das TOPS und für deren automatisierte Verwendung sollen die Funktionen der Einzelkomponenten über Programmnummern ansprechbar sein, wobei die Realisierung der Anweisungen an den Aufbau der Elementaranweisungen für die Robotersteuerung angelehnt ist, siehe **Kapitel 6.2.1**.



a) Absaughaube zur Verringerung der Partikelemission

b) Pneumatikventile zur Fixierung der Palette auf dem Transportsystem

Bild 6.7: Absaughaube und Ventil des Pilot-Demontagesystems

Für das Arbeiten mit der Komponente Transportsystem wurde ein Funktionsbaustein geschrieben, der alle Aktionen wie Fahrten zwischen zwei Positionen und die Steuerung der Drehtische an den Arbeitsstationen 7 und 9 zusammenfasst. Des Weiteren wurde für die Quellen- und Zielangabe eine Parametrisierung entwickelt, die ähnlich den globalen Parametern der Elementaranweisungen ist und damit die Verknüpfung mit nur einer Programmnummer (PG_Nummer) ermöglicht, siehe **Tabelle 6.4**.

Darüber hinaus wurde für die Komponenten Absaughaube und Pneumatikventile Funktionsbausteine programmiert, die die Parametrisierung nutzen, um die unterschiedlichen Zustände der Komponenten abzubilden. Unter der Voraussetzung, dass der Programmierleitstand einen Demontageplan an die Zellenhauptsteuerung übergeben hat, kann diese mit der Ausführung des Plans beginnen, wobei alle Steuerungsabläufe asynchron ausgeführt werden, siehe **Bild 6.8**.

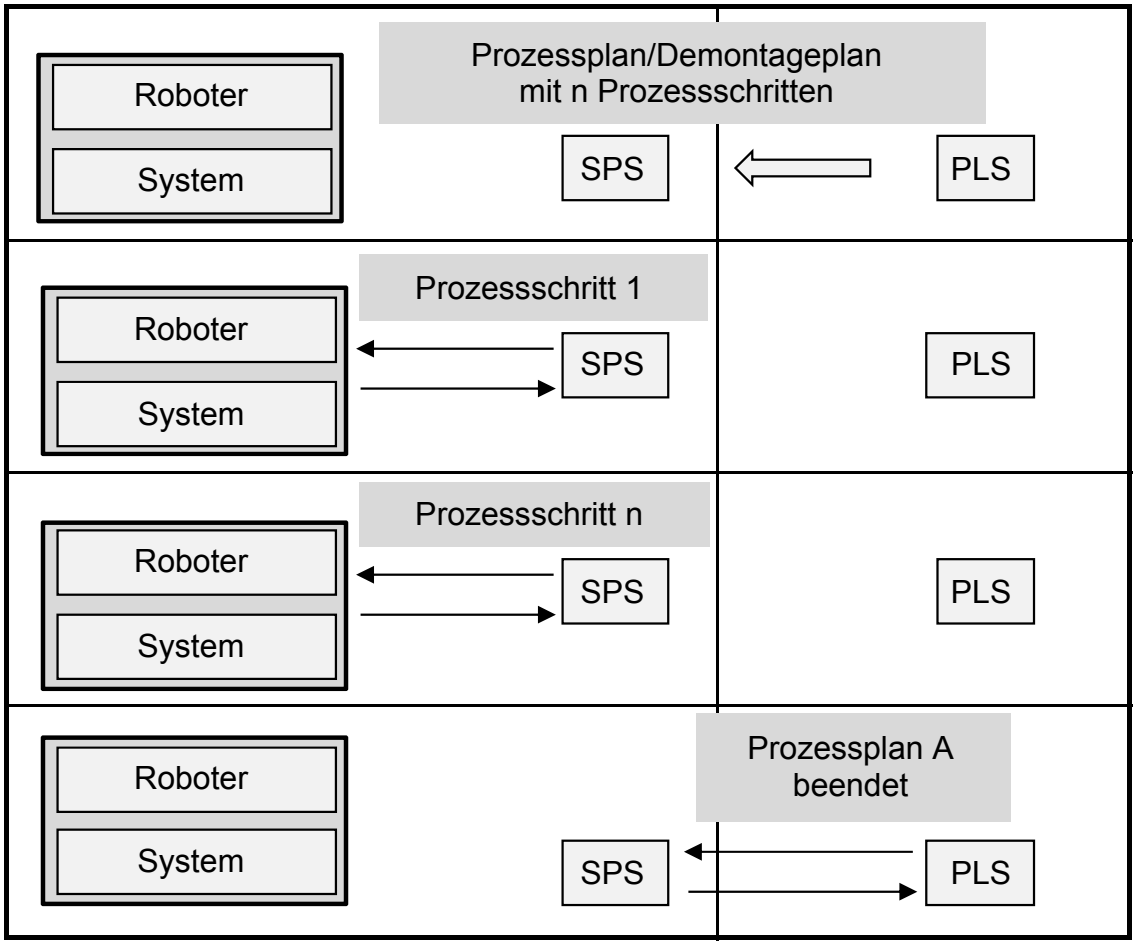


Bild 6.8: Idealer Ablauf der Demontageplanausführung

Tabelle 6.4: Codierung für die Komponenten Transportsystem, Absaughaube und Pneumatikventile

PG_Nummer			Aktion
Transportsystem			
Fahrten			
1	Quell-position	Ziel-position	Fahrt zwischen zwei direkt benachbarten Positionen
Drehpositionen für Arbeitsstation 7			
	Quelle	Ziel	
1	27	37	Position 0°
1	28	47	Position 180°
1	25	67	zu Position 6 vor
1	26	76	zu Position 6 zurück
1	30	78	zu Position 8 vor
1	29	87	zu Position 8 zurück
Drehpositionen für Arbeitsstation 9			
	Quelle	Ziel	
1	15	39	Position KUKA I
1	16	49	Position Dodekapod
1	17	89	zu Position 8 vor
1	18	98	zu Position 8 zurück
1	19	109	zu Position 10 vor
1	20	910	zu Position 10 zurück
Absaughaube			
2	0	1	Absaughaube senken
2	1	0	Absaughaube anheben
Pneumatikventile			
3	0	2	Pneumatikventile Station 2 auf
3	2	0	Pneumatikventile Station 2 zu
3	0	7	Pneumatikventile Station 7 auf
3	7	0	Pneumatikventile Station 7 zu
3	0	9	Pneumatikventile Station 9 auf
3	9	0	Pneumatikventile Station 9 zu

Der Demontageplan wird auf Gültigkeit überprüft und analysiert. Bei der Analyse des Demontageplanes wird die Anzahl der Zeileneinträge ermittelt, die der jeweilige Teilnehmer ausführen muss, wobei dieser Wert für die Abarbeitung in der Fortschritts-

anzeige gespeichert wird. Nach Erhalt des Signals „Start“ beginnt die Zellensteuerung mit der Ausführung der „Demontagepläne“. Nachfolgend wiederholt sich eine Prozedur, die darin besteht, den aktuellen Zeileneintrag jedes Teilnehmers einzulesen, daraufhin zu entscheiden, ob es sich um eine Aktion oder eine Synchronisationsmarke (Ident-Nummer=999) handelt, und dementsprechend zu reagieren. Bei Aktionen wird das Modul für den entsprechenden Teilnehmer aufgerufen, das die Ausführung einer Zeile bewirkt. Bei Erreichen einer Synchronisationsmarke wird für den jeweiligen Teilnehmer das Modul „Sync_Modul“ aufgerufen, das erst dann verlassen wird, wenn das Synchronisationsereignis eingetreten ist. Unabhängig davon, ob eine Synchronisation oder Aktion erfolgte, wird nach dem erfolgreichen Beenden dieses Schritts der Planindex der jeweiligen Komponente inkrementiert und die Ausführung beginnt für den Teilnehmer von Neuem. Dabei wird vor jeder Ausführung der Demontagepläne überprüft, ob der Zeileneintrag leer ist und somit ob die Ausführung beendet ist. Sobald der Plan für beide Teilnehmer keine Anweisungen mehr enthält, ist die Demontage beendet. Dieses ist die Aufforderung an das PLS, einen neuen Demontageplan zu generieren und zu übermitteln.

Für die Realisierung des beschriebenen Ablaufes wurden verschiedene Funktions- und Datenbausteine sowie Funktionen in der Zellenhauptsteuerung programmiert. Darüber hinaus wurde die Software „Simatic Net PC Software“ genutzt, um den Demontageplan inklusive der Steuerungsdaten von dem Programmierleitstand an die Zellenhauptsteuerung zu übermitteln. Der OPC-Client wurde eingerichtet, so dass Merker (M), Eingänge (E), Ausgänge (A) oder Datenbausteine der Zellenhauptsteuerung mit definierten Schreib- oder Leserechten für die PLS zugänglich gemacht wurden.

Für die getrennte Zwischenspeicherung des Demontageplans wurden Datenbausteine programmiert, die Listen in Form von Feldern (Arrays) darstellen, so dass bei Verwendung von mehreren Arrays außerdem Spalten analog zu einer Tabelle entstehen, siehe **Anhang 4**. Mit einem Index und den Namen der diversen Arrays lassen sich somit Zeilen darstellen, die für die Speicherung eines Programm- oder Demontageschritts verwendet werden. Die Datenbausteine 7-10 wurden programmiert, um den Demontageplan der Teilnehmer KUKA I, KUKA II, KUKA III und SYSTEM zu speichern.

Der entscheidende Funktionsbaustein (FB) zur asynchronen Ausführung des Steuerungsablaufes auf der Zellensteuerung (Simatic S7-400) ist der FB 27 „TOPS_Steuermodul“. Mittels dieses Funktionsbausteins ist es möglich, die aktuell eingebundenen Komponenten asynchron zu steuern und den Ablauf der Demontageplanausführung wie beschrieben zu realisieren, siehe **Anhang 5**.

Um eine Elementaranweisung mit den Robotern auszuführen, müssen die Parameter an die Robotersteuerung übertragen und eingelesen werden (Programmaufruf „DTTP.SRC“), damit anschließend die entsprechende ELA in der KUKA-Cell-Datei (Programmaufruf „ELA.SRC“) ausgeführt werden kann. Die Steuerung dieser Roboterfunktion übernimmt der Funktionsbaustein FB 22, der mit Hilfe der Funktion (FC) 4 die Parameter des Demontageschrittes auf die digitalen Eingänge der Roboter schreibt. Der FB 21 wird genutzt, um Roboterprogramme auszuführen, vgl. **Bild 6.9**. Sobald die digitalen Ausgänge eines Roboters beschrieben sind, wird mit Hilfe des FB 21 das Programm „DTTP.SRC“ ausgeführt und die Informationen der digitalen Eingänge in globalen Variablen der Robotersteuerung werden gespeichert. Der FB 21 wird erneut genutzt, um die entsprechende ELA mit Hilfe des Programms „ELA.SRC“ auszuführen.

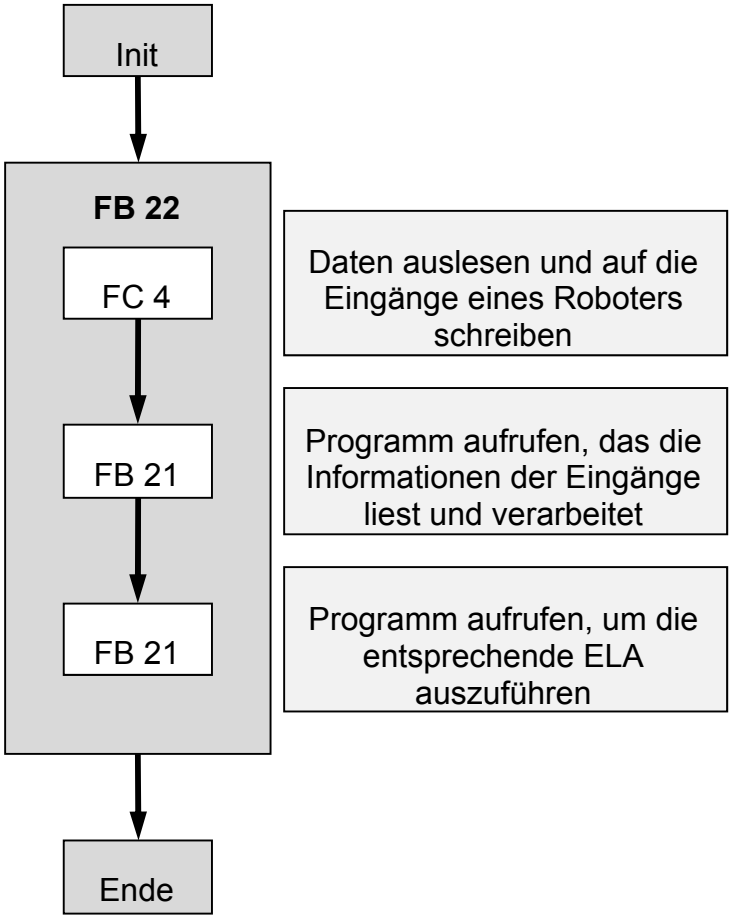


Bild 6.9: Ablauf der Ausführung einer Zeile des Demontageplanes für KUKA I, II und KUKA III

Die Ausführung einer Programmanweisung der Komponenten Transportsystem, Absaughaube und Pneumatikventile, die in einer Zeile des Demontageplans hinterlegt wurden, erfordert das Auslesen des entsprechenden Zeileneintrages und die Übergabe der Parameter, die durch den Funktionsbaustein FB 25 realisiert werden. Dieser initiiert aufgrund der eindeutigen Programm-Nummer die gewünschte Aktion unter Beachtung der Parameterwerte.

6.3 Realisierung zusätzlicher Komponenten

6.3.1 Datenbanksystem

Entwurfsgrundlage des Datenbankmodells ist das von Peter Chen entwickelte Entity-Relationship-Modell, das einen Ausschnitt der Realwelt unter Verwendung von „entities“ und „relationships“ beschreibt [Sum-07]. Hierbei sind die relevanten Informationen des Modells durch Entitäten (Gegenstände) repräsentiert sowie ihre

Abhängigkeit und Wechselwirkungen durch ein Relationship (Beziehung) gekennzeichnet. Für die Entwicklung und physische Integration des Datenbanksystems (DBS) wird „MySQL“ als Datenbankmanagementsystem (DBMS) verwendet. Ein Vorteil der Nutzung von „MySQL“ als DBMS ist der mögliche Einsatz bei Web-Anwendungen, wobei sich die entwickelten Datenbanken (DB) über einen Browser bedienen lassen [Buc-05]. Eine von vielen Integrationsalternativen ist die Skriptsprache „PHP“ (Hypertext Preprozessor), die zur Administration von MySQL-Datenbanken dient. PHP ist eine verbreitete und für den allgemeinen Gebrauch bestimmte Open-Source-Skriptsprache, die speziell für die Web-Programmierung geeignet ist und in „HTML“ (Hypertext Markup Language) eingebunden werden kann [Php-08].

Für die prototypische Entwicklung des Datenbanksystems im Pilot-Demontagesystem mit einer browserbasierten Web-Anwendung als Bedienoberfläche wurde das vorkonfigurierte Softwarepaket „XAMPP“ verwendet. „XAMPP“ ermöglicht das einfache Installieren und Konfigurieren des Webservers Apache mit der Datenbank „MySQL“ und der Skriptsprache „PHP“. Für die einfache sowie intuitive Administration und Darstellung der Daten wurde eine Schnittstelle zwischen dem DBS und dem Endanwender entwickelt, die es ermöglicht, Einträge eigenständig hinzuzufügen, zu editieren und zu löschen, siehe **Bild 6.10**. Weiterhin sollten für die Administration des DBS die Stammdaten der DB sich über einen Webbrowser verwalten lassen, wozu eine Verbindung zwischen der browserbasierten Web-anwendung und dem DBS realisiert wurde. Alle Informationen, die für die Demontage eines Produktes im Pilot-Demontagesystem notwendig sind, können durch „Klicken“ auf eine der verschiedenen Funktionsauswahlpunkte in der Menüliste der Hauptseite aufgerufen werden.

Die Flexibilitätsanforderungen des technologieorientierten Programmier- und Steuerungssystems erfordern die Verwaltung der folgenden Bereiche:

- Demontageprodukt,
- Roboterprogramm und
- Systeminformation.

Alle Daten dieser Bereiche werden durch eine interne Identifizierungsnummer (Id) erkannt. Das ermöglicht die einfache Verfolgung der Daten während der Übertragung und die relationale Beziehung der zu erzeugenden Tabellen.

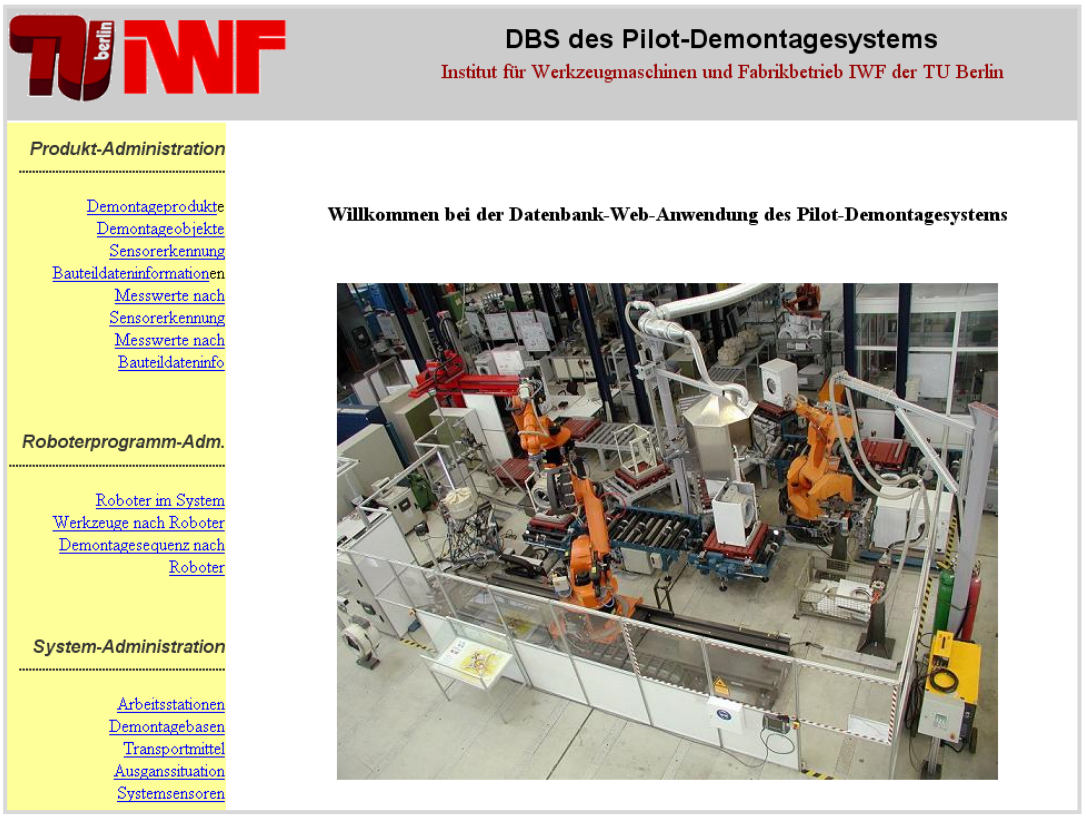
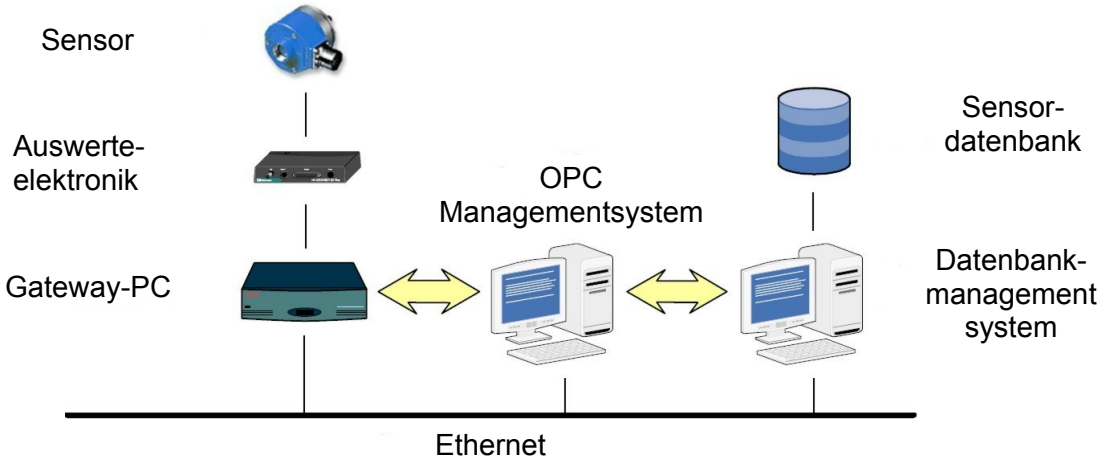
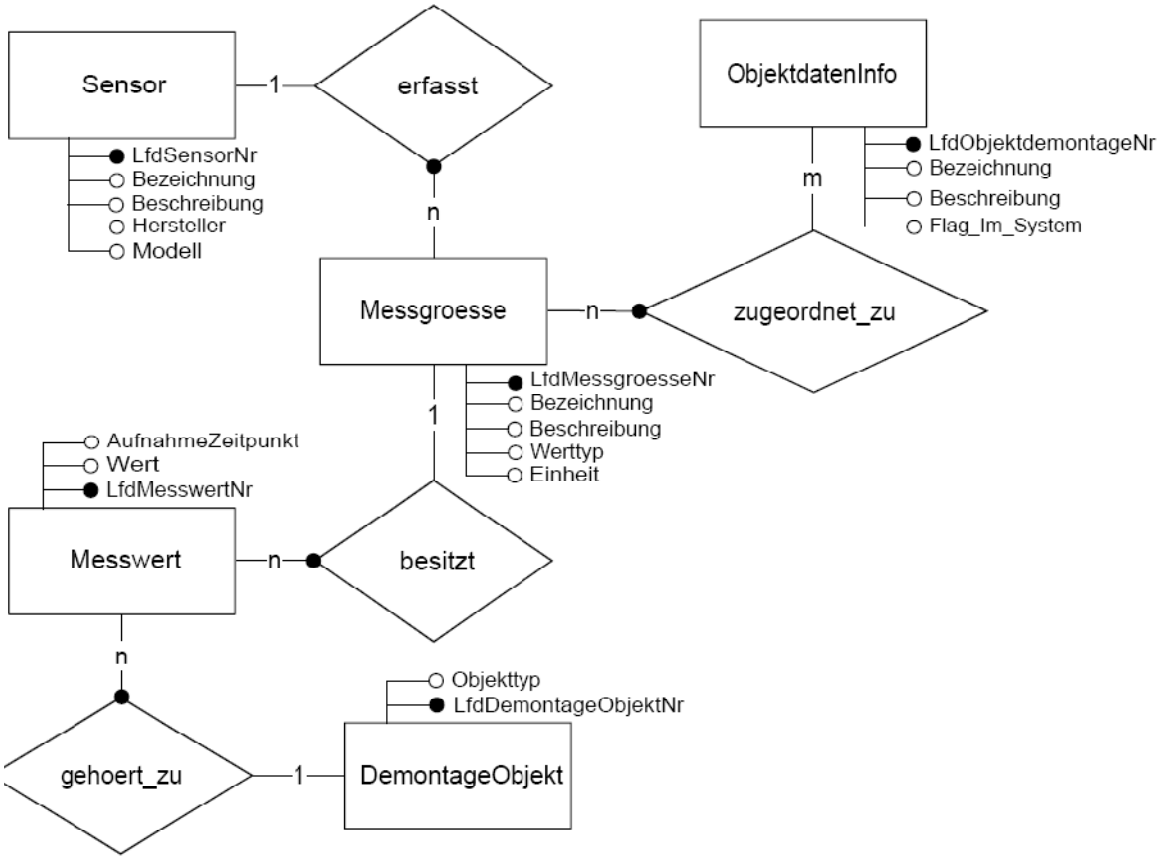


Bild 6.10: Hauptseite der browserbasierten Webanwendung

Der *Demontageproduktbereich* bezieht sich auf das physische Produkt des Pilot-Demontagesystems, eine Waschmaschine (WM). Der Demontageproduktbereich besteht aus zwei Komponenten: aus einer Sensordatenbank zur Erkennung des Produktes und einer weiteren Komponente, wo bestimmte Eigenschaften der WM zur Beschreibung ihres Aufbaus dokumentiert sind. Falls also ein Produkt zum ersten Mal im Pilot-Demontagesystem demontiert wird und keine technischen Informationen vorhanden sind, wird das Produkt im System mit Hilfe des 3D-Bildvearbeitungsystems gescannt, um bestimmte Metadaten wie beispielsweise Länge, Breite und Höhe oder Beschädigungen zu erfassen, vgl. **Bild 6.11**. Hierbei sind für die Sensordatenbank grundsätzlich der Sensor, die Messgröße und der Messwert von großer Bedeutung.



a) Akquisition der Messwerte im Demontagesystem



b) ER-Diagramm der Sensordatenbank

Bild 6.11: Integrierte Sensordatenbank

Die Produktdaten enthalten technische Informationen zur Erkennung einer bestimmten WM, wie z. B. Hersteller, Typ, Modell, Herstellungsdatum u. a. Für die Demontage einer WM im Pilot-Demontagesystem wird das Produkt in Demontageobjekte unterteilt, welche Koordinatenbasen besitzen, die für den Bewegungsablauf eines KUKA-Roboters erforderlich sind. Für diesen Zweck wurde die WM in zehn Demontageobjekte aufgeteilt, wobei neun der Objekte verschiedene Teile einer WM

sind, während das zehnte Objekt sich auf die WM selbst bezieht, siehe **Tabelle 6.5**. Das Objekt Waschmaschine wird benutzt, um die Handhabungsaufgaben zu ermöglichen. Alle Demontageobjekte sind in einer Liste der Datenbank hinterlegt und den Demontagesequenzen zugeordnet.

Darüber hinaus wurden den Demontageobjekten Objektbasen zugeordnet, die Referenzpunkte für die Berechnung der Bewegungskoordinaten der Roboter sind. Diese sind in einer Tabelle im *Demontageproduktbereich* der Datenbank dokumentiert. Die Beschreibung der Objektbasen erfolgt mit Hilfe des Konzeptes zur Lage eines Körpers im Raum durch sechs Koordinaten im Bezugskoordinatensystem.

Tabelle 6.5: Demontageobjekte einer Waschmaschine

Demontageobjekt	Demontage	
	zerstörungsfrei	zerstörend
Schläuche an der Rückwand		X
Abdeckplatte	X	
Fensterkomplex		X
Frontwand		X
linke Seitenwand		X
rechte Seitenwand		X
Schwingsystem		X
Betongewicht	X	
Elektromotor	X	

Das **Bild 6.12** zeigt eine der Funktionalitäten zur Verwaltung der Demontageprodukte. Aufgrund der geforderten Produktvariantenflexibilität des Demontagesystems muss es möglich sein, unterschiedliche Produkttypen und -varianten zu demontieren. Daher werden u. a. im Demontageproduktbereich Informationen über das Produkt Waschmaschine eingetragen, abgespeichert und verarbeitet. Die Funktionalität „Produkt-Administration“ dient der Verwaltung der beschriebenen Bauteile und Objekte einer WM, das heißt, alle zu demontierenden Objekte werden in diese Tabellen eingetragen und abgespeichert.

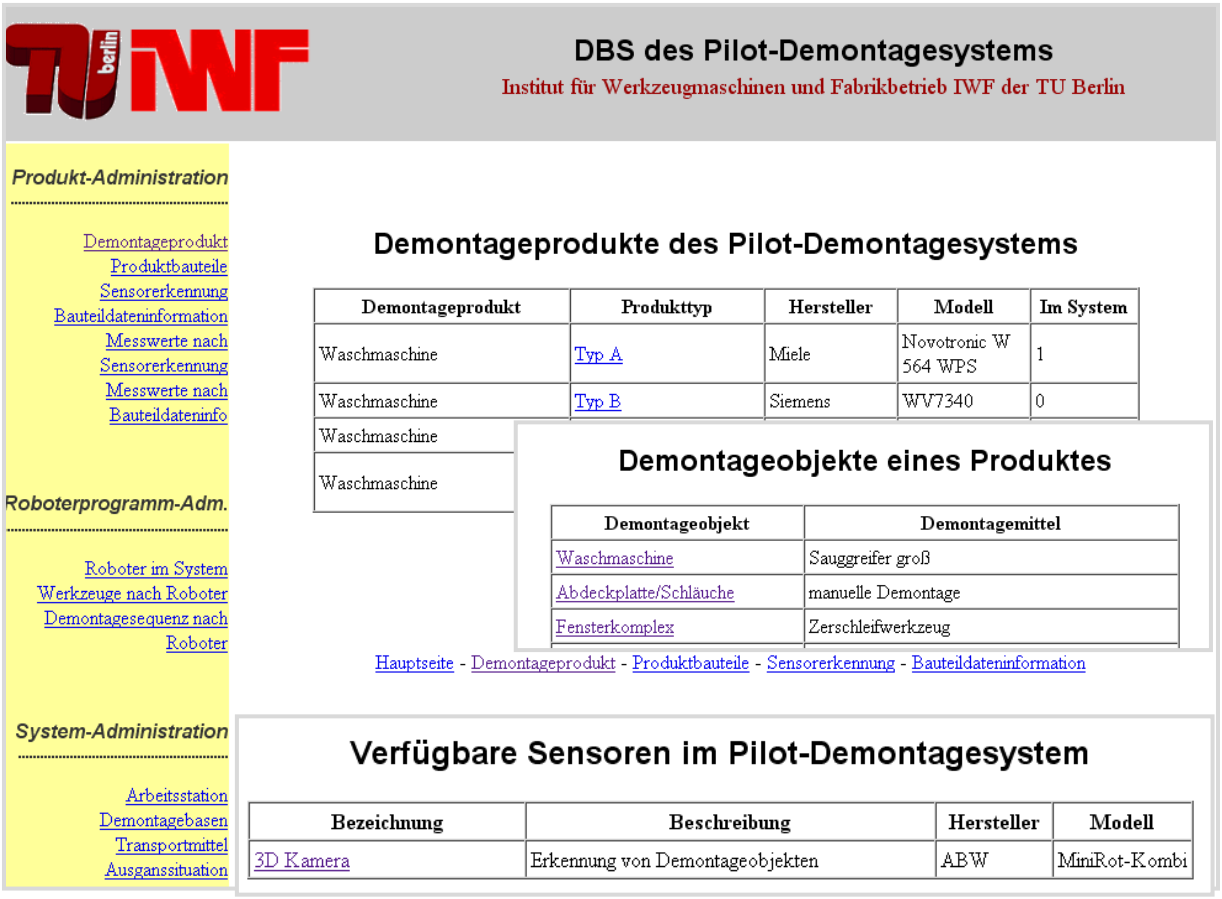


Bild 6.12: Auszug aus dem Demontageproduktbereich der Webanwendung

Der *Roboterprogrammbereich* bezieht sich auf die Industrieroboter, Werkzeuge und Demontagesequenzen, die notwendig sind, um die automatisierte Handhabung sowie die zerstörenden und nicht zerstörenden Demontageaufgaben an einer WM zu ermöglichen. Die Industrieroboterdaten beinhalten Informationen über die drei vorhandenen Industrieroboter des Pilot-Demontagesystems sowie den Dodekapod. Wichtige Informationen sind die interne Bezeichnung des Roboters (z. B. KUKA I) sowie dessen Typ und Beschreibung (z. B. KUKA KR 150/200, KR-C1 Steuerung). Die Werkzeugdaten enthalten die Information zur Identifizierung des Werkzeuges im System, Bezeichnung, Werkzeugnummer und Werkzeugwechselbasis, siehe **Kapitel 6.1**. Ein weiterer Aspekt des Roboterprogrammbereiches sind die Demontagesequenzen bzw. die Elementaranweisungen für die Realisierung der Demontageschritte. Hierbei sind alle notwendigen Befehle und ELA in der Datenbank gespeichert. Diese müssen in Bezug auf die Demontageobjekte verknüpft werden, damit eine Relation zwischen der Demontagesequenz und dem zu demontierenden Objekt existiert.

Der *Bereich Systeminformation* beinhaltet Informationen zum Betrieb des Pilot-Demontagesystems. Wichtige Beschreibungen sind die Ausgangssituation für die Demontageschritte im Pilot-Demontagesystem, die Systemsensoren, die Arbeitsstationen und das Transportmittel, siehe **Kapitel 6.1**. Für den Betrieb der Arbeitsstationen (AS) sind die Systemsensoren (Sensor/Aktoren) verantwortlich, wobei eine Ausgangssituation zum Betrieb einer AS definiert werden muss. Den Arbeitsstationen und den Transportmitteln werden Objektbasen zugewiesen, so dass die Demontagebasen berechnet werden können. Diese Demontagebasen werden aus der Transformation der Objekt-, der Arbeitsstation- und der Transportmittelbasis berechnet, siehe **Kapitel 5.5**.

In **Tabelle 6.6** und **Tabelle 6.7** sind die Entitäten aufgelistet, die im Pilot-Demontagesystem für die Realisierung des TOPS notwendig sind. Mit Hilfe dieser Entitäten wurden die Tabellen der DB erstellt, wobei diese neben den Definitionen von Attributen eventuelle Integritätsbedingungen beinhalten. Wichtig bei der Erzeugung einer relationalen DB sind die Schlüssel zur Identifizierung der Beziehungen zwischen Entitäten, wobei ein eingetragener Primärschlüssel die Identifizierung eines Objektes der reellen Welt darstellt.

Tabelle 6.6: Entitäten des Pilot-Demontagesystems (Teil 1)

Entität		Beschreibung	Beispiel
Demontageproduktbereich	Sensor	Einheit, die phys. oder chem. Zustände erfasst und in ein verarbeitbares Signal umwandelt	3D-Bildverarbeitungs-system
	Messgröße	phys. oder chem. Zustand, der von einem Sensor erfasst werden kann	Seitenwand vorhanden
	Messwert	Ergebnis der Erfassung einer Messgröße	vorhanden
	Objektdaten-information	Beschreibung der Zusammenhänge von Messgrößen	Höhe, Breite und Tiefe
	Demontage-objekt	Bauteile oder -gruppen des Produkts	Seitenwand
	Demontage-produkt	Demontageprodukt im Pilot-Demontagesystem	WM
	Objektbasis	Demontagebasis eines Demontage-objektes	[X=0,Y=0,Z=0, A=0, B=0, C=0]

Tabelle 6.7: Entitäten des Pilot-Demontagesystems (Teil 2)

Entität		Beschreibung	Beispiel
Roboterprogramm-bereich	Roboter	Mehrzweckmanipulator	Roboter A
	Arbeitsraum	Raum- und Traglasteigenschaften eines IR	2 m^3, 150 kg
	Werkzeug	Endeffektoren eines IR	Zerschleifwerkzeug
	Demontage-sequenzen	Demontageoperationen für einzelne Demontageprozesse	Demontage der Frontwand
	Elementar-anweisung	Darstellung einer Demontageoperation mit Hilfe von Elementaranweisungen	PTP_Base_Ohne Ereignis
	Werkzeug-wechselbasis	Koordinatenbasis für einen Werk-zeugwechsel	[X=0, Y=0,Z=0, A=0, B=0, C=0]
Systeminformationsbereich	Arbeitsstation	manueller oder automatisierter Arbeitsbereich im PDS	AS-2
	Ausgangs-situation	Beschreibung des Istzustands einer Arbeitsstation und des Demontage-produktes vor einer Demontage-operation	AS besetzt
	Arbeitsstations-basis	Koordinatenbasis einer Arbeitsstation	[X=2573,029, Y=-447,9890, Z=696,7180, A=3,099474, B=0,001239, C=-3,137730]
	Transportmittel	Hilfsmittel zur Beförderung bzw. Be-wegung eines Demontageproduktes	Spannpalette
	Transportmittel-basis	Koordinatenbasis eines Transport-mittels	[X=938,9150, Y=154,3100, Z=-174,406, A=1,570796, B=1,570796, C=-0,002061]
	Demontagebasis	notwendige Koordinatenbasis zur Realisierung der Demontageoperation	[X=1641,3000, Y=-253,6000, Z=869,4000, A=105,3697 B=-89,7676, C=162,0989]

Im **Anhang 6** wird das Modell der DB gezeigt, das alle Entitäten und alle notwendigen Beziehungen beinhaltet, wobei die drei geforderten Bereiche (Produkt, Roboterprogramm, Systeminformation) berücksichtigt sind.

6.3.2 Programmierleitstand

Der Programmierleitstand wurde mit Hilfe der objektorientierten Programmiersprache „Java“ realisiert. Hierbei ist die Beschränkung auf eine objektorientierte Sprache hilf-

reich, da sich objektorientierte Programmierung für die Verarbeitung von Daten aus relationalen Datenbanken am besten eignet. Darüber hinaus hat die Programmiersprache „Java“ den entscheidenden Vorteil, dass „Java“ vollkommen plattformunabhängig ist und somit der realisierte Programmierleitstand auf alle Computersysteme mit installierter „Java Virtual Machine“ ausführbar ist [Tio-09]. Des Weiteren werden für die Entwicklung der grafischen Benutzeroberfläche (GUI) die Programmierschnittstellen AWT (Abstract Window Toolkit) und SWING verwendet, da diese Programmierschnittstellen die Realisierung von grafischen Benutzeroberflächen unterstützen [Gea-99a, Gea-99b]. Bei der Verwendung dieser Technologie findet eine klare Funktionsteilung zwischen den Komponenten AWT und SWING statt. Grundkomponente ist dabei das Paket AWT, das standardisierte, einfache und grundlegende GUI-Komponenten und -Routinen zur Ereignisbehandlung zur Verfügung stellt. SWING stellt eine optionale Erweiterung von AWT dar und bietet zusätzlich erweiterte, grafisch aufwendiger gestaltete GUI-Komponenten für die Realisierung des Programmierleitstands.

Die grafische Benutzeroberfläche besteht für die Interaktion des Benutzers mit dem Programmier- und Steuerungssystem zum einen aus dem vom Programmkern initialisierten Hauptfenster und zum anderen aus den zusätzlichen Komponenten, die von dem Plug-in-Manager zur Verfügung gestellt werden. Das Hauptfenster bildet in seiner Umsetzung die Basis für den Programmierleitstand und somit für das gesamte System. Mit Hilfe des Hauptfensters werden für den Benutzer nicht sichtbare Funktionen wie z. B.:

- die Datenbankschnittstelle und
- die Schnittstelle zu der Speicherprogrammierbaren Steuerung

sowie sichtbare Funktionen wie z. B.:

- das Hauptmenü,
- die Symbolleisten und
- die grafische Aufbereitung der Plug-ins

bereitgestellt. Die nicht sichtbaren Funktionen sind für die Integration des Programmierleitstands in das Gesamtkonzept des technologieorientierten Programmier- und Steuerungssystems notwendig. Dagegen werden die sichtbaren Funktionen vom Bediener des Programmierleitstands genutzt, um die Demontage

einer Waschmaschine im Pilot-Demontagesystem auf der Basis des technologischen Wissens des Anwenders zu programmieren und zu steuern.

Für die Anpassung der nicht sichtbaren Funktionen wurde der Menüpunkt „Einstellungen“ realisiert, siehe **Bild 6.13**. Innerhalb dieser Funktion werden die Schnittstelle zu dem Datenbanksystem und die OPC-Schnittstelle zu der Speicherprogrammierbaren Steuerung konfiguriert, siehe **Kapitel 6.4**. Darüber hinaus ermöglicht der Plug-in-Manger es, einzelne Plug-ins zu aktivieren bzw. zu deaktivieren. Dadurch können Plug-ins abgeschaltet werden, so dass sie für den Benutzer nicht mehr zur Verfügung stehen. Weiterhin kann die Sprachauswahl getroffen werden.

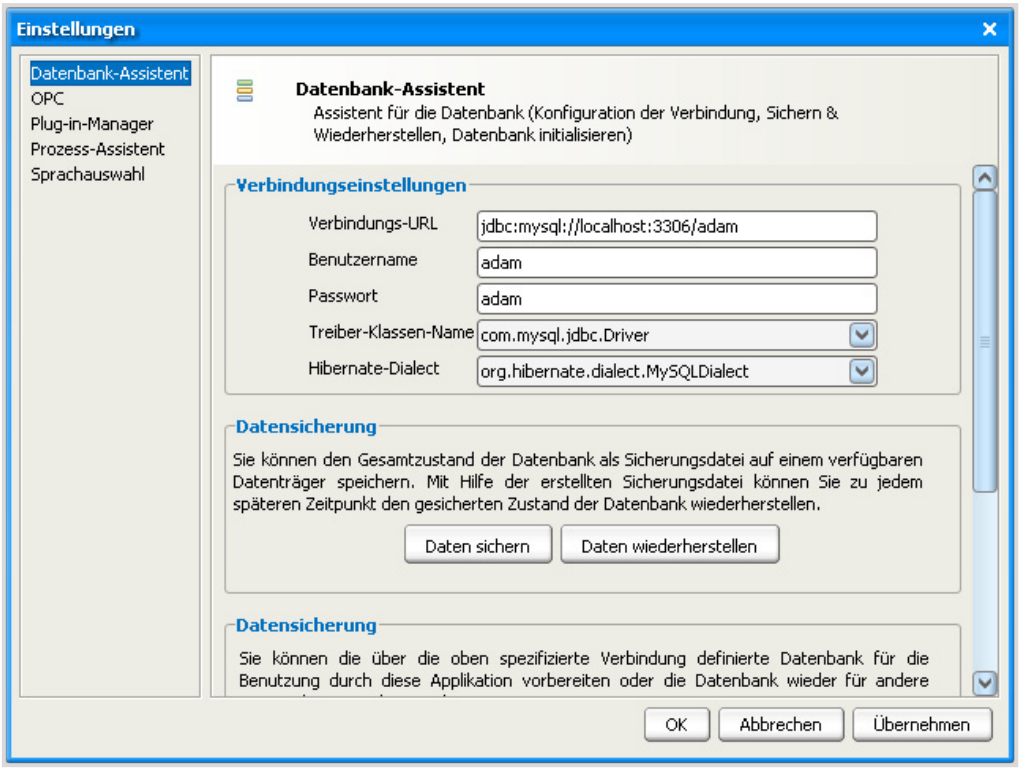


Bild 6.13: Menüpunkt „Einstellungen“ des Programmierleitstands

Das Hauptmenü bietet einen direkten Zugriff auf das Hilfesystem und ermöglicht es, dem Benutzer spezifische Symbolleisten sowie Fenster der Plug-ins ein- oder auszublenden, vgl. **Bild 6.14**. Der Benutzer kann so eine nach seinen Wünschen gestaltete Anwendungsansicht erstellen und speichern sowie diese als Standard definieren. Des Weiteren stellt das Hauptmenü Basismenüpunkte für die Anwendung des Programmierleitstands zur Verfügung. Die Symbolleiste bietet einen Schnellzugriff auf die Kernoperationen und -aufgaben der Plug-ins. Für die Verwendung

kann der Benutzer auch die verschiedenen Teilsymbolleisten nach Belieben sortieren, um die Anwendungsansicht in einem noch größeren Umfang individuell zu gestalten. Innerhalb des Hauptfensters stellen die Plug-ins interne Fenster für die Darstellung ihrer Inhalte bereit. Für die Konfiguration einer individuellen und an den Verwendungszweck angepassten Anwendungsansicht können diese Fenster aktiviert oder deaktiviert sowie innerhalb des Hauptfensters positioniert werden.

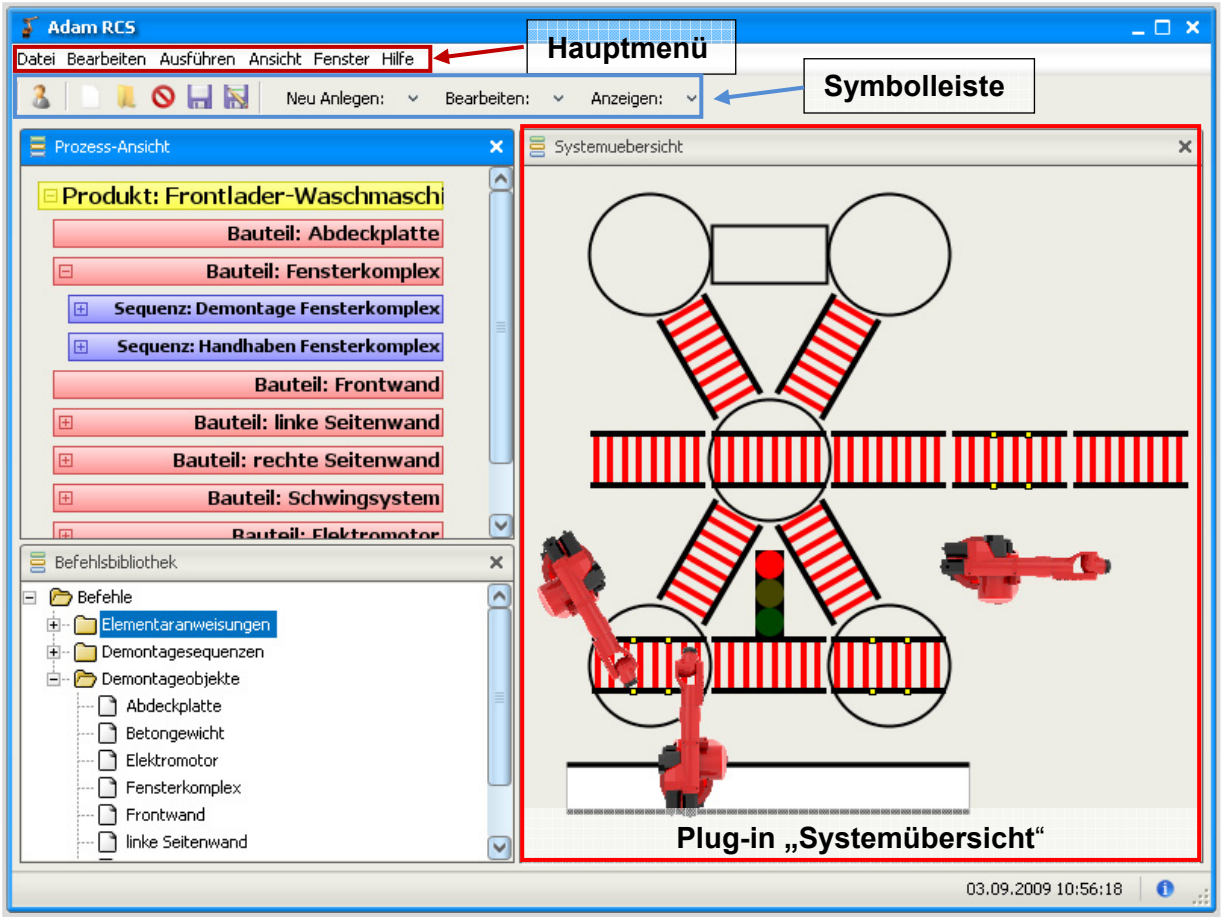


Bild 6.14: Hauptfenster des Programmierleitstands

Um den Einstieg in die Anwendung zu erleichtern, wurde der Prozess-Assistent entwickelt, der dem Benutzer ermöglicht, ein Produkt oder Bauteil für die Demontage zu öffnen oder ein neues Produkt anzulegen, vgl. **Bild 6.15**. Hierbei öffnet sich der Prozess-Assistent automatisch nach dem Start des Programmierleitstands.

Nach der Auswahl eines Demontageproduktes stehen dem Bediener des Pilot-Demontagesystems zwei weitere Funktionen, nämlich

- die Befehlsbibliothek und
- die Prozessansicht

für die Programmierung und Steuerung des Systems zur Verfügung, siehe **Bild 6.14**. Die Befehlsbibliothek umfasst die Darstellung aller Elementaranweisungen, Demontagesequenzen und Demontageobjekte in einer Baumansicht, die in dem Datenbanksystem dokumentiert sind, vgl. **Kapitel 6.3.1**. Die einzelnen Objekte können betrachtet, geändert und gelöscht werden oder per Drag-and-Drop-Funktion in die Prozessansicht übertragen werden. Die Funktion „Prozessansicht“ ist das Kernstück der Anwendung für die Programmierung und Steuerung des Demontagesystems. Durch diese Komponente werden alle Demontagesequenzen als leicht verständliches Flussdiagramm dargestellt.

Bild 6.15: Produkt-Assistent des Programmierleitstands

Für die Programmierung und Steuerung können alle Demontagesequenzen inklusive der dazugehörigen Elementaranweisungen und Steuerungsinformationen geladen, bearbeitet und gespeichert werden. Darüber hinaus ist die Reihenfolge aller Objekte mit Hilfe der Drag-and-Drop-Funktion veränderbar.

Um mit der Prozessansicht arbeiten zu können, muss zuerst ein neuer Demontageplan angelegt oder ein bereits bestehender Plan geöffnet werden. Dies geschieht entweder über den Prozess-Assistenten, die Symbolleiste oder das Menü. Ist ein Produkt geöffnet, steht eine Vielzahl an Möglichkeiten zur Verfügung, um auf den Demontageplan des Produkts Einfluss zu nehmen. Auf Basis des geöffneten Demontageplans kann der gesamte Plan auf den Ebenen

- Demontageobjekte,
- Demontagesequenzen und
- Elementaranweisungen

verändert werden. Das heißt, es können Elemente geöffnet, bearbeitet sowie aus dem Plan entfernt und zu dem Plan hinzugefügt werden. Das Bearbeiten der Elemente geschieht direkt in der Prozessansicht bzw. mit Hilfe der Befehlsbibliothek. Durch einen Klick auf den Prozesspfeil einer Elementaranweisung öffnet sich ein Fenster, in dem alle Einstellungen einer Elementaranweisung bearbeitet werden können. Darüber hinaus ist es möglich, mehrere Elementaranweisungen gleichzeitig zu öffnen, um Einstellungen zu vergleichen oder zu übernehmen, siehe **Bild 6.16**.

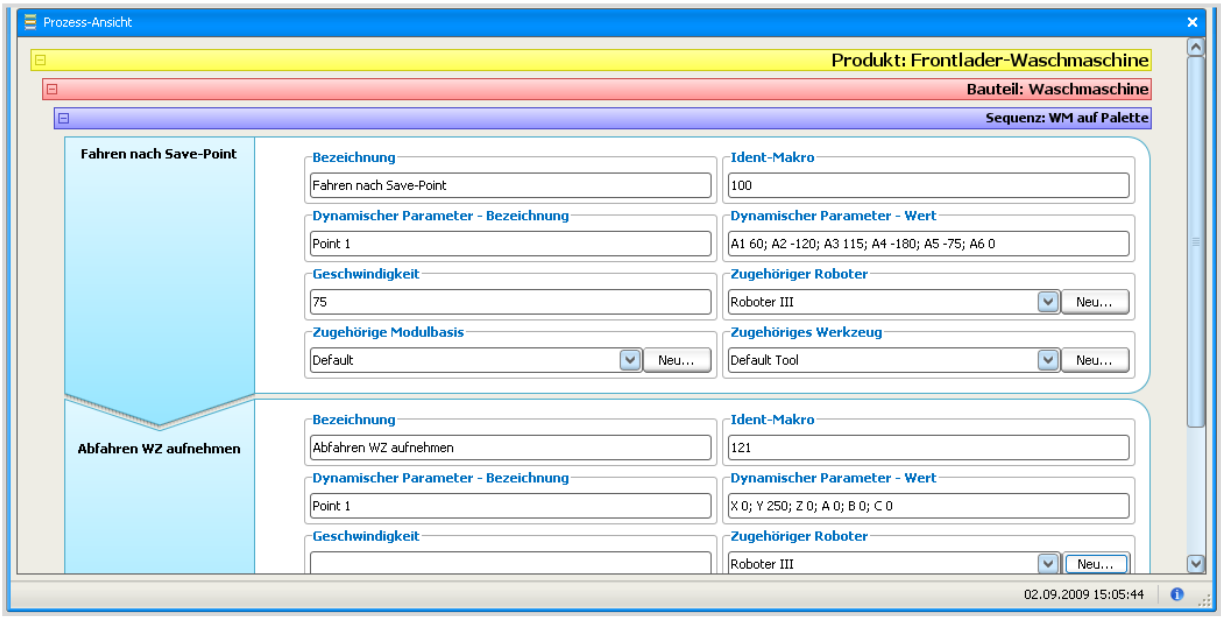


Bild 6.16: Manipulation von Elementaranweisungen

Des Weiteren kann die Reihenfolge aller Elemente per Drag-and-Drop-Funktion geändert werden. Zur optischen Unterstützung wird beim „Bewegen eines Elements“ das zu verschiebende Element orange eingefärbt, damit es von den anderen Komponenten unterschieden werden kann. Demontagesequenzen und -produkte können jeweils durch einen einfachen Klick zugeklappt werden, um die Ansicht übersichtlicher zu gestalten. Beim Zuklappen des Elements ändert sich das Minussymbol links auf dem farbigen Balken in ein Plus. Hierdurch ist gut zu erkennen, dass es noch weitere untergeordnete Elemente gibt. Neue Elemente können eingefügt werden, indem diese aus der Befehlsbibliothek per Drag-and-Drop-Funktion in die Prozessansicht „gezogen“ werden. Hierbei wird das neue Element zur besseren Erkennbarkeit wieder mit der Farbe orange eingefärbt. Nachdem das neue Element eingefügt wurde, stehen für dieses Element die gleichen Bearbeitungsoptionen zur Verfügung wie für alle anderen Elemente.

Nach der vollständigen Erstellung des Demontageplans wird der Plan an die Speicherprogrammierbare Steuerung übertragen. Für die Übertragung des Demontageplans wurde der OPC-Player entwickelt, der alle Elementaranweisungen inklusive der Steuerungsinformationen über die OPC-Schnittstelle an das Demontagesystem übermittelt. Hierfür sind eine Schaltfläche in der Symbolleiste und der Menüpunkt „Ausführen - Start“ vorgesehen. Diese Funktion leitet den gesamten Demontageplan an die Prozesssteuerung weiter. Des Weiteren ist der OPC-Player mit den Funktionen „Schrittweise ausführen“, „Stopp“ und „Pause“ ausgestattet. Diese Funktionen werden insbesondere bei einem Testbetrieb des Pilot-Demontagesystems benötigt. Die Funktion „Schrittweise ausführen“ ermöglicht die Übermittlung einer definierten Anzahl von Elementaranweisungen. Im Unterschied dazu versetzen die Funktionen „Stopp“ und „Pause“ den Bediener in die Lage, die Übertragung des Demontageplans permanent oder temporär zu unterbrechen.

Die Funktion „Speichern“ ermöglicht es alle Anpassungen, die am Demontageplan vorgenommen wurden, in das Datenbanksystem zu übertragen. Hierbei wird geprüft, ob veränderte Elemente mehrfach verwendet werden, um ein Überschreiben zu verhindern. Falls ein solches verändertes Element mehrfach verwendet wird, stehen zwei Optionen zur Auswahl. Einerseits können die Änderungen für alle Stellen, an denen dieses Element verwendet wird, übernommen werden. Dies ist z. B. sinnvoll, um einen Tippfehler oder einen falschen Parameter zu korrigieren. Andererseits können die geänderten Elemente gesondert gespeichert werden, um die Änderungen nur für diese eine Stelle zu übernehmen. Dies ist hilfreich, falls nur ein einzelnes Element angepasst werden soll, ohne diese Änderungen auch in alle anderen Produkten zu übernehmen. Darüber hinaus ermöglicht die Funktion „Speichern unter“, einen gesamten Demontageplan unter einem anderen Namen zu speichern, so dass alle Elemente dupliziert werden. Damit ist dieser Demontageplan nach dem Speichern vollkommen unabhängig von anderen Plänen und kann somit beliebig geändert werden. Die Verwendung dieser Funktion empfiehlt sich besonders dann, wenn für ein Produkt ein Demontageplan erstellt werden soll, welcher große Ähnlichkeit zu einem bereits vorhandenen Plan aufweist.

6.4 Verwendete Schnittstellen

Die Komponenten des technologieorientierten Programmier- und Steuerungssystems wurden durch die Profibus DP-Master- und -Client-Schnittstelle sowie die Schnittstelle Openess Productivity Collaboration (OPC) vernetzt, die mit den Datennetztechniken Profibus und Ethernet arbeiten, siehe **Bild 6.17** [Uhl-08b]. Profibus DP (DP: Dezentrale Peripherie) ist für den effizienten Datenaustausch in der Feldebene konzipiert, wobei es für die Kommunikation zwischen den Komponenten Speicherprogrammierbare Steuerung (SPS) und Robotersteuerung genutzt wird. Die SPS als Master regelt den zyklischen Datenverkehr und tauscht dabei in einem definierten Nachrichtenzyklus die Prozessdaten mit der Robotersteuerung aus. Die SPS liest zyklisch die Eingangsinformationen der Robotersteuerung (Slave) und schreibt die Ausgangsinformationen zyklisch an den Slave.

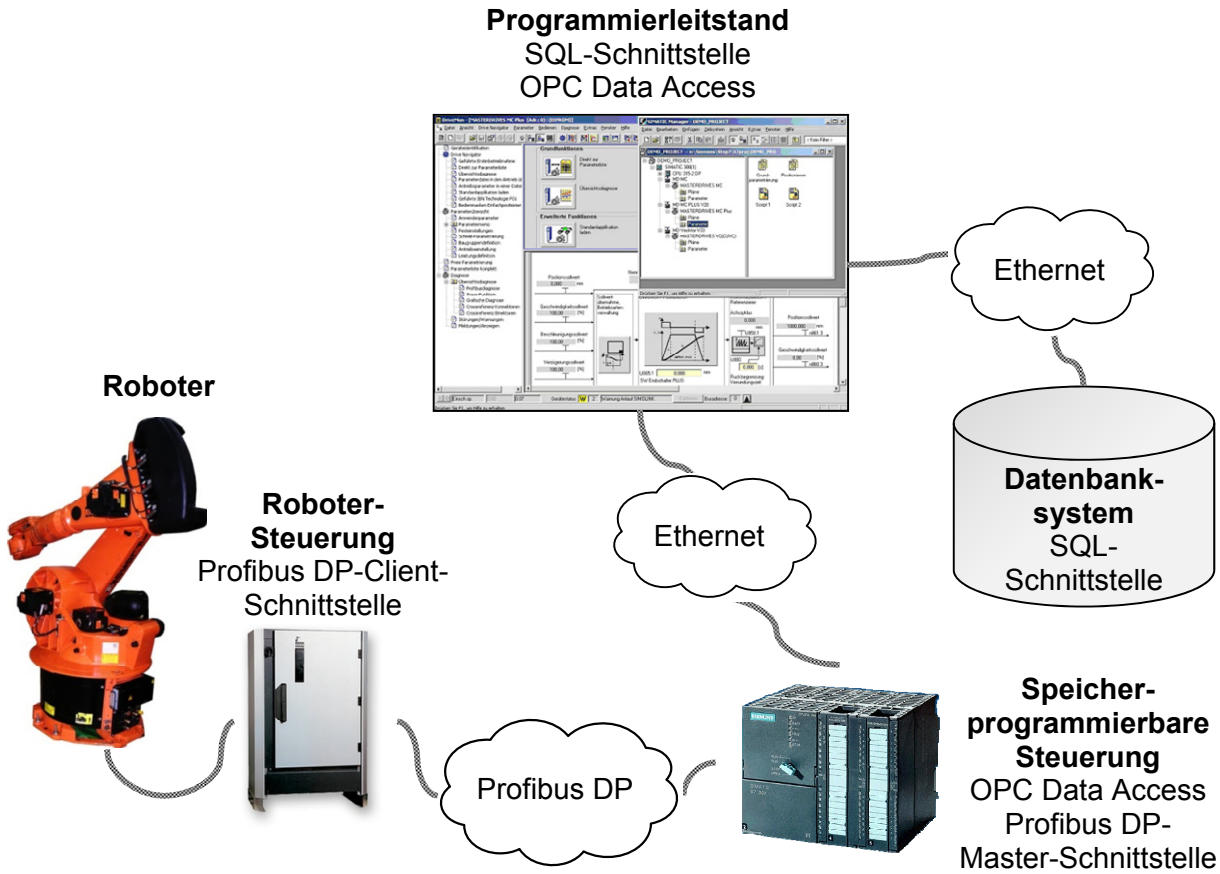


Bild 6.17: Verwendete Schnittstellen

OPC definiert eine offene Schnittstelle, über die PC-basierte Softwarekomponenten Daten austauschen können, wobei die OPC-Data-Access-Spezifikation genutzt wird, da diese sich als ein weltweiter herstellerunabhängiger Standard zum Lesen,

Schreiben und Beobachten von Prozessdaten etabliert hat. Die OPC-Schnittstelle basiert auf dem Prinzip der Zusammenarbeit zwischen initiierendem Prozess (stellt Anfragen, erteilt Aufträge) und reagierendem Prozess (bearbeitet Anfragen und Aufträge), also einer Client-/Serverstruktur. Die Speicherprogrammierbare Steuerung als OPC-Server stellt die steuerungsrelevanten Informationen für den Programmierleitstand als OPC-Client bereit.

Eine wichtige Aufgabe des PLS ist die Verarbeitung der benötigten Information für den Demontageplan. Um auf solche allgemeingültigen Daten permanent zugreifen zu können, wird eine SQL-Schnittstelle (Structured Query Language) eingesetzt. Durch die Nutzung der SQL-Schnittstelle ist es möglich, Manipulationen an einer relationalen Datenbank durchzuführen.

6.5 Implementierung des technologieorientierten Programmier- und Steuerungssystems

Implementierung bezeichnet die Installation eines Programms und dessen Anpassung an das Rechnersystem. In diesem Kapitel werden die notwendigen Arbeitsschritte für die Umsetzung des technologieorientierten Programmier- und Steuerungssystems zu einem lauffähigen Gesamtsystem beschrieben, die über die Beschreibung der Realisierungen in Kapitel 5 hinausgehen.

Als Bindeglied zwischen der Speicherprogrammierbaren Steuerung (Simatic S7-400) und dem Programmierleitstand wurde der OPC-Server des Softwarepakets „Simatic.Net“ der Firma Siemens eingesetzt. Die Einrichtung des OPC-Servers erfolgte in vier Schritten:

- In der Hardwarekonfiguration der SPS Simatic S7-400 wurde auf Baugruppenplatz 1 ein Objekt namens SimaticNet.OPC.Server hinzugefügt.
- Dieselbe Anordnung, die in der SPS konfiguriert wurde, wurde im Komponenten-Konfigurator hergestellt, so dass die CPU-412-2 auf Steckplatz 3 und der OPC-Server auf Steckplatz 1 eingefügt sind.
- Im dritten Schritt wurde eine Verbindung zu der projektierten CPU hergestellt.
- Abschließend wurden alle geänderten Baugruppen und Netzeinstellungen gespeichert, übersetzt und neu geladen.

Für die lokale Anwendung des Datenbanksystems mit einer Web-Anwendung als Bedienoberfläche müssen diverse Programme installiert werden. Diese Programme können entweder nacheinander oder mit einem vorkonfigurierten Softwarepaket installiert werden. Zur Erleichterung der Integration des DBS wurde das Programmpaket „XAMPP“ verwendet. Dieses Programmpaket ist eine Zusammenstellung von frei verfügbarer Software (General Public License) und unterstützt die einfache Installation eines „Apache-Webservers“ mit dem DBMS „MySQL“ sowie den Skriptsprachen „PHP“ und „Perl“ unter dem Betriebssystem „Windows“ [Apa-09].

Um die Beschreibung der Bewegungsoperationen eines Roboters im Arbeitsraum für den Bediener zu vereinfachen, wurden den Demontageobjekten unterschiedliche Demontage-Basen zugewiesen, die eine Hilfsebene beschreiben, siehe **Bild 6.18**. Die Lage und Orientierung einer Demontageobjektbasis ermöglichen die einfache Manipulation der Roboterprogramme.

Bild 6.18: Beispiele für Hilfsebenen

Die Flexibilität des technologieorientierten Programmier- und Steuerungssystems kann erhöht werden, wenn eine Demontageobjektbasis auf der Grundlage:

- der Weltbasis (Nullpunkt des Produktionssystems, Inertialsystem),
- der Arbeitsstationsbasis (Nullpunkt der Arbeitsstation) und
- der Transportmittelbasis (Nullpunkt des Transportmittels)

berechnet wird, vgl. **Bild 6.19**. Die Basisverschiebung der Arbeitsstationsbasis ist auf die Weltbasis bezogen, wogegen die Transportmittelbasis auf die Arbeitsstationsbasis und die Demontagetageobjektbasis auf die Transportmittelbasis bezogen ist. Daraus resultiert eine kaskadierte Demontageobjektbasis, die sich indirekt über die

Arbeitsstations- und Transportmittelbasis auf die Weltbasis bezieht. Durch die Verwendung von kaskadierten Basissystemen wird die Unabhängigkeit der Demontageobjektbasis vom Produktionssystem erhöht. Bei der Integration des Programmier- und Steuerungssystems in ein Produktionssystem werden ausschließlich die Arbeitsstation und das Transportmittel vermessen.

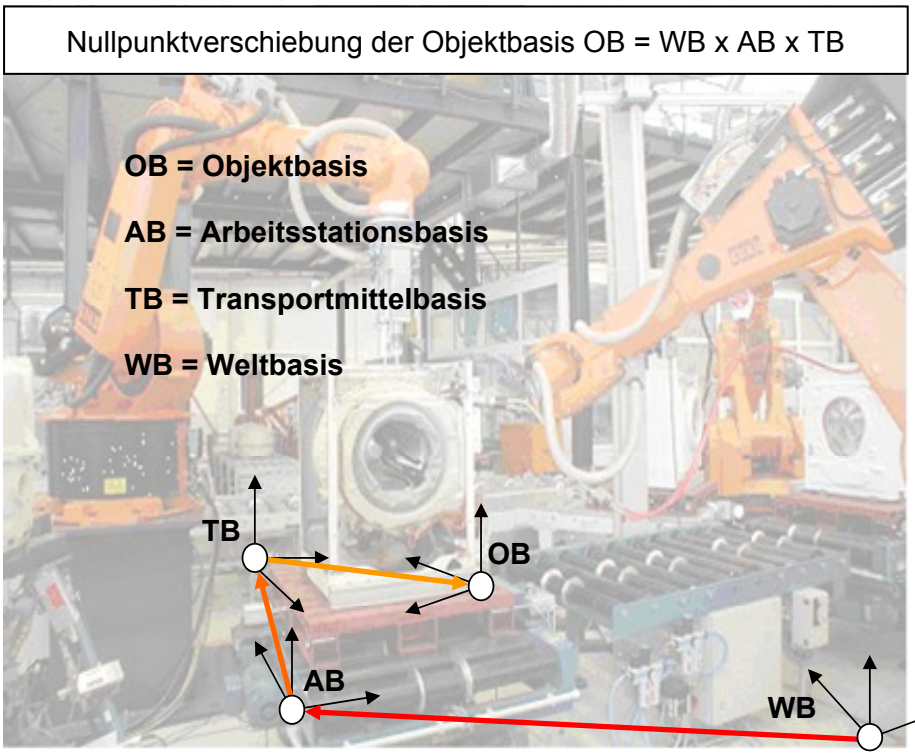


Bild 6.19: Nullpunktverschiebung der Objektbasis

Die komplexe Umwandlungs- bzw. Berechnungsprozedur wurde mit Hilfe der Anwendung „Matlab“ (Matrix Laboratory) des Unternehmens „The MathWorks“ realisiert. Darüber hinaus wurde die „Robotics Toolbox“ von Peter Corke verwendet, die eine Vielzahl von Matlab-Funktionen zur Bearbeitung von Aufgaben im Bereich Robotik bereitstellt [Cor-07].

7 Erprobung und Bewertung des technologieorientierten Programmier- und Steuerungssystems

7.1 Erprobung des Systems

Für die Erprobung des realisierten technologieorientierten Programmier- und Steuerungssystems wurden die vier Komponenten

- Robotersteuerung,
- Speicherprogrammierbare Steuerung,
- Datenbanksystem und
- Programmierleitstand

zusammengeführt und erprobt.

Da zur Demonstration der Produktflexibilität des Pilot-Demontagesystems das Produktspektrum um das Produkt „Verbrennungskraftmotoren“ erweitert wurde, waren zusätzliche informationstechnische Anpassungen erforderlich als die notwendige Anpassung der Komponenten Robotersteuerung und Speicherprogrammierbare Steuerung sowie der Realisierung der Komponenten Datenbanksystem und Programmierleitstand [Sel-06a, Sel-06b]. Aufgrund dieser realisierten Erweiterung waren bereits 60 Byte des Eingangsbereichs der Robotersteuerung projektiert. Bei einem gesamten Eingangsbereich von maximal 128 Byte (1024 boolesche Eingänge) standen nur weitere 68 Byte zur Verfügung, wobei für die Übertragung eines vollständigen Zeileneintrages 87 Byte erforderlich sind. Weiterhin wurde die WinCC-Oberfläche für die Ausführung eines Demontageplans erweitert, so dass Steuervariablen der Speicher-programmierbaren Steuerung mit Anzeigeeigenschaften für die Erprobung verbunden wurden, siehe **Bild 7.1.**

Bei dem Transfer der Daten zwischen der Zellensteuerung und den KUKA-Robotern traten Schwierigkeiten auf, die mit der unterschiedlichen Verarbeitung von Binärdaten begründbar sind. In einer SPS ist ein Byte eine Einheit von 8 Bits, wobei für die Prozess-Datenverarbeitung einzelne Bits in Byte angesprochen werden müssen. Weiterhin werden sie vom niedrigstwertigsten (LSB) zum höchstwertigsten Bit (MSB) geordnet. Hingegen wurde beim Empfang der Daten auf der KR-C1-Steuerung von den KUKA-Robotern eine Verschiebung von Bitwertigkeiten festgestellt. Die Reihen-

folge vom LSB zum MSB ist nicht linear, sondern versetzt. Da diese Verdrehung nachvollziehbar und immer gleich ist, lässt sich dieses Problem lösen, indem die entsprechenden Bits in einem temporären Feld sortiert werden und zur Auswertung an der richtigen Stelle zur Verfügung stehen. Hierzu wurde ein Modulationsprogramm („transfer_pb“) für die Robotersteuerung geschrieben.

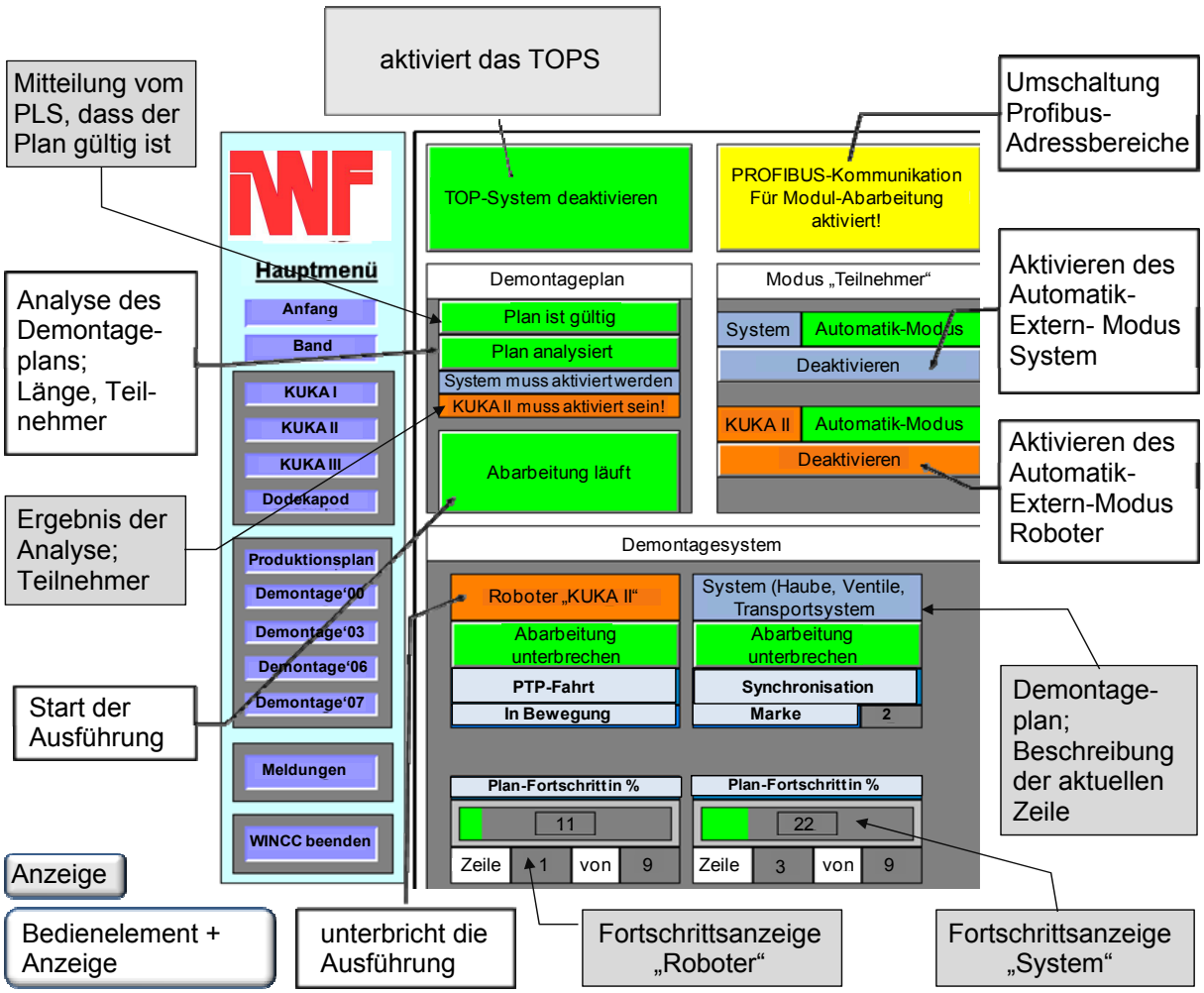
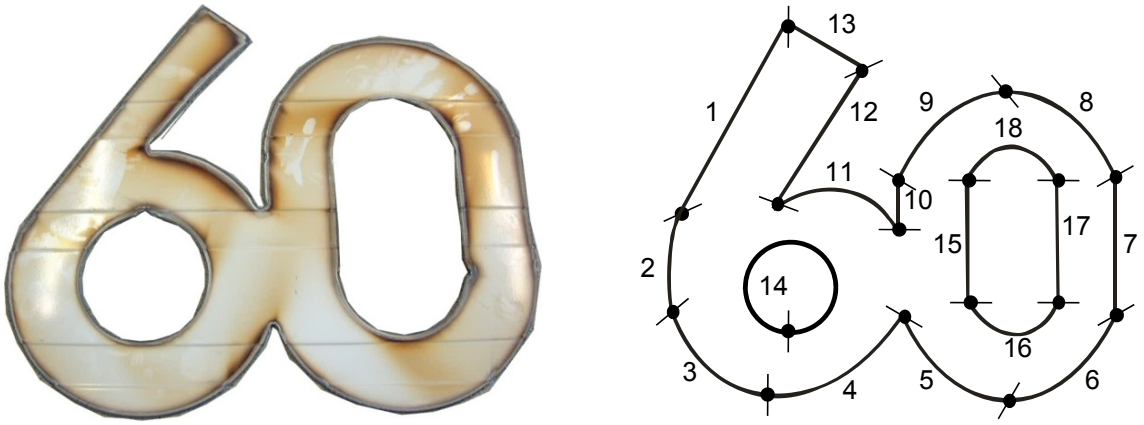


Bild 7.1: Erweiterte Bedienoberfläche in WinCC–Steuerung der Demontage

Um die Leistungsfähigkeit und die Bedienerfreundlichkeit des technologieorientierten Programmier- und Steuerungssystems nachzuweisen, wurden die bisher realisierten Demontageschritte des Pilot-Demontagesystems erprobt. Aufgrund des umgesetzten Konzepts wurde die Flexibilität der Demontageabläufe verbessert. Die Produktvariantenflexibilität konnte erhöht werden, so dass unterschiedliche Produkttypen und -varianten in beliebiger Reihenfolge demontiert werden können. Des Weiteren wurde die Möglichkeit geschaffen, das technologische Wissen des Bedieners bei der Planung der Demontageschritte zu berücksichtigen. Mit Hilfe der Mensch-Maschine-

Schnittstelle des Programmierleitstands können zu einem voreingestellten Demontageschritt Veränderungen durchgeführt werden. Für die Darstellung der Flexibilität wurde der Demontageschritt „Trennen einer Seitenwand“ mit dem Plasmazerstrahlwerkzeug durch den Bediener so angepasst, dass anstatt eines rechteckigen Ausschnitts der Seitenwand die Zahl 60 herausgetrennt wurde, vgl. **Bild 7.2**.



a) Ausschnitt b) Skizze

Bild 7.2: Zahl 60 ausgeschnitten aus der Seitenwand einer Waschmaschine

Das Heraustrennen der Zahl 60 wurde im Wesentlichen durch die Kombinationen der Elementaranweisungen „Lin_Base_Ohne Ereignis“ (Nr. 1, 7, 10, 12, 13, 15, 17) und „Circ_Base_Ohne Ereignis“ (Nr. 2, 3, 4, 5, 6, 8, 9, 11, 14, 16, 18) realisiert. Diese beiden ELA werden dazu genutzt, das Plasmazerstrahlwerkzeug entlang einer Geraden und einer Kreisbahn zu bewegen.

7.2 Vergleich und Bewertung der Demontageprozesse vor und nach der Implementierung des Programmier- und Steuerungssystems

Für einen grundsätzlichen Vergleich des technologieorientierten Programmier- und Steuerungssystems vor und nach der Inbetriebnahme des Systems werden die Systeme differenziert betrachtet und technische und nichttechnische Aspekte definiert. Es werden folgende Kriterien genutzt:

- flexible Anpassung an unterschiedliche Aufgabenstellungen,
- geringe Qualifikationsanforderungen an die Bediener,
- einheitliche Bedienung und Programmierung sowie

- Einbindung in die Gesamtsteuerungsarchitektur.

Bei den Anwendern von Roboterprogrammiersystemen kann zwischen zwei Gruppen unterschieden werden, dem Programmierexperten und dem Endanwender, wobei zwischen den beiden Extremen beliebige Zwischenstufen denkbar sind [Dam-96]. Die Schnittstellen zu den Anwendern definieren den Umfang und die Leistungsfähigkeit bei dem Umgang mit dem Pilot-Demontagesystem. Der Endanwender verfügt im Extremfall über geringe oder keine Programmierkenntnisse und hat sehr wenig Erfahrung mit Robotern, wogegen der RC-Experte im Allgemeinen sehr gute Qualifikationen besitzt [Dam-96]. Die Stillstandszeit des Pilot-Demontagesystems während der Programmierung ist ein wichtiges Kriterium, sie umfasst die Dauer, in der das System nicht verfügbar bzw. funktionstüchtig ist. **Tabelle 7.1** zeigt den Vergleich der Systeme.

Tabelle 7.1: Vergleich der Systeme

● gut ◑ durchschnittlich ○ schlecht	**bisheriges Bediensystem**	**Technologieorientiertes Programmier- und Steuerungssystem**
flexible Anpassung an unterschiedliche Aufgabenstellungen	◑ eingeschränkt	● gegeben
geringe Qualifikationsanforderungen an die Bediener	○ RC-Experte	● Prozesswissen
einheitliche Bedienung und Programmierung	◑ teilweise vorhanden	● vorhanden
Einbindung in die Gesamtsteuerungs-architektur	○ nicht gegeben	● vorhanden

Das bisherige System wurde über die Standardschnittstellen der Robotersteuerung und das Prozessvisualisierungssystem WinCC der Firma Siemens bedient und erfordert somit einen hohen Qualifizierungsgrad. Dagegen wird das technologieorientierte Programmier- und Steuerungssystem (TOPS) über eine benutzerfreundliche Mensch-Maschine-Schnittstelle bedient. Aus den zur Verfügung stehenden Anwenderschnittstellen leiten sich die erforderlichen Kenntnisse ab. Bisher waren

umfangreiche Kenntnisse der Roboterprogrammierung erforderlich, wohingegen bei der Bedienung des TOPS nur das technologische Prozesswissen notwendig ist. Die Stillstandszeit bei der Inbetriebnahme des neuen Systems ist geringer als bisher, da der überwiegende Teil der Roboterprogrammierung offline erfolgt. Bei der Realisierung des TOPS wurde ausschließlich Software mit General Public License (GPL) verwendet, wodurch keine zusätzlichen Kosten entstanden. Die bestehende Hardwarestruktur konnte unverändert genutzt werden.

Da bei kleinen und mittleren Unternehmen die Flexibilität eine wichtige Systemeigenschaft darstellt, wird die Systemflexibilität anhand der folgenden zwei Beispiele dargestellt:

- Produktwechsel und
- Fehler während eines Demontageablaufs.

Weiterhin wurde bewertet, wie unterschiedlich sich das „System alt“ (vor Inbetriebnahme des technologieorientierten Programmier- und Steuerungssystems) und das „System neu“ (nach Inbetriebnahme des TOPS) bei demontagetypischen Situationen verhalten. Hierbei werden die zwei Szenarien sowie das Verhalten vor und nach der Inbetriebnahme des Systems beschrieben. Das erste Szenario repräsentiert eine Situation, bei der ein Höchstmaß an Flexibilität erforderlich ist. Durch die Modularisierung der Roboterprogramme wurde eine Differenzierung der Steuerung erreicht, wodurch eine effektivere Demontage realisiert wird, was im zweiten Szenario vorgestellt wird.

7.2.1 Produktwechsel

Im Pilot-Demontagesystem wurde die Demontage des Modells „SIWAMAT WXLM 1063“ der Firma Siemens realisiert. Aufgrund der Erweiterung des Produktspektrums wird zukünftig die Waschmaschine des Typs „Novotronic W 564 WPS“ der Firma Miele demontiert. Bei beiden Waschmaschinen handelt es sich um Frontladerwaschmaschinen, wie sie gegenwärtig in ähnlicher Grundkonzeption weltweit an mehreren Standorten in hoher Stückzahl produziert werden [Här-05]. Diese Waschmaschinen bilden eine Fertigungsfamilie, sie können ohne Umrüsten, ohne neue Werkzeuge und Vorrichtungen hintereinander demontiert werden [Ehr-07]. Sie unterscheiden sich aber unter anderem durch ihre Außenmaße, siehe **Tabelle 7.2**.

Tabelle 7.2: Vergleich der Außenabmessungen

	 SIWAMAT WXLM 1063	 Novotronic W 564 WPS
Höhe [mm]	852	850
Breite [mm]	600	595
Tiefe [mm]	590	580

„System alt“

Vom Bediener des Pilot-Demontagesystems wird mit Hilfe der technischen Typenbeschreibung die Baustruktur des Demontageobjektes festgestellt, so dass die entsprechenden Werkzeuge sowie die Verfahren definiert werden können. Hierbei werden dieselben Demontageprozesse eingesetzt, die bisher realisiert wurden, und nur die Koordinaten für die Werkzeuge geändert, da sich die Waschmaschinen im Wesentlichen in den Außenabmessungen unterscheiden. Beispielsweise wird der Startpunkt des Plasmazerstrahlwerkzeugs bei dem Demontageprozess „Seitenwand“ angepasst.

Daraufhin müssen von einem erfahrenen Roboterprogrammierer die zugehörigen Programme in der Robotersteuerung geändert werden, das mit dem Teach-in-Verfahren realisiert wird. Nach diesen Arbeitsschritten kann der Bediener die modifizierten, aber in ihrer Aufrufart nicht veränderten Programme über die Bedienoberfläche WinCC starten und die Demontage durchführen.

„System neu“

Der Bediener des Pilot-Demontagesystems wird bei der Bedienung des technologieorientierten Programmier- und Steuerungssystems über den Programmierleitstand den entsprechenden Waschmaschinentyp auswählen**,** vgl. **Bild 7.3**.

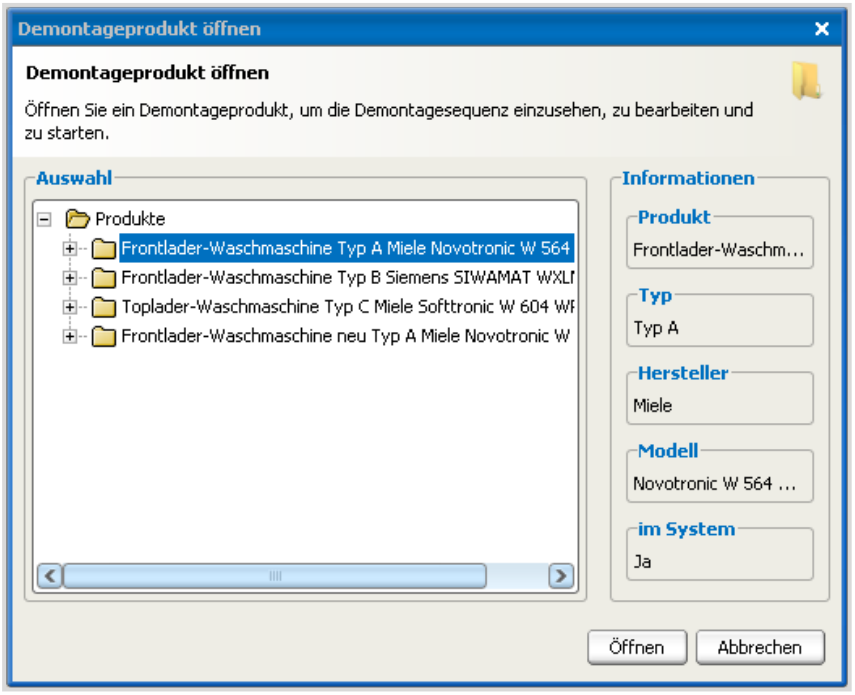


Bild 7.3: Typenwahl mit Hilfe der Mensch-Maschine-Schnittstelle

Aufgrund dieser Auswahl werden die entsprechenden Einträge in der Produktdatenbank gesucht und in den Programmierleitstand übertragen, sofern für diesen Waschmaschinentyp Datensätze existieren. Wenn bisher keine Produktdatensätze existieren, müssen sie einmalig erstellt werden, wobei dafür die CAD-Daten genutzt werden können. Darauffolgend wird der Demontageplan durch den Programmierleitstand generiert. Dieser Plan wird an die Zellenhauptsteuerung übertragen, woraufhin die erforderlichen Informationen an die Komponenten des Pilot-Demontagesystems übermittelt werden. Wurde dieses Objekt erfolgreich demontiert, kann das darauffolgende Demontageobjekt wiederum verschiedenen Typs sein, ohne eine Modifizierung der Roboterprogramme vorzunehmen. Aufgrund dieser effizienten Anpassung des Produktionssystems an das zu demontierende Produkt werden deutliche Zeit- und Kosteneinsparungen erreicht, siehe **Bild 7.4**.

Die Zeit- und Kosteneinsparung wird deutlicher, wenn sich nicht nur der Produkttyp ändert, sondern ein Produktwechsel außerhalb einer Fertigungsfamilie durchgeführt wird. Bei einem Produktwechsel z. B. vom Produkt Waschmaschine Typ „SIWAMAT WXLM 1063“ zum Produkt Verbrennungsmotor Typ „Mercedes-Benz Sprinter Ottomotor (2,3 Liter)“ muss die gesamte Roboterprogrammierung im „System alt“ neu durchgeführt werden. Dahingegen müssen beim „System neu“ der Programmierleitstand programmiert und die Datenbanken erstellt werden. Der Leitstand und die

Datenbank werden offline programmiert, wodurch keine Stillstandszeiten entstehen. Des Weiteren werden diese Anpassungen nicht von einem RC-Experten, sondern von einem Softwareentwickler durchgeführt, der meist in den Unternehmen verfügbar ist. Typischer Aufgabenbereich eines Softwareentwicklers ist die Erweiterung bzw. Anpassung von Software an die spezifischen Bedürfnisse eines Unternehmens.

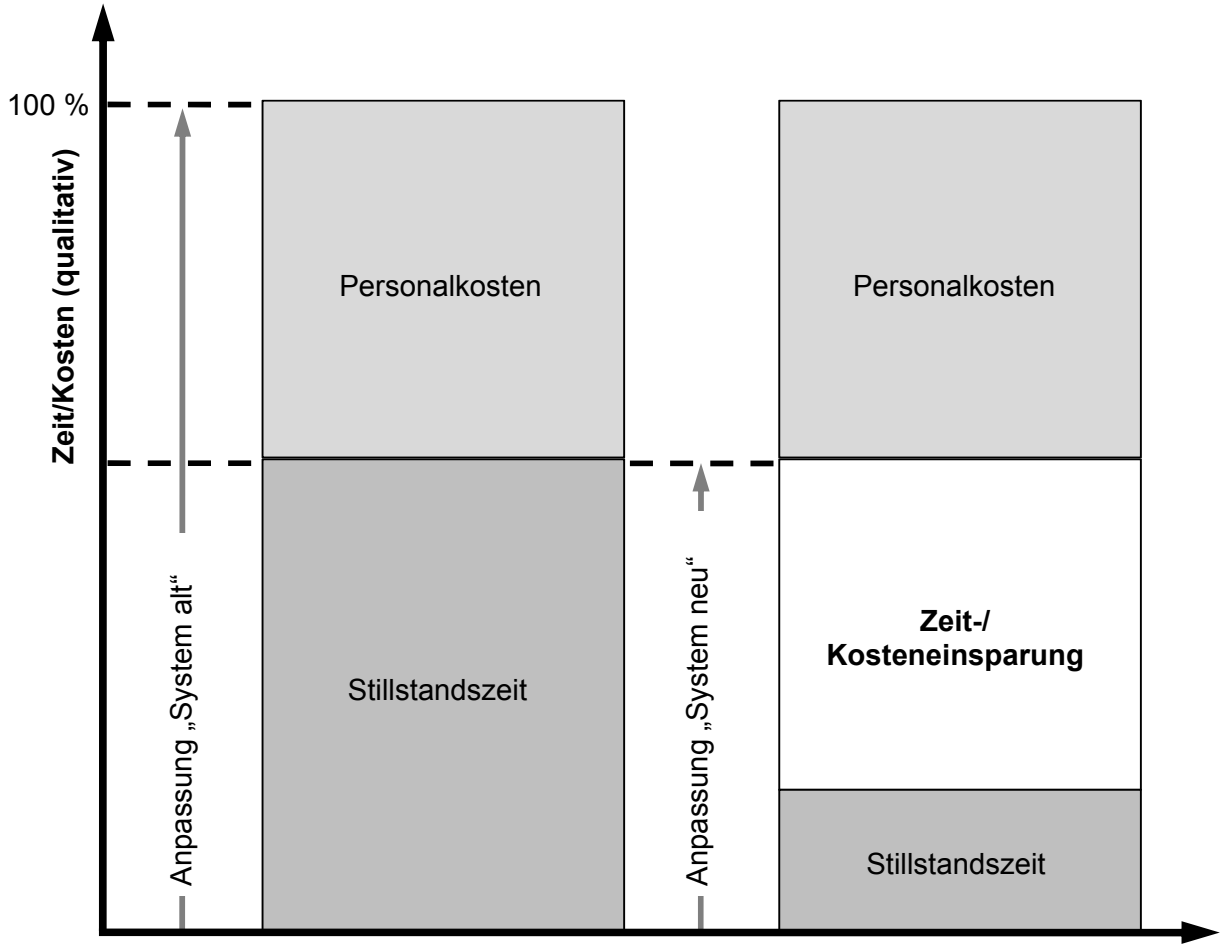


Bild 7.4: Qualitativer Vergleich des Aufwands aufgrund einer Produktanpassung

7.2.2 Fehler während des Demontageablaufs

Im gesamten Demontageablauf kann eine Vielzahl von Fehlern auftreten, exemplarisch wird hier die Nichtzündung des Schneidlichtbogens beim Arbeitsschritt „Demontage einer Seitenwand“ beschrieben. Für den Schneidprozess wird zunächst ein Pilotlichtbogen zwischen Düse und Katode mittels Hochspannung gezündet. Dieser energiearme Pilotlichtbogen bereitet durch erfolgende Ionisation die Strecke zwischen Plasmabrenner und Werkstück vor. Berührt der Pilotlichtbogen das Werk-

stück, wird durch eine automatische Leistungserhöhung die Zündung des Schneidlichtbogens eingeleitet [PSW-08].

Nach der Zündung des Schneidlichtbogens wird in der Robotersteuerung ein Interrupt ausgelöst und mit der Ausführung des Trennschnittes begonnen. Bei der Erprobung des Plasmazerstrahlwerkzeugs hat sich herausgestellt, dass nicht immer bei Abfahrt der Suchstrecke der Schneidlichtbogen zündet, da der elektrische Kontakt zwischen Pilotlichtbogen und Seitenwand nicht zustande gekommen ist. Für diesen Fehler wurde das Plasmazerstrahlwerkzeug mit einem zusätzlichen Näherungsschalter ausgestattet, um eine Kollision zu vermeiden, siehe **Bild 7.5**. Wenn der Schneidlichtbogen nicht gezündet wurde, muss für den erneuten Versuch ein anderer Punkt der Seitenwand in unmittelbarer Nähe angefahren werden, da aufgrund des Fehlversuches das Material an der Ursprungsstelle nicht mehr vorhanden ist und damit der Schneidlichtbogen nicht mehr gezündet werden kann.

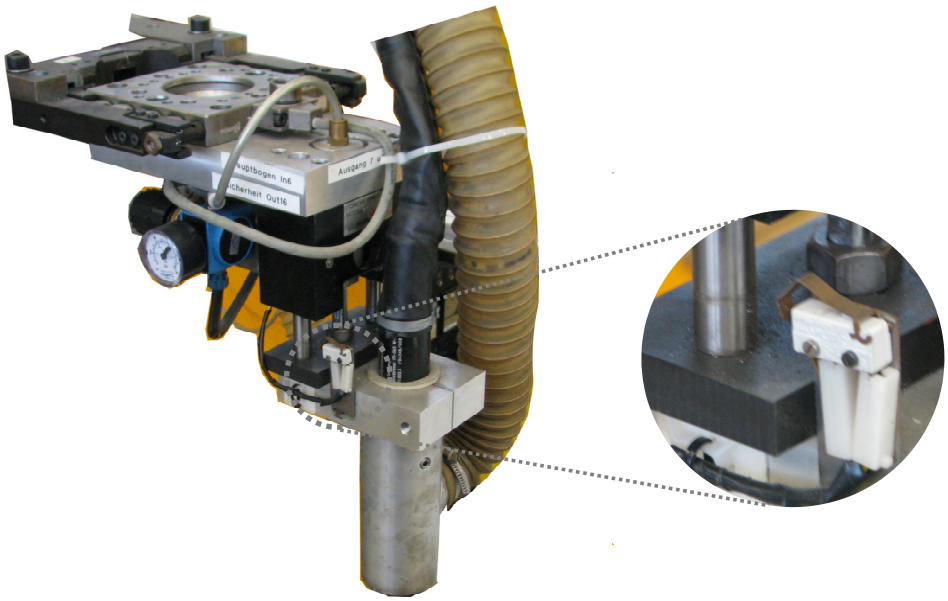


Bild 7.5: Plasmazerstrahlwerkzeug mit Näherungsschalter

„System alt“

Im bestehenden System ist das Heraustrennen einer Seitenwand bei einer WM ein vollständiges Programm, das auf der Robotersteuerung gespeichert ist. Das Programm wird aufgerufen und bis zum beschriebenen Fehler ausgeführt. Durch die Aktivierung des Näherungsschalters erfolgt der Abbruch der Programmausführung.

Aufgrund dieses Versuchs, das Plasmazerstrahlwerkzeug zu nutzen, ist eine Verschiebung des Startpunktes notwendig.

Um den Startpunkt zu verschieben, muss der Roboter den Automatik-Extern-Modus verlassen und ein Roboterprogrammierer muss im Programm den Startpunkt mittels Teach-in-Verfahren korrigieren. Anschließend kann der Automatik-Extern-Modus wiederhergestellt werden und der Programmschritt wird erneut ausgeführt.

„System neu“

Im neuen System unterteilt sich das Trennen einer Seitenwand in verschiedene Elementaranweisungen, die erst in ihrer Kombination ein Programm bilden, siehe **Kapitel 6.2.1**. All diese Elementaranweisungen werden von der Zellenhauptsteuerung vor ihrem Aufruf mit aktuellen Koordinatenparametern versorgt, wobei die Zellenhauptsteuerung die Daten wiederum vom Programmierleitstand erhält. Die Elementaranweisungen werden sequenziell aufgerufen. Während der Ausführung der Elementaranweisung „Lin_Base_Mit Ereignis Stopp wenn Sensor = true“ (Ident-Nummer 121) wird der Näherungsschalter aktiviert, siehe **Anhang 1**. Diese Information wird mit Hilfe der Zellenhauptsteuerung an den Programmierleitstand übertragen. Da dieser Fehler bekannt ist, wurde ein Steuerungsalgorithmus für die Korrektur vorgesehen. Daher definiert die PLS automatisch einen neuen Startpunkt und eine verkürzte Demontagesequenz für diesen Arbeitsschritt, da das Werkzeug sich immer noch vor dem Werkstück befindet. Die Steuerungsinformationen werden mit Hilfe der Zellenhauptsteuerung an den Roboter übergeben und der Demontageschritt kann vollendet werden.

7.3 Wirtschaftlichkeitsabschätzung

Für eine ökonomische Betrachtung des Programmier- und Steuerungssystems wurde eine Wirtschaftlichkeitsabschätzung durchgeführt. Diese Wirtschaftlichkeitsabschätzung dient der Orientierung und stellt keine detaillierte Kosten-Nutzen-Rechnung dar, wobei hier die Kategorien der Programmierverfahren in Bezug auf ihre Effizienz miteinander verglichen werden. Hierbei werden die Kategorien Online- und Offlineprogrammierverfahren mit dem technologieorientierten Programmier- und Steuerungssystem verglichen. Bei der Kategorie Onlineprogrammierverfahren ist das Teach-in-Verfahren das Referenzverfahren. Bei der Kategorie

Offlineprogrammierverfahren wurde die Programmierung per grafischer Simulation, wie das Produkt „eM-Engineer“ von der Firma Tecnomatix, als Referenzverfahren gewählt. Die Zeiten, die im Zusammenhang mit dieser Arbeit benutzt werden, wurden geschätzt und basieren auf Erfahrungen, die im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) geförderten Sonderforschungsbereiches 281 „Demontagefabriken zur Rückgewinnung von Ressourcen in Produkt- und Materialkreisläufen“ gesammelt wurden.

Unternehmen benutzen die drei Produktionsfaktoren Material, Mitarbeiter und Betriebsmittel, um ihre Produkte herzustellen. Welche Kombination von Produktionsfaktoren optimal ist, ist je nach Land, Branche und Wirtschaftslage verschieden. Der Aufwand für jeden Produktionsfaktor und der gesamte Aufwand lassen sich nicht direkt vergleichen, sondern nur indirekt, indem der Aufwand monetär ausgedrückt wird. Die Kostenrechnung beginnt immer mit der Abbildung von Material, Mitarbeiter und Betriebsmittel auf der gemeinsamen Ebene der Kosten. Damit können unterschiedliche Fertigungsverfahren gegenübergestellt werden [Con-02].

Die Maschinenstundensatzrechnung ist ein etabliertes, praxisnahes Kalkulationsverfahren, das in vielen Betrieben angewendet wird. Die Planung von Kapazitäten und Kosten sowie eine genaue Dokumentation der Fertigungsabläufe sind die Voraussetzung für eine sinnvolle Anwendung der Maschinenstundensatzrechnung. Die Anwendung bietet folgende Vorteile [Pfl-93]:

- Die Abrechnung der Leistungen erfolgt entsprechend der Kostenverursachung.
- Sie ist für den Prozess der Preisbildung und für die Preisentscheidung sehr gut geeignet.
- Die Kapazitäten und Kosten werden in die Planung einbezogen.
- Sie ermöglicht die sinnvolle Anwendung einer maschinenbauspezifischen Deckungsbeitragsrechnung.
- Die Wirtschaftlichkeitsberechnungen verschiedenster Art können auf Basis der Maschinenstundensatzrechnung durchgeführt werden.

Andere Kalkulationsverfahren können mit geringem Aufwand in die Maschinenstundensatzrechnung überführt werden. Als Nachteil der Maschinenkostenrechnung ist der hohe Anfangsaufwand bei der Datenermittlung und bei der späteren ständig

notwendigen Datenaktualisierung zu sehen. Außerdem arbeiten viele Abteilungen bei der Festlegung des Kalkulationsablaufes zusammen, da die Kostenkomponenten, die in der Kalkulation berücksichtigt werden, vor allem in größeren Betrieben von unterschiedlichen Bereichen verwaltet werden. In kleineren Betrieben bietet dieser Aspekt eher Vorteile, da die abteilungsübergreifende Zusammenarbeit zu einer höheren Akzeptanz und Transparenz führen kann [Bar-96].

Ausgangsbasis für die Maschinenstundensatzrechnung ist die Annahme, dass die Gemeinkosten einer Kostenstelle am stärksten durch die kapazitäts- bzw. maschinenabhängigen Kosten beeinflusst werden. Unterstützt wird diese Annahme durch die Tatsache, dass die Investitionskosten neuer Maschinen und Anlagen gegenüber den Lohnkosten einen immer größeren Teil einnehmen. Diese Kostenarten werden innerhalb der Kostenstelle gesondert ermittelt und verrechnet. Die Maschinenstundensatzrechnung unterteilt die Kosten nach solchen, die durch das Betriebsmittel verursacht werden und denen, die dem Bedienungspersonal zugerechnet werden müssen. Diese Kosten werden auch als kapazitätsabhängige Kosten bezeichnet. Die Fertigungskosten setzen sich aus drei Teilsummen zusammen [Bar-96]:

- Nettomaschinenstundensatz: Summe der maschinen- bzw. platzabhängigen Kostenarten,
- Bruttomaschinenstundensatz: Nettomaschinenstundensatz zuzüglich der personalabhängigen Kosten,
- Fertigungskosten: Bruttomaschinenstundensatz zuzüglich der Restfertigungsgemeinkosten.

Der Nettomaschinenstundensatz wird nach der **Gleichung 6.1** ermittelt; hierbei sind alle Kosten einzubeziehen, die- bezogen auf eine Leistungseinheit- bedeutsam, planbar und überwachbar sind [Bar-96].

$$\text{Nettomaschinenstundensatz } \mathrm{NK_{Mh}} = \frac{\sum \text{Kostenarten } [\text{€}/\text{h}]}{\text{jährliche Lastlaufzeiten [h]}}$$

(Gleichung 7.1)

Unter Berücksichtigung der einzelnen Kostenarten und Stundensätze ergibt sich: [in Anlehnung Bar-96]

$$\mathrm{NK_{Mh}} = \frac{\mathrm{K_A} + \mathrm{K_Z} + \mathrm{K_I} + \mathrm{K_E} + \mathrm{K_R} + \mathrm{K_W} + \mathrm{K_{ST}}}{\mathrm{T_{LA}}}$$

(Gleichung 7.2)

mit folgenden Variablen:

- NK_{Mh}: Nettomaschinenstundensatz [€],
- K_A: Abschreibungskosten [€/h],
- K_Z: Zinskosten [€/h],
- K_I: Instandhaltungskosten [€/h],
- K_E: Energiekosten [€/h],
- K_R: Raumkosten [€/h],
- K_W: Werkzeugkosten [€/h],
- K_{ST}: Kosten für Stillstands-und Testzeiten [€/h] sowie
- T_{LA}: jährliche Laufzeit der Einrichtung [h].

In diesem Kapitel wird der wirtschaftliche Vorteil des technologieorientierten Programmier- und Steuerungssystems gegenüber anderen Programmierverfahren aufgezeigt, wobei folgende Annahmen getroffen werden:

- Es wird mit Hilfe der verschiedenen Programmierverfahren der gleiche Industrieroboter programmiert, damit heben sich die Abschreibungs-, Zins-, Instandhaltungs-, Energie-, Raum- und die Werkzeugkosten bei der Berechnung des Nettomaschinenstundensatzes auf.
- Für den wirtschaftlichen Vergleich wird der Bruttomaschinenstundensatz verwendet, wobei zusätzlich nur die Fertigungslöhne betrachtet werden, alle anderen personalabhängigen Kostenkomponenten werden vernachlässigt.
- Für die Wirtschaftlichkeitsabschätzung wird vereinfacht die Programmierung eines einzelnen Industrieroboters betrachtet.

Aufgrund der Annahmen wird der Bruttomaschinenstundensatz wie folgt berechnet

$$BK_{Mh} = (Z_{ST} \cdot E_D) + \left(\sum_{i=1}^{n} Zi_{Pr} \cdot L_{Pr}\right) \qquad \text{(Gleichung 7.3)}$$

mit folgenden Variablen:

- BK_{Mh}: Bruttomaschinenstundensatz [€],
- Z_{ST}: Stillstands-und Testzeit [h],
- E_D: Erlöse des Demontagesystems pro Stunde [€/h],
- Zi_{Pr}: Programmierzeiten [h] und
- L_{Pr}: Lohnkosten [€/h].

Es gibt verschiedene Programmierebenen, in denen verschiedene Personengruppen (z. B. ein RC-Experte oder Werker) Programmieraufgaben übernehmen. Hierbei variieren die personalabhängigen Kosten und hängen von der jeweiligen Gehalts- bzw. Tarifstufe ab, siehe **Tabelle 7.3**.

Tabelle 7.3: Aufteilung der Stundenlöhne
(fiktive Annahmen als Berechnungsgrundlagen)

Personal	Aufgabenbereich	Stunden-lohn
Bediener (Werker)	Bedienung des Industrieroboters	30 €/h
RC-Experte	Programmierung eines Industrie-roboters	60 €/h
Anwendungsprogrammierer	Programmierung von Datenbanken und des Programmierleitstands	50 €/h
Automatisierungsingenieur	Programmierung der Speicher-programmierbaren Steuerung	50 €/h

Die Wirtschaftlichkeitsabschätzung der Programmierverfahren erfolgt anhand von Szenarien, die Demontageprozesse von Konsumgütern in kleinen und mittleren Unternehmen repräsentieren. Hierbei werden folgende Szenarien betrachtet:

1. Erstprogrammierung der Demontageprozesse eines Produkts,
2. Produktwechsel innerhalb einer Fertigungsfamilie,
3. Produktwechsel außerhalb einer Fertigungsfamilie und
4. Betrachtung des Lebenszyklus eines Industrieroboters.

Erstprogrammierung der Demontageprozesse eines Produkts

In diesem Szenario wird der Programmieraufwand für die Erstprogrammierung von Industrierobotern für ein Produkt betrachtet, wobei bisher keine Anwender-programme erzeugt wurden. Es wird die Programmierung der Demontageprozesse einer Waschmaschine betrachtet. Eine Firma TOPS-Recycling bietet Demontage-dienstleistungen von Produktgütern an und schließt mit der Firma Siemens einen Vertrag über die zukünftige Demontage von Waschmaschinen ab. In diesem Szenario werden die Neuprogrammierkosten für die Demontageprozesse von Waschmaschinen der Firma Siemens (Produkt A) ermittelt. Ferner gilt: Der Erlös für

den Verkauf der Demontageerzeugnisse einer Waschmaschine beträgt 100 € und pro Stunde werden fünf Waschmaschinen demontiert.

Bei der Online-Programmierung muss der RC-Experte am Industrieroboter die notwendige Programmierung durchführen. Dadurch ist während der Programmierung das System nicht verfügbar und damit entspricht die Stillstandzeit der Programmierzeit. Dahingegen wird bei der Offline-Programmierung der Roboter vom Demontagesystem unabhängig programmiert; aufgrund der Unterschiede zwischen dem Modell und der Realität muss die Programmierung anschließend im Demontagesystem getestet werden. Das bedeutet, dass während der Testphase das Demontagesystem nicht genutzt werden kann, somit entspricht die Stillstandzeit der Testzeit.

Die Programmierung mit Hilfe des technologieorientierten Programmier- und Steuerungssystems erfolgt durch mehrere Personenklassen mit und ohne Demontagesystem. Die Entwicklung des Datenbanksystems und des Programmierleitstandes erfolgt offline, wohingegen die Elementaranweisungen einmalig online mit Hilfe der Robotersteuerung programmiert werden müssen. Daher entstehen Stillstandszeiten des Demontagesystems während der Roboterprogrammierung und der Testphase. Die Berechnung der Gesamtkosten für die Erstprogrammierung ist in der **Tabelle 7.4** dargestellt.

Tabelle 7.4: Kostenaufstellung des ersten Szenarios (Teil 1)
(Zeitangaben basieren auf Erfahrungswerten)

was	**wer**	**Dauer** [h]	**Kosten** [€]	
			Einzel-kosten	Gesamt-kosten
Onlineprogrammierung				
Roboterprogrammierung	RC-Experte	250	15.000	
Stillstandszeit	-	250	125.000	
				140.000
Offlineprogrammierung				
Roboterprogrammierung	RC-Experte	250	15.000	
Test	RC-Experte	30	1.800	
Stillstandszeit	-	30	15.000	
				31.800

Tabelle 7.5: Kostenaufstellung des ersten Szenarios (Teil 2)
(Zeitangaben basieren auf Erfahrungswerten)

was	**wer**	**Dauer** [h]	**Kosten** [€]	
			Einzel-kosten	Gesamt-kosten
Programmierung mit dem TOPS				
Roboterprogrammierung	RC-Experte	10	600	
SPS-Anpassung	Automatisierungsingenieur	10	500	
DBS-Erzeugung	Anwendungsprogrammierer	100	5.000	
PLS-Erzeugung	Anwendungsprogrammierer	250	12.500	
Test	Werker	10	3.000	
Stillstandszeit	-	20	10.000	
				28.900

Produktwechsel innerhalb einer Fertigungsfamilie

Die Firma TOPS-Recycling schließt weiterhin mit der Firma Miele einen Vertrag über die Demontage von Waschmaschinen (Produkt B) ab, so dass zukünftig Waschmaschinen der Firma Siemens und der Firma Miele demontiert werden. Es werden die Kosten für die Umprogrammierung der Demontageprozesse der Waschmaschinen der Firma Siemens auf die Demontageprozesse der Waschmaschine der Firma Miele bewertet. Da beide Produkte zu der Produktfamilie Waschmaschinen gehören, unterscheiden sich die Demontageprozesse nur unwesentlich, siehe **Kapitel 7.2.1**. Darüber hinaus gilt: Die Anzahl der demontierten Waschmaschinen pro Stunde sowie der Erlöse sind bei beiden Produkten identisch. Die Kosten für die Umprogrammierung bei einem Produktwechsel innerhalb der Fertigungsfamilie sind in der **Tabelle 7.5** dargestellt.

Tabelle 7.6: Kostenaufstellung des zweiten Szenarios (Teil 1)
(Zeitangaben basieren auf Erfahrungswerten)

was	**wer**	**Dauer** [h]	**Kosten** [€]	
			Einzel-kosten	Gesamt-kosten
Onlineprogrammierung				
Roboterprogrammierung	RC-Experte	30	1.800	
Stillstandszeit	-	30	15.000	
				16.800

Tabelle 7.7: Kostenaufstellung des zweiten Szenarios (Teil 2)
(Zeitangaben basieren auf Erfahrungswerten)

was	**wer**	**Dauer** [h]	**Kosten** [€]	
			Einzel-kosten	Gesamt-kosten
Offlineprogrammierung				
Roboterprogrammierung	RC-Experte	20	1.200	
Test	RC-Experte	10	600	
Stillstandszeit	-	10	5.000	
				6.800
Programmierung mit dem TOPS				
DBS-Erzeugung	Anwendungsprogrammierer	20	1.000	
				1.000

Produktwechsel außerhalb einer Fertigungsfamilie

Das Szenario beschreibt den Aufwand bei der Umprogrammierung wegen eines Produktwechsels außerhalb einer Fertigungsfamilie. Die Firma TOPS-Recycling hat neben den beiden Waschmaschinenherstellern mit einem Automobilhersteller einen Vertrag abgeschlossen über die Demontage eines PKW (Produkt C). Die Kosten für die Umprogrammierung der Demontageprozesse der Produktfamilie Waschmaschine auf die Demontageprozesse des PKW werden analysiert. Des Weiteren gilt: Der Erlös für den Verkauf der Demontageerzeugnisse eines PKW beträgt 1.000 € und pro Stunde wird ein Fahrzeug demontiert. Die Kosten für die Umprogrammierung bei einem Produktwechsel außerhalb einer Produktfamilie sind in der **Tabelle 7.6** dargestellt.

Tabelle 7.8: Kostenaufstellung des dritten Szenarios (Teil 1)
(Zeitangaben basieren auf Erfahrungswerten)

was	**wer**	**Dauer** [h]	**Kosten** [€]	
			Einzel-kosten	Gesamt-kosten
Onlineprogrammierung				
Roboterprogrammierung	RC-Experte	750	45.000	
Stillstandszeit	-	750	750.000	
				795.000

Tabelle 7.9: Kostenaufstellung des dritten Szenarios (Teil 2)
(Zeitangaben basieren auf Erfahrungswerten)

was	**wer**	**Dauer** [h]	**Kosten** [€]	
			Einzel-kosten	Gesamt-kosten
Offlineprogrammierung				
Roboterprogrammierung	RC-Experte	750	45.000	
Test	RC-Experte	60	3.600	
Stillstandszeit	-	60	60.000	
				108.600
Programmierung mit dem TOPS				
DBS-Erzeugung	Anwendungsprogrammierer	200	10.000	
PLS-Erzeugung	Anwendungsprogrammierer	500	25.000	
Test	Werker	20	600	
Stillstandszeit	-	20	20.000	
				55.600

Betrachtung des Lebenszyklus eines Industrieroboters

In diesem Szenario wird die Programmierung von Industrierobotern über den Lebenszyklus analysiert. Hierbei wird angenommen, dass ein Industrieroboter eine Lebensnutzungsdauer von zehn Jahren aufweist. Weiterhin wird vereinbart, dass in diesem Zeitraum keine weiteren Demontageprodukte hinzukommen. Die Umrüstung der Demontage der Produkte A, B und C erfolgt alle zehn Wochen und demontiert wird an 50 Wochen im Jahr. Es wird angenommen, dass zu Beginn der Demontage die Anlage auf Produkt A eingestellt ist. Nach Ablauf des ersten Jahres befindet sich die Demontageanlage auf Produkt C und nicht auf dem Ausgangsprodukt A. Damit ergibt sich, dass sich in den ersten drei Jahren der Wechsel von A → B sowie C → A zehnmal ereignet und der Wechsel von B → C fünfmal. Da sich dieser Ablaufzyklus alle drei Jahre wiederholt, werden diese Werte mit drei multipliziert und die Anzahl der Anpassungen nach neun Jahren ermittelt. Der Ablauf des zehnten Jahres ist dem ersten Jahr gleich. Daraus folgt die Aufteilung der Anpassungen, siehe **Tabelle 7.7**. Da sich die Anpassungskosten aufgrund der Erlöse des Demontagesystems pro Stunde unterscheiden, ergeben sich zwei Kostenszenarien: Anpassung an eine Waschmaschinendemontage (A → B und C → A) und Anpassung an die Pkw-Demontage (B → C).

Tabelle 7.10: Aufteilung der Anpassungen

nach 3 Jahren	
Art der Anpassung	Anzahl der Anpassungen
A → B und C → A	10
B → C	5
nach 9 Jahren	
Art der Anpassung	Anzahl der Anpassungen
A → B und C → A	30
B → C	15
nach 10 Jahren	
Art der Anpassung	Anzahl der Anpassungen
A → B und C → A	**33**
B → C	**17**

Nachdem alle Kostenbestandteile und die Anzahl der jeweiligen Anpassungen ermittelt wurden, erfolgt hier die Aufstellung der Gesamtkosten über den Lebenszyklus, vgl. **Tabelle 7.8**. Indem die Gesamtkosten der Offline- und Onlineprogrammierung durch die Gesamtkosten des technologieorientierten Programmier- und Steuerungssystems dividiert werden, wird das Verhältnis ermittelt. Daraus folgt, dass die Onlineprogrammierung 11,5-Mal und die Offlineprogrammierung 1,7-Mal teurer als die Programmierung mit Hilfe des technologieorientierten Programmier- und Steuerungssystems sind.

Tabelle 7.11: Kostenaufstellung des vierten Szenarios (Teil 1)

was	**Anzahl der Anpassungen**	**Kosten** [€]	
		Einzel-kosten	Gesamt-kosten
Onlineprogrammierung			
Neuprogrammierung Produkt A	1	140.000	
Produktwechselprogrammierung Produkt B	1	16.800	
Neuprogrammierung Produkt C	1	795.000	
Anpassung Waschmaschinendemontage	33	1.120	
Anpassung Pkw-Demontage	17	2.120	
			1.024.800

Tabelle 7.12: Kostenaufstellung des vierten Szenarios (Teil 2)

was	Anzahl der Anpassungen	Kosten [€]	
		Einzel-kosten	Gesamt-kosten
Offlineprogrammierung			
Neuprogrammierung Produkt A	1	31.800	
Produktwechselprogrammierung Produkt B	1	6.800	
Neuprogrammierung Produkt C	1	108.600	
Anpassung Waschmaschinendemontage	33	1.120	
Anpassung Pkw-Demontage	17	2.120	
			220.200
Programmierung mit dem TOPS			
Neuprogrammierung Produkt A	1	31.800	
Produktwechselprogrammierung Produkt B	1	6.800	
Neuprogrammierung Produkt C	1	108.600	
Anpassung Waschmaschinendemontage	33	50	
Anpassung Pkw-Demontage	17	100	
			88.850

Da die Wirtschaftlichkeitsabschätzung auf der Annahme beruht, dass mit Hilfe der verschiedenen Programmierverfahren identische Produktionssysteme programmiert werden, sind die Gesamtkosten im Wesentlichen von den Lohnkosten und den Stillstandszeiten des Demontagesystems abhängig. Die Lohnkosten für diese Kostenrechnung sind fiktive Annahmen, wobei die Lohndifferenzen zwischen den unterschiedlichen Personengruppen auch bei Erhöhung oder Verringerung und somit die Differenzen der Gesamtkosten erhalten bleiben. Weiterhin werden durch die Programmierung der Industrieroboter mit dem technologieorientierten Programmier- und Steuerungssystem die Stillstandszeit verkürzt, da eine Anpassung des Produktionssystems an verschiedene Produkte vereinfacht wurde.

7.4 Konzeptergänzende Lösungen

7.4.1 Hilfesystem

Intelligente Hilfesysteme sollen den Anwender bei der Erreichung seines Arbeitsziels so unterstützen, dass er die Software möglichst effizient und ohne Fehlbedienungen einsetzt. Darüber hinaus hängt die Akzeptanz komplexer Softwaresysteme beim Anwender immer stärker von der Qualität der Hilfen ab, die er bei der Einarbeitung in ein neues System und später bei Bedienproblemen angeboten bekommt [Wah-94].

Weiterhin wird die Funktionalität von einer Anwender-Software immer umfangreicher. Keine Software, sei sie noch so anwenderfreundlich entwickelt, wird ganz den Anforderungen der selbsterklärenden Oberfläche gerecht [Kno-09].

Das Ziel eines benutzerorientierten Hilfesystems ist es, den Anwender bei der Arbeit mit dem technologieorientierten Programmier- und Steuerungssystem zu unterstützen. Hierbei müssen folgende Anforderungen bei der Realisierung berücksichtigt werden:

- Inhalt soll vom Anwender leichter nachvollzogen werden können (prägnante, leicht verständliche, kurze Sätze),
- aktive Formulierungen sollen verwendet werden,
- unklarer Fachjargon soll vermieden werden,1
- konsistente Terminologie muss verwendet werden, d. h., Fachausdrücke stehen nur für ein und dieselbe Sache und müssen eindeutig erklärt werden,
- „Schritt-für-Schritt-Erklärungen“ sind zu verwenden,
- unterschiedliche Bedürfnisse der unterschiedlichen Anwendergruppen sind zu berücksichtigen (Problemorientierung) und
- Benutzerinformationen in kleine, geschlossene Informationseinheiten sind zu zerlegen [Kno-09].

Ein planbasiertes Hilfesystem ist eine denkbare Lösung, da der Benutzer des Programmier- und Steuerungssystems zur Erreichung seines Arbeitsziels einen bestimmten Plan verfolgt, der sich in einer Folge von Bedienaktionen konkretisiert. Ein solcher Plan kann fehlerhaft, suboptimal oder unvollständig sein, da der Bediener die Funktionalität des Systems nur unvollständig beherrscht. Ein planbasiertes Hilfesystem muss den intendierten Plan des Benutzers möglichst frühzeitig erkennen, um rechtzeitige und effiziente Hilfestellungen anzubieten. Andererseits muss das Hilfesystem auch in der Lage sein, selbst Pläne generieren zu können, die das angenommene Benutzerziel effizient erreichen, beispielsweise wenn der Benutzer selbst nicht weiterkommt oder einen offensichtlich fehlerhaften oder im gegebenen Kontext suboptimalen Plan verfolgt [Wah-94].

7.4.2 Unternehmensnetzwerk

Aufgrund der beschränkten Ressourcenverfügbarkeit von kleinen und mittleren Unternehmen sollte es ein Ziel sein, ein Unternehmensnetzwerk zu entwickeln und

zu etablieren, das die Nutzung von technologieorientierten Programmier- und Steuerungssystemen unterstützt. Unternehmensnetzwerke stellen einen Weg zur Nutzung der Chancen und Minimierung der Risiken von Innovationsprozessen dar. Eine verteilte Innovation im Rahmen einer Kooperation (z. B. im Rahmen eines Netzwerkes) bietet vor allem KMU die Möglichkeit, eigenständig zu bleiben sowie gleichzeitig den Aktionsradius zu erweitern [Löf-04].

Ein denkbares Szenario für ein Unternehmensnetzwerk ist die Kooperation von:

- Roboterhersteller,
- Unternehmen für Softwareentwicklung,
- Dienstleister für Automatisierungstechnik und
- KMU als Anwender.

Die *Roboterhersteller* könnten Industrieroboter anbieten, bei denen Elementaranweisungen auf der Robotersteuerung bereits vorprogrammiert und die Industrieroboter damit für die Nutzung in einem technologieorientierten Programmier- und Steuerungssystem vorbereitet sind. *Unternehmen für Softwareentwicklungen* könnten Datenbanksysteme und Programmierleitstände für Fertigungsfamilien entwickeln, die sie Endanwendern anbieten. Darüber hinaus würde eine modulare Entwicklung des Datenbanksystems und des Programmierleitstands eine individuelle Anpassung an die spezifischen Bedürfnisse ermöglichen. Unternehmen für Softwareentwicklung könnten Umsätze mit dem Verkauf von Datenbanksystemen, Programmierleitständen und deren Anpassungen sowie einem nachgelagerten Service erzielen. *Dienstleister für Automatisierungstechnik* könnten die Realisierung des technologieorientierten Programmier- und Steuerungssystems bei einem Endanwender durchführen. Das bedeutet, im Datenbanksystem werden die spezifischen Steuerungsinformationen gespeichert und die komplette Integrationsarbeit wird durchgeführt, um die einzelnen Komponenten zu einem Gesamtsystem zu verknüpfen. Darüber hinaus werden die Bediener (Werker) nach einem Test des Gesamtsystems geschult und das System wird an den Endanwender übergeben. Weiterhin könnte durch die Dienstleister ein Online-Hilfesystem angeboten werden, das eine Fernunterstützung für die Bediener der Unternehmen bietet. Durch dieses Unternehmensnetzwerk wird das *KMU als Anwender* dahin gehend entlastet, dass durch das KMU für die Nutzung von Industrierobotern nur das technologische Wissen über die Fertigungsprozesse bereitgestellt werden muss.

7.4.3 Softwareerweiterungsmodule für einen effizienten Einsatz

Bei der bisherigen Entwicklung von Programmierverfahren für Industrieroboter wurden unterschiedliche Komponenten entwickelt, die für eine Programmoptimierung eingesetzt werden und zukünftig in das technologieorientierte Programmier- und Steuerungssystem integriert werden können. Als Beispiele für solche Entwicklungen werden hier die Bahn-, Greif- und Trajektorienplanung vorgestellt.

Bahnplanung

Bei gegebener Roboterumwelt, Anfangskonfiguration und Zielkonfiguration findet eine Sequenz von Roboterbewegungen so statt, dass es zu keiner Kollision mit irgendeinem Objekt dieser Roboterumwelt kommt. Hier wird von „Sequenz von Roboterbewegungen“ oder auch von „Roboterkonfigurationen“ gesprochen, da es für mehrachsige Roboter mehrere Möglichkeiten gibt, die Position und Orientierung eines Endeffektors zu erreichen [Wec-06].

Globale Verfahren berücksichtigen den gesamten Roboterarbeitsraum bei der Suche nach einer kollisionsfreien Bahn. Die Suche stützt sich auf einen sogenannten Hinderniskonfigurationsraum, der für jede Stellung des Roboters Informationen über etwaige Kollisionen enthält. Der Hinderniskonfigurationsraum muss vor der Bahnsuche berechnet werden. Dagegen betrachten lokale Verfahren den Bereich des Arbeitsraums, in dem sich der Endeffektor momentan befindet. Sie verfügen über bestimmte Strategien zur Generierung von Roboterbahnen. Die generierten Bahnen werden während der Planung einer Kollisionsprüfung unterzogen und je nach Ergebnis der Prüfung verworfen oder beibehalten. Weiterhin entfällt bei lokalen Verfahren der Aufwand zur Berechnung des Hinderniskonfigurationsraums, sie benötigen jedoch bei schwierigen Problemstellungen oft sehr lange Zeit zur Bestimmung einer kollisionsfreien Bahn [Kug-99].

Zur Berechnung des Hinderniskonfigurationsraums wird der gesamte Arbeitsraum des Roboters mit möglichst hoher Auflösung diskretisiert. Eine Roboterposition kann bei einem 6-Achs-Roboter entweder durch die Achswinkelwerte der Gelenke oder die kartesischen Koordinaten des Endeffektors einschließlich der Raumwinkel beschrieben werden. Bei der Suche nach einer kollisionsfreien Bewegungsbahn für den Roboter werden benachbarte kollisionsfreie Punkte im Hinderniskonfigurationsraum zu einer Bahn verbunden. Die Berechnung des Hinderniskonfigurationsraums ist

aufgrund der feinen Diskretisierung sehr zeitaufwendig, insbesondere wenn mehr als drei Roboterachsen betrachtet werden. Da die Betrachtungen von mehr als drei Achsen bei vielen Anwendungen gefordert sind und der Hinderniskonfigurationsraum zudem bei jeder Änderung der Hinderniskonstellation angepasst und neu berechnet werden muss, wurden mehrere Bahnplanungsverfahren entwickelt, die umgekehrt vorgehen. Das heißt, es werden zuerst Bahnen vorgeschlagen und es wird dann überprüft, ob diese Bahnen kollisionsfrei sind, bis eine kollisionsfreie Gesamtbahn gefunden wird. Diese Verfahren werden als lokale Verfahren bezeichnet, da sie eine lokale Strategie zur Hindernisumgehung verfolgen und nicht global im gesamten Arbeitsraum nach einer Bahn suchen [Kug-99].

Greifplanung

Aufgabe der Greifplanung ist es, eine günstige Greiferposition für die Handhabung eines Objektes zu finden, so dass der sichere Transport, die richtige Lage und keine Beschädigung der Werkstücke während des Greifens möglich werden [Wec-04]. Mögliche Greifkonfigurationen müssen geometrische und physikalische Randbedingungen einhalten. Die meisten Verfahren zur Greifplanung suchen zunächst nach geometrischen Elementen an dem zu greifenden Objekt, die für das Greifprinzip des verwendeten Greifers geeignet sind. Für die ermittelten geeigneten Geometrieelemente werden dann bestimmte Griffe oder Greifkonfigurationen und deren Greifsicherheit bestimmt. Die Greifsicherheit wird beeinflusst durch das Gewicht des zu greifenden Objekts, die maximal vom Greifer aufgebrachte Greifkraft, die wirkenden Reibungskräfte, die Angriffspunkte des Greifers am Werkstück sowie die Erreichbarkeit der Position durch den Roboter. Die jeweilige Greifsicherheit der einzelnen Greifkonfigurationen dient als Bewertungskriterium zur Auswahl der besten Greifkonfiguration, diese erfolgt in der Regel automatisch. Im nächsten Schritt wird die sich aus der Greifplanungskonfiguration und der Ziellage des Objekts ergebende Ablegekonfiguration betrachtet. Kann die Ablegekonfiguration vom Roboter angefahren werden, so erfolgt die Planung der Feinbewegung für das Greifen und Ablegen [Kug-99].

Trajektorienplanung

Die Berechnung eines geeigneten Geschwindigkeitsprofils entlang einer Bahn wird als Trajektorienplanung bezeichnet. Das Ziel der Trajektorienplanung für Roboter ist

die Berechnung einer Steuerfolge, um das Handhabungssystem von der Startposition in die Zielposition zu bringen. Dabei müssen je nach Zielsetzung und Anwendungsfall die unterschiedlichen Rand- und Nebenbedingungen berücksichtigt werden. Diese Bedingungen sind das statische und dynamische Systemmodell, Anfangs- und Endzustand sowie eine festgelegte Folge von Zwischenzuständen und konstruktive, geometrische, kinematische und dynamische Beschränkungen für die Antriebe und die Bewegung. Darüber hinaus sind noch Aspekte wie Materialschonung, Verschleiß, Lärmentwicklung und Lebensdauer bedeutsam. Für die zielgerichtete Berechnung einer Trajektorie ist immer die Formulierung eines Gütekriteriums notwendig. Dadurch lassen sich weitere Trajektorien definieren:

- Minimierung von Zeit,
- Minimierung des Energieverbrauchs,
- Minimierung von Fehlern und
- Maximierung der Lebensdauer [Tsc-91].

8 Zusammenfassung und Ausblick

Industrieroboter sind eine Standard-Automatisierungskomponente in der Automobilbranche, wo sie in Großserienproduktion eingesetzt werden. Dagegen werden Industrieroboter in anderen Branchen mit anderen Anwendungen sowie in kleinen und mittleren Unternehmen (KMU) im Vergleich zu der Automobilbranche deutlich seltener eingesetzt. Für die spezifischen Produktionsbedingungen von KMU wurden bisher keine geeigneten technischen und wirtschaftlichen Lösungen entwickelt, die hinreichend die geforderten Anforderungen von KMU erfüllen. Die spezifischen Produktionsherausforderungen von KMU, wie beispielsweise ein variantenreiches Produktspektrum bei kleinen Stückzahlen oder die Bedienbarkeit der Roboter durch Werker mit eingeschränkten Kenntnissen auf dem Gebiet der Robotik, bedingen vielfältige Anforderungen an die zukünftigen Programmier- und Steuerungssysteme von Industrierobotern. Zukünftige Programmier- und Steuerungssysteme müssen einen effizienten Einsatz von Industrierobotern bei kleinen Losgrößen ermöglichen, wobei zusätzlich eine einfache Anpassung der Steuerungsparameter gewährleistet werden muss. Darüber hinaus müssen Roboterprogramme flexibel an unterschiedliche Aufgabenstellungen angepasst werden können. Das in dieser Arbeit vorgestellte technologieorientierte Programmier- und Steuerungssystem (TOPS) leistet einen Beitrag für die effiziente Nutzung von Industrierobotern bei KMU.

Zur Erfüllung der genannten Anforderungen wurden die Herausforderungen für kleine und mittlere Unternehmen sowie die Defizite aktuell verfügbarer Programmierverfahren hinsichtlich des Einsatzes bei KMU analysiert. Darauf aufbauend wurden Anforderungen an technologieorientierte Programmier- und Steuerungssysteme erarbeitet und systematisiert.

Das entwickelte Programmier- und Steuerungssystem nutzt den Ansatz, bestehende Anwenderprogramme in kleinste generische Einheiten, die Elementaranweisungen (ELA), zu unterteilen. Die Kombination von offline programmierten Elementaranweisungen und die Online-Verknüpfung mit den entsprechenden Steuerungsinformationen ergibt ein lauffähiges Anwenderprogramm, wobei dieses nur ein virtuelles Programm ist. Das Gesamtkonzept des TOPS besteht aus den vier Komponenten Robotersteuerung, Prozesssteuerung, Datenbanksystem und Programmierleitstand.

Das Konzept wurde prototypisch am Beispiel der Demontage realisiert, da die Demontage von Gebrauchtgütern durch eine enorme Vielfalt an Produktvarianten sowie -zuständen gekennzeichnet ist und somit höchste Flexibilitätsanforderungen an automatisierte Demontagesysteme gestellt werden. Hierzu wurde das technologieorientierte Programmier- und Steuerungssystem in das Pilot-Demontagesystem der TU Berlin implementiert. Die Speicherprogrammierbare Steuerung und die Robotersteuerung wurden adaptiert sowie die Konzepte für das Datenbanksystem und den Programmierleitstand prototypisch realisiert.

Für eine Erprobung des Systems wurden die bisher realisierten Demontageschritte des Pilot-Demontagesystems untersucht. Hierbei wurden für eine Bewertung demontagetypische Situationen vor und nach der Implementierung des technologieorientierten Programmier- und Steuerungssystems miteinander verglichen, wobei die Leistungsfähigkeit, die Bedienerfreundlichkeit und die Wirtschaftlichkeit des Programmier- und Steuerungssystems nachgewiesen wurden.

Im Rahmen zukünftiger Forschungsarbeiten kann das entwickelte Konzept des technologieorientierten Programmier- und Steuerungssystems dahingehend ergänzt werden, dass der Programmierleitstand um kognitive Fähigkeiten erweitert wird, um unbekannte Situationen begreifen und darauf reagieren zu können. Darüber hinaus könnte die Entwicklung eines generischen Programmierleitstands ein universelles Programmier- und Steuerungssystem schaffen, das für unterschiedliche Fertigungsfamilien geeignet ist und mit Hilfe des Datenbanksystems angepasst wird. Die diskutierten konzeptergänzenden Lösungen können umgesetzt werden, um das Programmier- und Steuerungssystem anwenderfreundlicher zu gestalten.

Durch den Einsatz von kollaborierenden Robotern (Kobot) in automatisierten Produktionsanlagen können Prozesse effizienter und kostengünstiger realisiert werden. Hierbei werden die kognitiven Fähigkeiten des Menschen mit der Präzision und Leistungsfähigkeit eines Industrieroboters verknüpft sowie neue Sicherheitseinrichtungen für die Zusammenarbeit von Mensch und Industrieroboter entwickelt. Bei dem Einsatz von Kobot und dem TOPS wäre folgendes Szenario denkbar: Der Werker kann die Industrieroboter nicht nur von einem Programmierarbeitsplatz außerhalb des Produktionssystems programmieren, sondern er besitzt einen mobilen Computer (z. B. ein Personal Digital Assistant) für die Programmierung und Steuerung in direkter Nähe des Roboters. Dadurch wird der Werker in das

Produktionssystem integriert und erhält die Möglichkeit, direkt den Produktionsablauf zu beobachten und auf Veränderungen zu reagieren. Der mobile Computer wird genutzt, um nur die wichtigsten Steuerungsparameter (z. B. Auswahl der Werkzeuge oder deren Parameter) während der Produktion anzupassen. Das bedeutet, dass ein speziell angepasster Programmierleitstand mit reduziertem Funktionsumfang auf dem mobilen Computer installiert wird.

9 Literaturverzeichnis

[Alm-06]	Almeida, C.; Hork, M.:	*Handhabungstechnik und Robotik*. In: Eversheim, W.; Pfeifer, T.; Weck, M. (Hrsg.): *100 Jahre Produktionstechnik - Werkzeugmaschinenlabor WZL der RWTH Aachen von 1906 bis 2006*. Berlin, Heidelberg, New York: Springer-Verlag, 2006. S. 515-528.
[Apa-09]	N. N.:	*XAMMP*. URL: **http://www.apachefriends.org/de/xampp.html** - Aktualisierungsdatum: 09.04.2009.
[Arm-06a]	Armbruster, H.; Kirner, E.; Kinkel, S.:	*Neue Nutzungspotenziale für Industrieroboter*. In: wt – Werkstattstechnik Online Nr. 96 (2006) 9, S. 631-636.
[Arm-06b]	Armbruster, H.; Kirner, E.; Kinkel, S.:	*Neue Kundengruppen für Industrieroboter: Wo liegen unausgeschöpfte Anwendungspotenziale für Roboter im deutschen verarbeitenden Gewerbe?.* URL: **http://www.isi.fraunhofer.de/i/mitteilung_pi.htm#ir5**. - Aktualisierungsdatum: 06.03.2008.
[Arz-02]	Arzberger, P.; Beilschmidt, L.; Ellerckmann, H.:	*Tabellenbuch Informations- und Telekommunikationstechnik*. 5. Auflage. Troisdorf: Gehlen Verlag, 2002.
[Bar-96]	Barmbold, T.:	*Integration von CNC-Maschinen in die Serienfertigung und ihre Kostenverteilung*. Leibniz Universität Hannover, Dissertation, 1996.
[Bar-08]	N. N.:	*Robots: A useful introduction to robots and their applications including*. URL: **http://www.bara.org.uk/info_robots.htm**. - Aktualisierungsdatum: 09.09.2008.
[Ble-01]	Bley, H.; Bernardi, M.; Franke, C.; Seel, U.:	*Process-based Assembly Planning Using a Simulation Sys-tem with Cell Calibration*. In: *Proceeding of the International Symposium on Assembly and Task Planning (ISATP)*. Fukuoka (Japan), 2001, S. 116-121.
[Bot-05]	Bottazi, V.; Fonseca, J.:	*Off-Line Robot Programming Framework*. In: *Proceeding of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS 2005)*. Papeete (Tahiti), 2005, S. 71-76.
[Bre-04a]	Brecher C.; Schröter, B.; Almeida, C.; Dai, F.; et al.:	*Intuitiv bedienbare Programmiersysteme zur effizienten Programmierung von Handhabungsaufgaben*. In: *VDI Robotik 2004*. Düsseldorf: VDI Verlag, 2004, S. 303-310.
[Bre-04b]	Brecher, C.; Almeida, C.; Schröter, B.; Matthias, B.:	*Portables Robotersystem. Wirtschaftliche Durchführung wechselnder Handhabungsaufgaben*. In: VDI-Z Integrierte Produktion. Band 146 (2004) 7/8, S. 27-30.
[Buc-05]	Buchmann, A.; Smolarek, R.:	*SQL & MySQL – interaktiv*. 1. Auflage. Heidelberg: Dpunkt.verlag GmbH, 2005.
[Bus-96]	Bussiek, J.:	*Anwendungsorientierte Betriebswirtschaftslehre für Klein- und Mittelunternehmen*. 2. Auflage. München, Wien: Oldenburg Verlag, 1996.

[Con-02]	Conrad, K.-J.:	*Taschenbuch der Werkzeugmaschinen*. 1. Auflage. München, Wien: Carl Hanser Verlag, 2002.
[Cor-07]	Corke, P.:	*MATLAB Toolboxes: robotics and vision for students and teachers*. In: *Robotics & Automation Magazine* Vol. 14 (2007), No. 2, S. 16-17.
[Dam-96]	Dammertz, R.:	*Ein Programmiersystem zur grafisch strukturierten Erstellung von Roboterprogrammen und Programmieroberflächen*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 1996.
[Den-05]	Denkena, B.; Wörn, H.; Apitz, R.; Bischoff, R.; et al.:	*Roboterprogrammierung in der Fertigung. Einfache Roboterprogrammierung für die Produktion von morgen (Ergebnisse des IRoProg-Projekts)*. In: wt-Werkstattstechnik Online Nr. 95 (2005) 9, S. 656-660.
[Dre-97]	Drews, P.:	*Roboter in der Werkstatt. Modellarbeitsplatz zum Schweißen*. Frankfurt am Main: MaschinenbauVerl. GmbH, 1997.
[Ehr-07]	Ehrlenspiel, K.; Kiewert, A.; Lindemann, U.:	*Kostengünstig Entwickeln und Konstruieren: Kostenmanagement bei der integrierten Produktentwicklung*. 6. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2007.
[Emp-03]	N. N.:	*Empfehlung 2003/361/EG der Europäischen Kommission betreffend die Definition der kleinen und mittleren Unternehmen*. URL: **http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/index_de.htm**. - Aktualisierungsdatum: 20.03.2008.
[Ens-07]	Enste, U.; Müller, J.:	*Datenkommunikation in der Prozessindustrie: Darstellung und anwendungsorientierte Analyse*. 1. Auflage. Essen: Oldenburg Industrieverlag, 2007.
[Ent-09]	N. N.:	*Enterprise and Industry: Facts and figures – SMEs in Europe*. URL: **http://ec.europa.eu/enterprise/entrepreneurship/facts_figures.htm**. - Aktualisierungsdatum: 20.03.2009.
[Fre-91]	Freund, E.; Weber, H.:	*Entwicklung eines VAL-II-Compilers für die Zielsprache IRDATA*. In: Robotersysteme - Zeitschrift für Informationstechnologie und Handhabungstechnik Band 7 (1991), S. 139-147.
[Fre-93]	Freund, E.; Uthoff, J.; Hypki, A.; Van der Valk, U.:	*COSIMIR und PCROB: Integration von Zellensimulation und Robotersteuerung auf PCs*. In: *Intelligente Steuerung und Regelung von Robotern*. Düsseldorf: VDI Verlag, 1993, S. 823-833.
[Fre-98]	Freund, E.; Rokossa, D.; Uthoff, J.:	*Prozessorientierte Roboterprogrammierung auf Knopfdruck*. In: VDI-Z Integrierte Produktion 140 (1998) 9, S. 62-65.
[Fre-02]	Freund, E.; Lüdemann-Ravit, B.:	*A System to Automate the Generation of Program Variants for Industrial Robot Applications*. In: *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*. Lausanne (Switzerland), 2002, S. 1856-1861.

[Fri-09]	Friedrich, T.:	*KMU-gerechte Programmierung und Steuerung von Industrierobotern.* In: *Berliner Industrie-Arbeitskreis (BIAK) Fertigungstechnik und Werkzeugmaschinen 2009*. Berlin, 2009, Vortrag vom 25. und 26. März.
[Gea-99a]	Geary, D.:	*Graphic Java 2.0. - Die JFC beherrschen (AWT).* 3. Auflage. München: Markt+Technik Verlag, 1999.
[Gea-99b]	Geary, D.:	*Graphic Java 2.0 Band II - Die JFC beherrschen (Swing).* 3. Auflage. München: Markt+Technik Verlag, 1999.
[Got-01]	Gotttschald, J.:	*Place&Play-Roboter: Ein portables Handhabungssystem für die Werkstatt*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 2001.
[Gra-98]	Gradwohl, H.; Wüstenberg, D.:	*Einflüsse auf die Verwertung von Altgeräten*. In: Abfallwirtschaftsjournal 6 (1994) 4, S. 193-197.
[Hau-06]	Haun, M.:	*Handbuch Robotik - Programmieren und Einsatz intelligenter Roboter*. 1. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2007.
[Här-05]	Härtwig, J.-P.:	*Verfahren und Systeme zur Demontage komplexer Gebrauchsgüter*. In: Uhlmann, E. (Hrsg.): *Berichte aus dem Produktionstechnischen Zentrum Berlin*. Fraunhofer IRB Verlag: Stuttgart, 2005.
[Hec-95]	Heck, H.:	*Ein frei konfigurierbares System zur aufgaben-orientierten Offline-Programmierung von Robotern: Konzeption und Realisierung*. Fernuniversität in Hagen, Dissertation, 1995.
[Hei-00]	Hein. A.:	*Eine interaktive Robotersteuerung für chirurgische Applikationen*. Fortschrittberichte VDI Reihe 17, Nr. 195, VDI Verlag Düsseldorf 2000.
[Hen-96]	Hentschel, C.:	*Beitrag zur Organisation von Demontagesystemen.* Technische Universität Berlin, Dissertation, 1996.
[Hen-02]	Henrich, D.; Kahl, B.:	*Virtual Robot Programming for Deformable Linear Objects: System concept and Prototype Implementation*. In: *12th International Symposium on Measurement and Control in Robotics (ISMCR02)*, Bourges (France), 2002, Compact Disc.
[Hen-04]	Henrich, D.; Kahl, B.:	*Virtuelle Roboter Programmierung: Konzept und Prototypische Implementierung*. In: *VDI Robotik 2004*. Düsseldorf: VDI Verlag, 2004, S. 689-692.
[Her-05]	Herczeg, M.:	*Software-Ergonomie: Grundlagen der Mensch-Computer-Kommunikation.* 2. Auflage. München: Oldenbourg- Wissenschaftsverlag GmbH, 2005.
[Hes-94]	Hesseler, M.:	*Off-line-Programmierung von Industrierobotern*. In: wt - Produktion und Management 84 (1994) 7/8, S. 479-482.
[Hes-95]	Hesseler, M.:	*Beitrag zur Benutzerunterstützten Offline-programmierung von Industrierobotern*. Technische Universität Dortmund, Dissertation, 1995.
[Hes-98]	Hesse, S.:	*Industrieroboterpraxis*. 1. Auflage. Wiesbaden: Vieweg Verlagsgesellschaft, 1998.

[Heß-09]	Heß, P.:	*Roboter übernehmen neue Aufgaben. Programmierung von Industrierobotern wird einfacher.* In: VDI-Z Integrierte Produktion Band 151 (2009) 10, S. 51-53.
[Hir-99]	Hirzinger, H.; Vogel, J.; Brunner, B.; Landzettel, K.:	*Internet Virtual Reality Technologien zur Fernvisualisierung für Teleservice- und Telerobotikanwendungen*. In *VDI-Bericht 1515 Industrielle Automation und Internet/Intranet-Technologie*, Düsseldorf: VDI Verlag, 1999, S.199-210.
[Hir-01]	Hirzinger, G.; Brunner, B.; Vogel, J.:	*Aufgabenorientierte Fernprogrammierung von Robotern*. In: at - Automatisierungstechnik Nr. 49 (2001) 7, S. 312-319.
[Hol-93]	Hollenberg, F.; Vietze, L.:	*Offline-Programmierung von Schweißrobotern: Kenntnisse einer Roboterprogrammiersprache sind für den Anwender nicht mehr nötig*. In: VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (Hrsg.): *Intelligente Steuerung und Regelung von Robotern, VDI-Bericht 1094*, Düsseldorf, VDI-Verlag 1993, S. 87-96.
[Hol-95]	Hollenberg, F.:	*CAD-basierte Offline-Programmierung von Lichtbogenschweißrobotern*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 1995.
[IFR-08]	N. N.:	*IFR International Federation of Robotics - World robotics market 2006*. URL: **http://www.ifr.org/statistics/keyData2006.htm** - Aktualisierungsdatum: 19.03.2008.
[IPA-06]	N. N.:	*Eine neue Generation von Industrierobotern für kleine und mittlere Unternehmen (KMU)*. URL: **www.smerobot.org/08_scientific_papers/papers/Oberer_KMU_IPA_2006.pdf** - Aktualisierungsdatum: 12.03.2009.
[Iwa-05]	Iwanitz, F.; Lange, J.:	*OPC - Grundlagen, Implementierung und Anwendung*. 3. Auflage. Heidelberg: Hüthig-Verlag, 2005.
[Jun-05]	Jungermann, H.; Pfister, H.-R.; Fischer, K.:	*Die Psychologie der Entscheidung - Eine Einführung*. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, 2005.
[Kah-08]	Kahl, B.:	*Abstract, Virtual Robot Programming*. URL: **http://www.ai3.uni-bayreuth.de/index.php?size=280&page=projects%252Findex-buttons.php&page2=projects%252Fvirop%25%0b252Ffiles%252Fda-3-dlo-simulation-in-robcad_03.php** - Aktualisierungsdatum: 23.9.2008.
[Kam-00]	Kampker, M. A.:	*Werkzeuge für die verbesserte, nutzenorientierte Werkstattprogrammierung von Schweißrobotern*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 2000.
[Kei-04]	Keil, T.:	*Informationstechnische Integration hybrider Demontagesysteme*. Technische Universität Berlin, Dissertation, 2004.
[Kle-92]	Kleinwächter, U.:	*Ein Modell für die externe Industrieroboter-Programmierung auf der Basis eines Geometriemodellierers*. Dissertation, Technische Universität Chemnitz, 1992.

[Kem-04]	Kemper , A.; Eickler, A.:	*Datenbanksysteme - Eine Einführung*. 5. Auflage. München: Oldenbourg- Wissenschaftsverlag GmbH, 2004.
[Kno-09]	Knopp, S.:	*Effektive Hilfesysteme konzipieren, schreiben und gestalten*. URL: **http://www.tekom.de/index_neu.jsp?url=/servlet/ControllerGUI?action=voll&id=535** - Aktualisierungsdatum: 12.05.2009.
[Koc-07]	Koch, F.; Pyzalla, G; Lehberger, J.:	*Einführung in die Technologie*. 13. Auflage. Köln: Stam Verlag GmbH, 2007.
[Kre-94]	Kreuzer, E. J.; Lugtenburg, J.-B.; Meißner, H.-G.; Truckenbroft, A.:	*Industrieroboter - Technik, Berechnung und anwendungsorientierte Auslegung*. 1. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 1994.
[Kud-07]	Kudraß, T.:	*Taschenbuch Datenbanken*. 1. Auflage. München, Wien: Carl Hanser Verlag, 2007.
[Kug-99]	Kugelmann, D.:	*Aufgabenorientierte Offline-Programmierung von Industrierobotern*. Technische Universität München, Dissertation, 1999.
[Kuk-08]	N. N.:	*Applikation*. URL: **http://www.kuka.com/germany/de/products/software/**. - Aktualisierungsdatum: 18.09.2008.
[Lau-91]	Lauffs, H.-G.:	*Bediengeräte zur 3D-Bewegungsführung*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation 1991.
[Lie-01]	Liebenow, D.:	*Ein Beitrag zur Makroprogrammierung in der automatisierten schweißtechnischen Fertigung*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 1991.
[Lüd-05]	Lüdemann-Ravit, B.:	*Ein System zur Automatisierung der Planung und Programmierung von industriellen Roboterapplikationen*. Technische Universität Dortmund, Dissertation, 2005.
[Löf-04]	Löffler, S.:	*Ganzheitliche Innovationsprozesse in modularen Unternehmensnetzwerken (GINA)*. In Zelewski, S.; Alparslan, A. (Hrsg.): *Proceeding zum Abschlussworkshop der Verbundprojekt GINA, KoEffizient und KOWIEN*. Braunschweig, 2004, S. 3-10.
[Lüt-04]	Lüth, T.; Rose, A.; Hein, A.:	*An optically based tactile system for interactive gradual surface scanning*. In: *Proceeding of the 18th International Congress and Exhibition*. Chicago (USA), 2004, S. 573-578.
[Mat-04]	Matthias, B.; Dai, F.; Hug, K.; Kock, S.; et al.:	*Ein flexibel einsetzbares Robotersystem für variierende Aufgaben in der Maschinenbeschickung*. In: *VDI Robotik 2004*. Düsseldorf: VDI Verlag, 2004, S. 567-574.
[Mey-06]	Meyer, C.:	*Intuitive Industrieroboterprogrammierung*. In: *Proceeding of Automation in der Holzwirtschaft*. Biel (Switzerland), 2006, Compact Disc.
[Möb-96]	Möbius, F.:	*Visuelle Programmierung von Industrieroboters - Ein Beitrag zur bedienergerechten Gestaltung von Programmiersystemen*. Technische Universität Kaiserslautern, Dissertation, 1996.

[Mün-94]	Münch; S; Kreuziger, J.; Kaiser, M.; Dillmann, R.:	*Robot Programming by Demonstration (RPD) - Using Machine Learning and User Interaction Methods for the Development of Easy and Comfortable Robot Programming Systems*. In: *Proceeding of the 24th International Symposium on Industrial Robots (ISIR '94)*. Tokyo (Japan), 1994, Compact Disc.
[Mün-08]	Münch; S.:	*Programmierung von Robotern durch Benutzervorführungen - Ideen und Konzepte*. URL: **http://wwwipr.ira.uka.de/de/Publikationen/All_publications.htm**. - Aktualisierungsdatum: 22.06.2006.
[Mys-08]	N. N.:	*MYSQL - die wichtigsten Argumente für den Einsatz von MySQL*. URL: **http://www.mysql.de/oem/**. - Aktualisierungsdatum: 24.10.2008.
[Neu-97]	Neugebauer, J.-G.:	*Einsatz neuer Mensch-Maschine-Schnittstellen für Robotersimulation und -programmierung*. Fraunhofer-Instituts für Produktionstechnik und Automatisierung IPA, Dissertation, 1997.
[Pep-97]	Peper, S.:	*Schweißstruktur-orientierte Offline-Programmierung von Lichtbogenschweißrobotern*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 1997.
[Pfl-93]	Pfleger, H.; Reiche, W.:	*Das Rechnen mit den Maschinenstundensätzen*. 7. Auflage. Frankfurt/Main: Maschinenbau-Verlag, 1993.
[Php-08]	N. N.:	*PHP - Einleitung*. URL: **http://de.php.net/manual/de/introduction.php**. - Aktualisierungsdatum: 24.10.2008.
[Pir-06]	Pires, J. N.:	*Robotics for Small and Medium Enterprises: Control and Programming Challenges*. In: Industrial Robot, 33 (2006) Nr. 6, S. 485-487.
[Pri-04]	Pritschow, G.:	*Steuerungen*. In: Beitz, W.; Grote, K.-H. (Hrsg.): *Dubbel Taschenbuch für den Maschinenbau*. 21. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2004.
[Pri-05]	Pritschow, G.:	*Automatisierung in der Produktion: Teil 1, Einführung in die Steuerungstechnik*. 1. Auflage. München, Wien: Carl Hanser Verlag, 2005.
[Pro-07]	N. N.:	*Produktionsforschung: 57 erfolgreiche Projekte für Menschen und Märkte*. Bundesministerium für Bildung und Forschung (BMBF). Bonn, Berlin, 2007.
[PSW-08]	N. N.:	*Arbeitsweise - Was ist Plasmaschneiden?* URL: **http://www.plasmaschneiden.com/technologie.html**. - Aktualisierungsdatum: 24.11.2008.
[Ram-07]	Raml, G.:	*Roboter spricht die Sprache des Anwenders*. In: Westkämper, E. (Hrsg.): *Industrieroboter schnell und einfach programmieren*. Stuttgart, 05.12.2007, Vortag.
[Reb-03]	Rebafka, U.:	*Beitrag zur Entwicklung modularer Demontagewerkzeuge*. Technische Universität Berlin, Dissertation, 2003.
[Ref-93]	N. N.:	*Ausgewählte Methoden der Planung und Steuerung. Fachbuchreihe Betriebsorganisation*. 1. Auflage. München, Wien: Carl Hanser Verlag, 1993.
[Rei-07]	Reichle, R.:	*Roboterprogrammierung durch Anlagenhersteller und Endkunden – Erfahrungen und Wünsche*. In: Westkämper, E. (Hrsg.): *Industrieroboter schneller und einfacher programmieren*. Stuttgart, 2007, Vortrag.

[Rei-09] Reinhart, G.; Krug, S.; Hatwig, J.: *Effiziente Programmierung von Schweißrobotern für die Einzel- und Kleinserienfertigung*. In: VDI-Z Integrierte Produktion 151 (2009) ½, S. 51-53.

[Rok-99] Rokossa, D.: *Prozessorientierte Offlineprogrammierung von Industrieroboter*n. Technische Universität Dortmund, Dissertation, 1999.

[Sch-00] Scholz-Reiter, B; Scharke, H.; Li, W.: *A methodology to derive disassembly information for obsolete appliances*. In: *Proceeding of the International CIRP Design Seminar*, 2000, Haifa (Israel), S. 217-222.

[See-99] Seel, U.: *Robotergestützte Zellenkalibrierung als Basis einer Feature-basierten Montageplanung*. Universität des Saarlandes, Dissertation, 1999.

[Sei-08] Seitz, M.: *Speicherprogrammierbare Steuerungen: System- und Programmentwurf für die Fabrik- und Prozessautomatisierung, vertikale Integration*. 2. Auflage. München, Wien: Carl Hanser Verlag, 2008.

[Sel-00a] Seliger, G.; Uhlmann, E.; Härtwig, J.-P.; Keil, T.: *Teilprojekt A6 - Realisierung eines Pilot-Demontagesystems*. In: *Arbeits- und Ergebnisbericht 1998-2000 des Sonderforschungsbereiches 281 - Demontagefabriken zur Rückgewinnung von Ressourcen in Produkt- und Materialkreisläufen.* Technische Universität Berlin, 2000, S. 201-233.

[Sel-00b] Seliger, G.; Uhlmann, E.; Härtwig, J.-P.; Keil, T.: *A Pilot System For The Disassembly Of Home Appliances Using New Tools And Concepts*. In: *Proceeding of the Third World Congress on Intelligent Manufacturing Processes & Systems*. Cambridge (United Kingdom), 2000, S. 453-456.

[Sel-01] Seliger, G.; Uhlmann, E.; Härtwig, J.-P.; Keil, T.: *Pilot Disassembly System*. In: Production Engineering - Annals of the German Academic Society for Production Engineering, Vol. 8 (2001) 1, 2001, S. 83-88.

[Sel-02] Seliger, G.; Uhlmann, E.; Härtwig, J.-P.; Keil, T.: *Destructive Disassembly of Home Appliances*. In: *Proceeding of the 6th World Congress on Integrated Resources Management R'02*, Genf (Switzerland), 2002, Compact Disc.

[Sel-03] Seliger, G.; Uhlmann, E.; Härtwig, J.-P.; Keil, T.: *Teilprojekt A6 - Realisierung eines Pilot-Demontagesystems. Arbeits- und Ergebnisbericht 2001-2003 des Sonderforschungsbereiches 281 Demontagefabriken zur Rückgewinnung von Ressourcen in Produkt- und Materialkreisläufen*. Technische Universität Berlin, 2003, S. 235-276.

[Sel-04] Seliger, G.: *Montage und Demontage*. In: Beitz, W.; Grote, K.-H. (Hrsg.): *Dubbel Taschenbuch für den Maschinenbau*. 21. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2004.

[Sel-05] Seliger, G.; Uhlmann, E.; Friedrich, T.; Harms, R.: *Realization of an Adaptive Modular Control for a Disassembly System*. In: *Proceeding of the 6th IEEE International Symposium on Assembly and Task Planning (ISATP 2005)*. Montreal (Canada), 2005, Compact Disc.

[Sel-06a] Seliger, G.; Uhlmann, E.; Friedrich, T.; Harms, R.: *Pilot Disassembly System for Automotive Engines*. In: *Proceeding of the 39th CIRP International Seminar on Manufacturing Systems (ISMS 2006)*. Ljubljana (Slovenia), 2006, S. 63-66.

[Sel-06b]	Seliger, G.; Uhlmann, E.; Friedrich, T.; Harms, R.:	*Tools and Processes for Hybrid Disassembly of Automotive Engines*. In: *Proceeding of the 1st CIRP International Seminar on Assembly Systems (ISAS 2006).* Stuttgart, 2006, S. 209-214.
[Sin-01]	Sink, P.:	*Which bus, when?* URL: **http://machinedesign.com/article/which-bus-when-0712**. - Aktualisierungsdatum: 12.07.2001.
[Som-07]	Som, F.:	*Besonderheiten und Anforderungen sehr kleiner KMU am Beispiel eines Handwerkbetriebes*. In: Westkämper, E. (Hrsg.): *Industrieroboter schnell und einfach programmieren*. Stuttgart, 05.12.2007, Vortag.
[Spu-86]	Spur, G.; Helwig, H.-J.:	*Einführung in die Montagetechnik*. In: Spur, G.; Stöferle, T. (Hrsg.): *Handbuch der Fertigungstechnik*. Band 5. Fügen, Handhaben und Montieren. München, Wien: Carl Hanser Verlag, 1986.
[Spu-00]	Spur, G.; Uhlmann, E.; Härtwig, J.-P.; Seibt, M.:	*Pilot-Disassembly System for Home Appliances Using a New Twelve Degrees of Freedom Parallel Manipulator*. In: *Proceeding of the 33rd CIRP International Seminar on Manufacturing Systems*, Stockholm (Sweden), 2000, S. 13-16.
[Spu-79]	Spur, G.; Auer, B. H; Sinning, H.:	*Industrieroboter: Aufbau, Steuerung, Programmierung.* 1. Auflage. München, Wien: Carl Hanser Verlag, 1979.
[Spu-04]	Spur, G.; Uhlmann, E.:	*Industrieroboter*. In: Beitz, W.; Grote, K.-H. (Hrsg.): *Dubbel Taschenbuch für den Maschinenbau*. 21. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2004.
[Ste-01]	Stenzel, A.:	*Beitrag zum flexiblen Greifen in der Demontage*. Technische Universität Berlin. Dissertation, 2001.
[Sum-07]	Sumathi, S.; Esakkirajan, S.:	*Fundamentals of Relational Database Management Systems - Studies in Computational Intelligence*. 1. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2007.
[Tio-09]	N. N.:	*TIOBE Programming Community Index for August 2009.* URL: **http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html**. - Aktualisierungsdatum: 01.008.2009.
[Tsc-91]	Tschiesche, T.:	*Direkte Regelung von Industrierobotern*. Universität Duisburg-Essen, Dissertation, 1991.
[Uhl-04a]	Uhlmann, E.; Friedrich, T.; Früsch, I.:	*Possibilities for Improvement of the Automated Disassembly*. In: *Proceeding Global Conference on Sustainable Product Development and Life Cycle Engineering*. Berlin, 2004, S. 171-174.
[Uhl-04b]	Uhlmann, E.; Friedrich, T.; Früsch, I.:	*Strategies for Increasing the Efficiency of Disassembly Systems*. In: *Proceeding of the 11th International CIRP Life Cycle Engineering Seminar: Product Life Cycle – Quality Management Issues*. Belgrad (Serbia), 2004, S. 77-81.
[Uhl-07]	Uhlmann, E.; Friedrich, T.; Harms, R.; Sönnischen, C.:	*Hybrid Disassembly System*. In: Seliger, G. (ed.): *Sustainability in Manufacturing*. Berlin, Heidelberg, New York: Springer Verlag, 2007, S. 290-311.

[Uhl-08a] Uhlmann, E.; Friedrich, T.: *Development of a technology orientated programming system for industrial robots*. In: Production Engineering - Annals of the German Academic Society for Production Engineering, Vol. 2 (2008), No. 1, S. 103-108.

[Uhl-08b] Uhlmann, E.; Friedrich, T.: *Technologieorientiertes Programmier- und Steuerungskonzept für Industrieroboter*. In: Zeitschrift für wirtschaftlichen Fabrikbetrieb: ZWF 103 (2008) 6, S. 432-435.

[Val-95] von der Valk, U.: *Konzeption und Realisierung einer PC-basierten Robotersteuerung*. Technische Universität Dortmund, Dissertation, 1995.

[Ver-08a] Verl, A.; Naumann, M.: *Plug'n'Produce-Steuerungsarchitektur für Roboterzellen: Automatische Code-Generierung für Roboterzellen aus Prozess- und Geräte-Beschreibungen.* In: wt - Werkstattstechnik Online Nr. 98 (2008) 5, S. 384-390.

[Ver-08b] Verl, A.; Naumann, M.: *Kleine Losgrößen sicher im Griff: Flexible Architektur für Automatisierungssysteme.* In: Elektrotechnik: Das Automatisierungs-Magazin Nr. 90 (2008) 7/8, S. 64-68.

[Vos-00] Vossen, G.: *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*. 4. Auflage. München: Oldenbourg-Wissenschaftsverlag GmbH, 2000.

[Wah-94] Wahlster, W.; Bauer, M.; Biundo, S.; Dengler, D.; et al.: *Kooperative Hilfesysteme für Anwendungssoftware: Entwicklungsstand und Forschungstrends*. In: Jähnichen, S. (Hrsg.): *Innovative Softwaretechnologien: Neue Wege mit objektorientierten Methoden und Client/Server-Architekturen*. Velbert: Velbert Online Verlag, 1994, S. 401-426.

[Web-92] Weber, H.: *Konzeption und Realisierung einer hardwareunabhängigen Robotersteuerung mit offenen Programmier-Schnittstellen*. Technische Universität Dortmund, Dissertation, 1992.

[Wec-94] Weck, M.; Dammertz, R.: *OPERA – Offene Programmierumgebung zur Entwicklung von Roboterprogrammen*. In Pritschow, G.; Spur, G.; Weck, M. (Hrsg.): *Roboteranwendung für die flexible Fertigung*. München, Wien: Carl Hanser Verlag, 1994. S. 129-149.

[Wec-97] Weck, M.; Dammertz, R.: *Graphical robot programming and generation of application specific programming interfaces*. In: Production Engineering - Annals of the German Academic Society for Production Engineering, Karlsruhe, 4 (1997) 1, S. 87-90.

[Wec-03] Weck, M.; Almeida, C.: *Unscharfe Roboter-Programmierung: Roboter programmieren auf der Basis von ungenauen Angaben*. In: wt - Werkstattstechnik Online Nr. 93 (2003) 9, S. 627-631.

[Wec-04] Weck, M.; Matthias, B.; Almeida, C.; Schröter, B.; et al.: *Ein flexibles Robotersystem für Maschinenbeschickung und Materialhandhabung*. In: Gausemeier, J. (Hrsg.): *2. Paderborner Workshop*. Paderborn: März 2004.

[Wec-05]	Weck, M.; Almeida, C.:	*Unscharfe Roboterprogrammierung - Programmierung von sensorgeführten Handhabungsaufgaben*. In: Adam, W. (Hrsg.): *Neue Sensoren und Aktoren für produktionstechnische Anwendungen*. Düsseldorf: VDI Verlag, 2005, S. 1-16.
[Wec-06]	Weck, M.:	*Werkzeugmaschinen Band 4: Automatisierung von Maschinen und Anlagen*. 6. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2006.
[Wol-04]	Wolf, A.:	*Greifer in Bewegung - Faszination der Automatisierung von Handhabungsaufgaben*. 1. Auflage. München, Wien: Carl Hanser Verlag, 2004.
[Wör-04]	Wörn, H.:	*Entwicklungstendenzen in der Industrierobotik*: In: *VDE Kongress 2004 Berlin*. Offenbach: VDE-Verlag, 2004, S. 239-244.
[Wör-06]	Wörn, H.; Brinkschulte, U.:	*Echtzeitsysteme - Grundlagen, Funktionsweisen, Anwendungen*. 1. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2006.
[Zab-93]	Zabel, A.:	*Werkstattorientierte Programmierung von Industrieroboternfür automatisiertes Lichtbogenschweißen*. Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 1993.
[Zac-00]	Zacher, S.:	*Automatisierungstechnik kompakt: Theoretische Grundlagen, Entwurfsmethoden, Anwendungen*. 1. Auflage. Wiesbaden: Vieweg Verlagsgesellschaft, 2000.
[Zäh-04]	Zäh, M.; Vogl, W.; Munzert, U.:	*Beschleunigte Programmierung von Industrierobotern*. In: wt - Werkstattstechnik Online Nr. 94 (2004) 9, S. 438-441.
[Züh-04]	Zühlke, D.:	*Useware-Engineering für technische Systeme.* 1. Auflage. Berlin, Heidelberg, New York: Springer-Verlag, 2004.

Normen und Richtlinien

[EG-2006]	Maschinenrichtlinie 2006/42/EG	Richtlinie 2006/42/EG des europäischen Parlaments und des Rates vom 17. Mai 2006 über Maschinen und zur Änderung der Richtlinie 95/16/EG. Amtsblatt der Europäischen Union
[DIN199-1]	DIN 199, Teil 1:	Technische Produktdokumentation. CAD-Modelle, Zeichnungen und Stücklisten. Teil 1 - Begriffe. Berlin: Beuth, 2002-03.
[DIN 8580]	DIN 8580	Fertigungsverfahren. Begriffe, Einleitung. Berlin: Beuth, 2003-09.
[DIN19226-1]	DIN 19226, Teil 1:	Leittechnik. Regelungstechnik und Steuertechnik. Teil 1 - Allgemeine Grundbegriffe. Berlin: Beuth, 1994-02.
[DIN19226-5]	DIN 19226, Teil 5:	Leittechnik. Regelungstechnik und Steuertechnik. Teil 5 - Funktionelle Begriffe. Berlin: Beuth, 1994-02.

[ISO 8373]	DIN EN ISO 8373:	Industrieroboter Wörterbuch. Berlin: Beuth, 1996-08.
[VDI 2343-3]	VDI 2343, Blatt 3	Recycling elektrischer und elektronischer Geräte: Demontage und Aufbereitung. Düsseldorf: VDI, 2002-02.
[VDI 2863]	VDI 2863, Blatt 1	Programmierung numerisch gesteuerter Handhabungs-einrichtungen: IRDATA; Allgemeiner Aufbau, Satztypen und Übertragung. Düsseldorf: VDI, 1987-12.

10 Anhang

A 1 - Übersicht typischer Elementaranweisungen

Gruppe	Ident-Nummer	Benennung	Ereignis	Anwendung im TOPS
Elementar-Bewegungsanweisung	100	PTP_Base _Ohne Ereignis	-	X
	101	PTP_Base _Mit Ereignis	Stopp wenn Sensor1 = „true“	
	102	PTP_Base _Mit Ereignis	Ausgang1 = „true“ wenn Sensor2 = „true“	
	103	PTP_Base _Mit Ereignis	Stopp wenn Sensor1 = „true“ und Sensor2 = „true“	
	110	PTP_Achsspezifisch _Ohne Ereignis	-	
	111	PTP_Achsspezifisch _Mit Ereignis	Stop wenn Sensor1 = „true“	
	112	PTP_Achsspezifisch _Mit Ereignis	Ausgang1 = „true“ wenn Sensor2 = „true“	
	113	PTP_Achsspezifisch _Mit Ereignis	Stopp wenn Sensor1 = „true“ und Sensor2 = „true“	
	120	LIN_Base _Ohne Ereignis	-	X
	121	LIN_Base _Mit Ereignis	Stopp wenn Sensor1 = „true“	X
	122	LIN_Base _Mit Ereignis	Ausgang1 = „true“ wenn Sensor2 = „true“	
	123	LIN_Base _Mit Ereignis	Stopp wenn Sensor1 = „true“ und Sensor2 = „true“	X
	130	LIN_Achsspezifisch _Ohne Ereignis	-	
	131	LIN_Achsspezifisch _Mit Ereignis	Stopp wenn Sensor1 = „true“	
	132	LIN_Achsspezifisch _Mit Ereignis	Ausgang1 = „true“ wenn Sens2 = „true“	
	133	LIN_Achsspezifisch _Mit Ereignis	Stopp wenn Sensor1 = „true“ und Sensor2 = „true“	
	140	CIRC_Base _Ohne Ereignis	-	X
	141	CIRC_Base _Mit Ereignis	Stopp wenn Sensor1 = „true“	

Gruppe	Ident-Nummer	Benennung	Ereignis	Anwendung im TOPS
Elementar-Bewegungsanweisung	142	CIRC_Base _Mit Ereignis	Ausgang1 = „true“ wenn Sensor2 = „true“	
	143	CIRC_Base _Ohne Ereignis	Stopp wenn Sensor1 = „true“ und Sensor2 = „true“	
	150	CIRC_Achsspezifisch _Mit Ereignis	-	
	151	CIRC_Achsspezifisch _Mit Ereignis	Stopp wenn Sensor1 = „true“	
	152	CIRC_Achsspezifisch _Mit Ereignis	Ausgang1 = „true“ wenn Sensor2 = „true“	
	153	CIRC_Achsspezifisch _Mit Ereignis	Stopp wenn Sensor1 = „true“ und Sensor2 = „true“	
Elementar-Kontroll-anweisungen	400	$OUT	digitaler Ausgang = „true“	X
	410	Pulse	digitaler Ausgang = temporär „true“ (1 sec)	
	411	Pulse_Temporär	digitaler Ausgang = „true“ bis Sensor2 = „true“	
	412	$OUT_Abfragen	digitaler Ausgang = „true“ oder „false“	X
	413	$IN_Abfragen	digitaler Eingang = „true“ oder „false“	
	414	$IN_Abfragen_bis_ Ereignis_Zeit (sec)	digitaler Eingang = „true“ oder „false“ (definierter Zeitraum)	
	415	$IN_Abfragen_bis Ereignis_$IN (Sense2) = „true“	digitaler Eingang = „true“ oder „false“ bis zweiter digitaler Eingang = „true“	
	416	Wait	Wartefunktion	
Elementar-Diagnose-anweisungen	500	$POWERMODUL	Informationen zum Powermodul	
	501	$POS_ACT_ Aktuelle IST-Position des Roboters	aktuelle kartesische Roboterposition	
	502	AXIS_ACT	aktuelle achsspezifische Roboterposition	
	503	$JUS_TOOL_NO	Nummer des aktuellen Werkzeugs bei EMT-Justage	

A 2 - Parameter der Elementaranweisungen

Parameter „Basis“ besitzt die Komponenten: Basis_X, Basis_Y, Basis_Z, Basis_A, Basis_B, Basis_C

Parameter „Pkt_1/Achsen“ besitzt die Komponenten: Punkt1_X/A1, Punkt1_Y/A2, Punkt1_Z/A3, Punkt1_A/A4, Punkt1_B/A5, Punkt1_C/A6

Parameter „Pkt_2“ besitzt die Komponenten: Punkt2_X, Punkt2_Y, Punkt2_Z, Punkt2_A, Punkt2_B, Punkt2_C

Ident-Nummer	Parameter einer Elementaranweisung																	
	Basis	Pkt_1/ Achsen	Pkt_2	WZ_Nr	PTP_Vel bzw. Zeit (sec)	LIN_Vel	Sensor1 bzw. $IN1/$OUT1	Sensor2 bzw. $IN2/$OUT2										
100	X	X		X	X													
101	X	X		X	X		X	X										
102	X	X		X	X		X	X										
103	X	X			X		X	X										
110		X			X													
111		X			X		X											
112		X			X		X	X										
113		X			X		X	X										
120	X	X		X		X												
121	X	X		X		X	X											
122	X	X		X		X	X	X										
123	X	X		X		X	X	X										
130		X				X												
131		X				X	X											
132		X				X	X	X										
133		X				X	X	X										
140	X	X	X	X		X												
141	X	X	X	X		X	X											

Ident-Nummer	Parameter einer Elementaranweisung							
	Basis	Pkt_1/ Achsen	Pkt_2	WZ_Nr	PTP_Vel bzw. Zeit (sec)	LIN_Vel	Sensor1 bzw. $IN1/$OUT1	Sensor2 bzw. $IN2/$OUT2
142	X	X	X	X	X	X	X	
143		X	X	X	X	X	X	
150		X	X		X			
151		X	X		X		X	
152		X	X		X		X	
153		X	X		X		X	
400							X	
410							X	
411							X	
412							X	
413							X	
414							X	X
415					X			
416								
500				X				
501		X						
502		X						
503				X				

A 3 - Darstellung des Prozesses „Demontage Seitenwand mit Zerschleifwerkzeug“ für KUKA II

Demontage Seitenwand (Zerschleifwerkzeug) – Teil I					
Prozessschritt	**Ident-Nummer**	**Dynamischer Parameter**	**Statische Parameter**		
		Wert	Base	WZ_Nr.	Vel.
Fahren in sichere Achsstellung	100	60; -120; 115; -180; -75; 0	default	0	100
Fahren nach „Save-Point 4“	100	400	default	0	100
Anfahren „WZ Aufnehmen“	100	0; 0; -300; 0; 0; 0	WW-Base-Flex	0	75
Anfahren „WZ Aufnehmen“	111	0; 0; -50; 0; 0; 0	WW-Base-Flex	0	0,2
Anfahren „WZ Aufnehmen“	111	0; 0; -10; 0; 0; 0	WW-Base-Flex	0	0,1
Anfahren „WZ Aufnehmen“	111	0; 0; -5; 0; 0; 0	WW-Base-Flex	0	0,02
Verriegelung des WZ-Wechslers	400	2 = false	default	0	0
Verriegelung des WZ-Wechslers	400	1 = true	default	0	0
Kontrolle der Verriegelung	412	2 = false	default	0	0
Kontrolle der Verriegelung	412	1 = false	default	0	0
Wartezeit	416	-	default	0	1,5 (sec)
Abfahren „WZ Aufnehmen“	120	0; 250; 0; 0; 0; 0	WW-Base-Flex	0	0,1
Abfahren „WZ Aufnehmen“	110	-400; 350; -350; 0; 0; 0	WW-Base-Flex	0	0,25
Fahren nach „Save-Point 4“	110	60; -120; 115; 180; -75; 0	default	0	75
Synchronisationsmarke System					
Synchronisationsmarke KUKA III					
Fahren nach „Start-punkt für WZ ein-schalten“	100	250; 250; -400; 0; 0; 0	Base Dreh-tisch [0grad] / Base Palette / Ebene 0	4	75
Fahren nach “Startpunkt für WZ ein-schalten“	120	150; 150; -100; 0; 0; 0	Base Dreh-tisch [0grad] / Base Palette / Ebene 0	4	0,5
WZ einschalten	400	17 = true	default	4	0

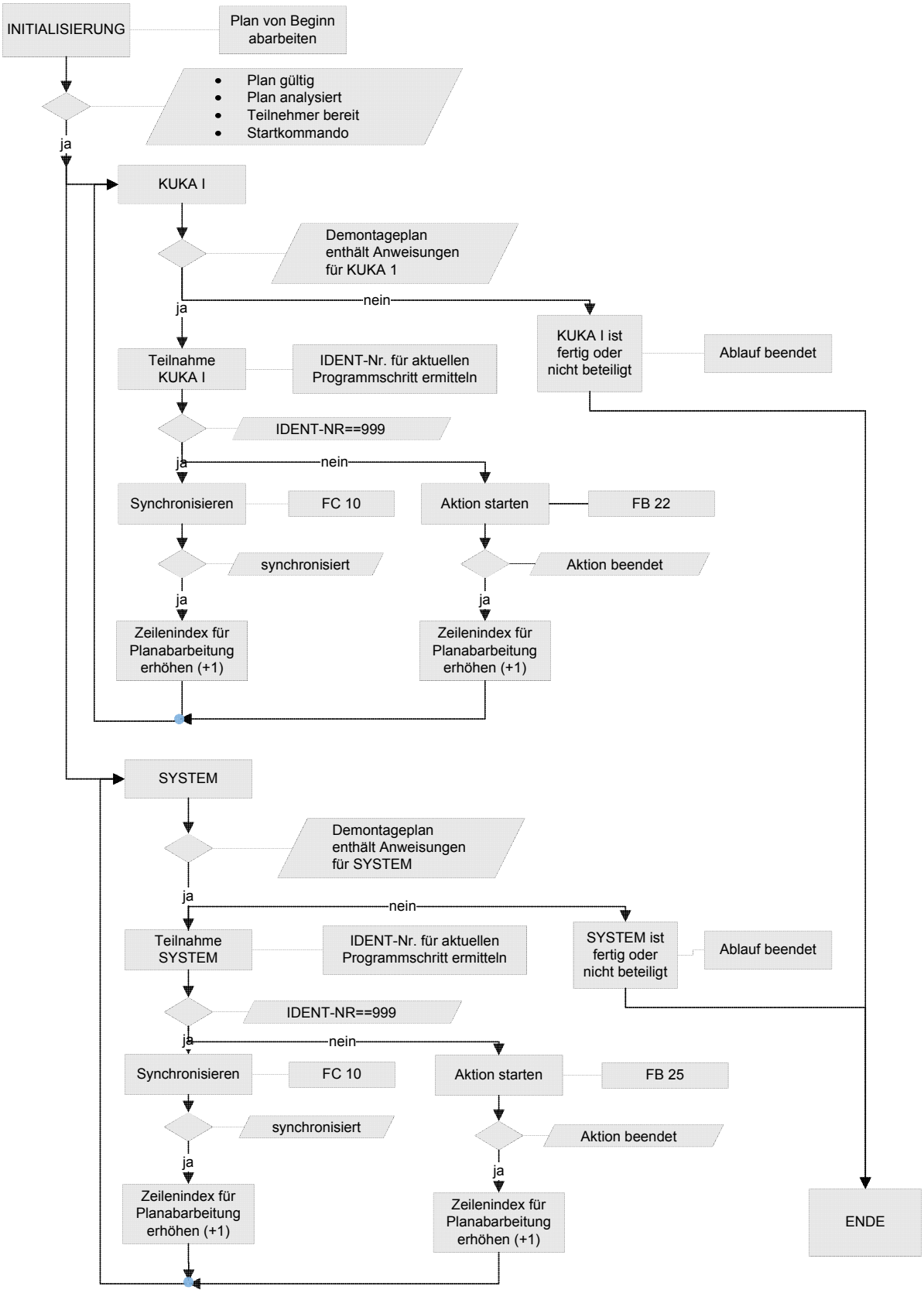
Demontage Seitenwand (Zerschleifwerkzeug) – Teil II					
Prozessschritt	**Ident-Nummer**	**Dynamischer Parameter**	**Statische Parameter**		
		Wert	Base	WZ_Nr.	Vel.
Orientierungsfahrt	111	150; 150; -30; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Eintauchfahrt	111	50; 150; 10; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Schnittfahrt	111	150; 450; 10; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Austauchfahrt	111	150; 450; -30; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Orientierungsfahrt	111	150; 450; -30; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Eintauchfahrt	111	150; 450; 10; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Schnittfahrt	111	600; 450; 10; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Austauchfahrt	130	600; 450; -30; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Orientierungsfahrt	130	600; 450; -30; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Eintauchfahrt	111	600; 450; 10; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Schnittfahrt	111	600; 150; 10; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Austauchfahrt	111	600; 150; -30; 0; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Orientierungsfahrt	111	600; 150; -30; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Eintauchfahrt	111	600; 150; 10; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06
Schnittfahrt	111	150; 150; 10; -90; 0; 0	Base Drehtisch [0grad] / Base Palette / Ebene 0	4	0,06

Demontage Seitenwand (Zerschleifwerkzeug) – Teil III					
Prozessschritt	**Ident-Nummer**	**Dynamischer Parameter**	**Statische Parameter**		
		Wert	Base	WZ_Nr.	Vel.
Fahren nach „Start-punkt für Werkzeug ausschalten“	111	250; 250; -400; 0; 0; 0	Base Dreh-tisch [0grad] / Base Palette / Ebene 0	4	0,2
WZ ausschalten	400	17= false	default	4	0
Kontrolle, ob WZ ausgeschaltet	416	17 = false	default	4	0
Synchronisationsmarke System					
Synchronisationsmarke KUKA III					
Fahren nach Save-Point 4	110	60; -120; 115; -180; -75; 0	default	0	75
Anfahren „WZ ablegen“	100	-400; 350; -350; 0; 0; 0	WW-Base-Flex	0	75
Anfahren „WZ ablegen“	111	0; 250; 0; 0; 0; 0	WW-Base-Flex	0	0,2
Anfahren „WZ ablegen“	111	0; 130; 0; 0; 0; 0	WW-Base-Flex	0	0,1
Anfahren „WZ ablegen“	111	0; 0; 5; 0; 0; 0	WW-Base-Flex	0	0,02
Entriegelung des WZ-Wechslers	400	1 = false	default	0	0
Entriegelung des WZ-Wechslers	400	2 = true	default	0	0
Kontrolle der Entriegelung	412	1 = false	default	0	0
Kontrolle der Entriegelung	412	2 = false	default	0	0
Wartezeit	416	-	default	0	1,5 (sec)
Abfahren „WZ ablegen“	120	0; 0; -50; 0; 0; 0	WW-Base-Flex	0	0,1
Abfahren „WZ ablegen“	120	0; 0; -300; 0; 0; 0	WW-Base-Flex	0	0,25
Fahren nach Save-Point 4	110	60; -120; 115; -180; -75; 0	default	0	75

A 4 - Definition des Datenbausteins für die Zwischenspeicherung des Demontageplans eines Industrieroboters

```
DATA_BLOCK “TOPS_KUKA_II”
STRUCT
        KUKA_II_Base_x: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Base_y: ARRAY [0 … 40] OF DINT;
        KUKA_II_Base_z: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Base_a: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Base_b: ARRAY [0 … 40] OF DINT;
        KUKA_II_Base_c: ARRAY [0 … 40] OF DINT;
        KUKA_II_Point1_x: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point1_y: ARRAY [0 … 40] OF DINT;
        KUKA_II_Point1_z: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point1_a: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point1_b: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point1_c: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point2_x: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point2_y: ARRAY [0 … 40] OF DINT;
        KUKA_II_Point2_z: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point2_a: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point2_b: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Point2_c: ARRAY [0 ... 40] OF DINT;
        KUKA_II_Lin_Vel ARRAY [0 ... 40] OF DINT;
        KUKA_II_Ident: ARRAY [0 ... 40] OF INT;
        KUKA_II_Sens_1: ARRAY [0 ... 40] OF INT;
        KUKA_II_Sens_2: ARRAY [0 ... 40] OF INT;
        KUKA_II_Sens_3: ARRAY [0 ... 40] OF INT;
        KUKA_II_PTP_Vel: ARRAY [0 … 40] OF INT;
        KUKA_II_Tool_No: ARRAY [0 ... 40] OF BYTE;
END_STRUCT;
```

A 5 - Ablaufdiagramm „TOPS-Steuermodul“ für die Komponenten „KUKA II“ und „SYSTEM“



A 6 - Darstellung der Datenbankstruktur