PhD-FSTC-2019-16
Fakultät für Naturwissenschaften, Technologie und Kommunikation

DISSERTATION

verteidigt am 01/04/2019 in Luxemburg

zur Erlangung des Titels

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN SCIENCES DE L'INGENIEUR

von

Jan Jungbluth
geboren am 25 September 1985 in Birkenfeld (Deutschland)

ENTWICKLUNG EINES INTELLIGENTEN, ROBOTERGESTÜTZTEN ASSISTENZSYSTEMS FÜR DIE DEMONTAGE INDUSTRIELLER PRODUKTE

Prüfungskommission

Prof. Dr. Peter Plapper
Professor, Universität Luxemburg

Prof. Dr.-Ing. Holger Voos
Professor, Universität Luxemburg

Prof. Dr.-Ing. Wolfgang Gerke
Professor, Hochschule Trier - Umwelt Campus Birkenfeld

Prof. Dr. Gabriel Abba
Professor, Université de Lorraine

Prof. Dr.-Ing. Rainer Müller
Professor, Universität Saarbrücken
Geschäftsführer, Zentrum für Mechatronik und Automatisierungstechnik, Saarbrücken

Kurzfassung

In vielen Bereichen des privaten Umfelds haben technische Assistenzsysteme ihren Platz eingenommen und vereinfachen das tägliche Leben. Diese Tendenz wäre auch im Arbeitsleben wünschenswert, besonders in körperlich belastenden Tätigkeiten wie der Demontage von Produkten in der Wartung, Instandsetzung oder Refabrikation.

Doch der Einsatz robotergestützter Assistenzsysteme wird durch die Unwägbarkeiten im Demontageprozess unterschiedlicher Produkte wegen der fehlenden Autonomie der technischen Systeme verhindert.

Im Rahmen dieser Dissertation wird daher die Entwicklung von intelligenten, robotergestützten Assistenzsystemen betrachtet, die den Menschen in solch komplexen Prozessen zielgerichtet unterstützen können. Dabei wird die folgende Forschungsfrage gestellt: Welche technischen Anforderungen werden an ein solches Assistenzsystem gestellt und wie können sie umgesetzt werden?

In der gesichteten wissenschaftlichen Literatur konnte für die Entwicklung dieser Systeme kein ganzheitlicher Ansatz ermittelt werden. Es wurden jedoch viele Ansätze zu Teilaspekten eines solchen Systems, über mehrere Forschungsdisziplinen hinweg, zusammengetragen.

Zur Klärung der Forschungsfrage erfolgt in dieser Dissertation die Darstellung der Themengebiete der technischen Assistenzsysteme, der Mensch-Roboter-Systeme sowie des Einsatzgebiets, um technische Anforderungen zu definieren. Deren Umsetzung in einem Demonstrator zur experimentellen Validierung erfolgte in Form eines Multiagentensystems, in das verschiedene technische Systeme durch Software integriert und vernetzt wurden. Weiterhin wurde die notwendige technische Autonomie des Assistenzsystems auf Basis zweier modellbasierten Planungssysteme sowie des zentralen Steuerungssystems durch Software implementiert. In Verbindung mit geeigneten Mensch-Maschine-Kommunikationsschnittstellen, konnte die Funktion des Assistenzsystem in konkreten Demontageprozessen verifiziert werden.

Jedoch zeigte sich auch, dass für die Einführung solcher Systeme im industriellen Umfeld noch weitere herausfordernde Forschungsfragen bearbeitet werden müssen.

Abstract

Technical assistance systems are used in many aspects of our private environment to simplify our daily lives. Such assistance would also be desirable in our working environment, especially in physically demanding activities such as dismantling products for maintenance, corrective maintenance, or remanufacturing.

However, the use of robot-supported assistance systems is prevented by the imponderabilities in the dismantling process of different products due to the lack of autonomy of the technical systems.

In the course of this dissertation, the development of intelligent, robot-supported assistance systems that can support people in such complex processes in a target-oriented manner are considered, and the following research question is posed: What are the technical requirements for such assistance systems and how can they be implemented?

In the reviewed scientific literature, no holistic approach has been identified for the development of these systems, but many approaches to partial aspects of such a system have been collected across several research disciplines.

In order to address the research question, this dissertation discusses the theoretical fields of technical assistance systems, human-robot systems, and the field of application in order to define technical requirements. A demonstrator for experimental validation is implemented in the form of a multi-agent system in which various technical systems are integrated and interconnected by software. The function of the developed robot-based assistance system could be verified in concrete dismantling processes in conjunction with suitable man-machine communication interfaces.

Finally, this dissertation identifies further research questions that must be addressed before such systems can be introduced in the industrial environment.

Vorwort

Die vorliegende Arbeit entstand innerhalb eines vierjährigen Forschungsprojekts in Kooperation mit der Universität Luxemburg, der Hochschule Trier sowie der Firma SEW Eurodrive. Als Angestellter der SEW Eurodrive arbeitete ich während dieser Zeit am Institut für Betriebs- und Technologiemanagement am Umwelt-Campus Birkenfeld im Fachbereich Umweltplanung und Umwelttechnik und war als Ph.D. Student Teil der Research Unit in Engineering Sciences (RUES) an der Fakultät für Naturwissenschaften, Technologie und Kommunikation (FTSC) der Universität Luxemburg. In dieser Zeit hatte ich das Vergnügen, mit vielen Menschen zusammenzuarbeiten, an die ich nun dankende Worte richten möchte.

Herzlich bedanken möchte ich mich zunächst bei den Professoren des Prüfungskomitees. Ganz besonders bedanken möchte ich mich in diesem Zusammenhang bei den drei CET-Komiteemitgliedern Prof. Peter Plapper, Prof. Wolfgang Gerke und Prof. Holger Voos. Mein besonderer Dank gilt Herrn Prof. Plapper sowie Herrn Prof. Gerke, die mir als Betreuer während der Dissertation wertvolle fachliche sowie strategische Kompetenzen vermittelt haben und zu erheblichen Maßen zum erfolgreichen Abschluss meiner Promotion beigetragen haben. Ein weiterer Dank geht an meinen Arbeitgeber, insbesondere Herrn Josef Schmidt, der als Leiter der Abteilung ‚Forschung und Technik' Fördergeber des Forschungsprojekts war, und an meinen Betreuer Herrn Joachim Stieber.

Des Weiteren möchte ich mich bei einigen Arbeitskollegen, Studenten und Mitarbeitern am Umwelt-Campus und der Universität Luxemburg bedanken. Danke: Stefan Schmitt, Sebastian Groß, Thomas Bartscherer, Florian Schäfer, Karsten Siedentopp, Markus Tilch, Christopher Hut, Christan Seibert, Carsten Böhmer, Sophie Klecker, Bassem Hichri, Abir Gallala und alle anderen die mich während dieser Zeit unterstützt haben.

Abschließend möchte ich mich bei meiner Familie für den Rückhalt und die wertvolle Unterstützung auf dem Weg zu meiner Promotion bedanken.

Birkenfeld, den 12.2.2019

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Liste mathematischer Symbole

Themengebiet	Symbol	Bedeutung
Aussagenlogik	a	Aussage, die wahr oder falsch sein kann
	$\neg a$	Negierung der Aussage a
	$a \wedge b$	Konjunktion, (a und b)
	$a \vee b$	Disjunktion, (a oder b)
Prädikatenlogik	$P(x)$	Prädikat mit der freien Variable x. Z. B. x ist Element der ganzen Zahlen.
	$\forall x P(x)$	Allquantor, z.B. alle natürlichen Zahlen x sind Elemente der ganzen Zahlen
	$\exists x P(x)$	Existenzquantor, z.B. es existieren rationale Zahlen x, die Element der ganzen Zahlen sind.
	$\exists ! x P(x)$	Einzigkeitsquantor, es existiert genau ein x das das Prädikat erfüllt.
	$\neg \exists x P(x)$	Nichtexistenz, es existiert kein x das das Prädikat erfüllt.
Mengen	$A = \{a_1, a_2, .., a_n\}$	Definition einer Menge A mit Elementen a_i, i = 1, 2, ..., n
	$x \in A$	x ist Element der Menge A
	$x \notin A$	x ist kein Element der Menge A
	$A = B$	Gleichheit von Mengen
	$C \subset A$	C ist Teilmenge von A
	$D \subseteq A$	D ist echte Teilmenge von A

	$\varnothing$	Leere Menge
	$B = \{x \in A \mid P(x)\}$	Mengenbildung durch Auswahl
	(x, y)	Geordnetes Paar
	$(a_1, a_2, ..., a_n)$	N-Tupel
	$A \cup B$	Vereinigung zweier Mengen
	$A \cap B$	Schnittmenge zweier Mengen
	$A \times B$	Kartesische Produkt zweier Mengen
	$A - B$	Differenz zweier Mengen
	$\bigcup_{i=1}^{n} A_i$	Vereinigung der Mengen A_i, i = 1, 2, .., n
	$\bigcap_{i=1}^{n} A_i$	Schnittmenge der Mengen A_i, i=1, 2, .., n
Relationen	r^n	Relation zwischen n Mengen, n wird auch als Arität der Relation bezeichnet
Abbildungen	$g : X \to Y$	Jedem Element x der Menge X wird genau ein Element y der Menge Y zugeordnet
	$g(x) = y$	Dem Element x zugeordnete Element y
Funktionen	$f : X \to Y$	Jedem Element x aus der Zahlenmenge X wird genau ein Element y der Zahlenmenge Y zugeordnet
	$f(x) = y$	Der Zahl x zugeordnete Zahl y

1 Einleitung

Die Verfügbarkeit und Marktreife von günstigen, einfach zu bedienenden Leichtbaurobotern die mit dem Menschen ohne trennende Schutzeinrichtungen zusammenarbeiten können, leitete einen Wandel im Bereich der industriellen Robotik und der robotergestützten Automatisierung ein [1]. Industrieroboterapplikationen sollen menschliche Arbeit nicht länger substituieren, sondern diese unterstützen. Das Konzept der Assistenz wurde im industriellen Umfeld neu entdeckt. Dieser Gedanke lässt sich in den Produktnamen vieler Hersteller wiederfinden, zum Beispiel beim Industrieroboter Kuka LBR iiwa (*intelligent industrial work assistant*) oder Boschs APAS (Automatischer Produktionsassistent). Durch Marketing der Roboterhersteller wurden allerdings zu früh hohe Erwartungen an diese neue Technologie erzeugt. Denn für die Planung, Konzeption und Entwicklung von Mensch-Roboter-Systemen im industriellen Umfeld fehlten die Erfahrungen [2], insbesondere das Thema Sicherheit wurde durch fehlende Normung in der Anfangszeit intensiv diskutiert und sorgte für Verwirrung [3]. Durch Normung [4–7], praktischer sowie wissenschaftlicher Pionierarbeit sind die Hürden der Anfangszeit überwunden, jedoch verblieb Ernüchterung bzgl. dieser neuen Technologie, die teilweise auch aus den hohen Erwartungen resultiert [2, 8].

Bei der Planung und Entwicklung wurde dabei klar, dass viele Aspekte solcher Mensch-Roboter-Systeme, wie die Mensch-Roboter-Interaktion (MRI), noch nicht vollständig verstanden sind. Dadurch fehlen insbesondere für die Zusammenarbeit in komplexen Prozessen, die keinem fest definierten Ablauf folgen, technologische Vorgaben für deren Entwicklung. Doch gerade in komplexen Prozessen wünschen sich Menschen technische Unterstützung. Hier könnten robotergestützte Assistenzsysteme (RAS) die Arbeitsergonomie für Menschen verbessern, zum Beispiel, indem der Assistent Aufgaben temporär übernimmt oder den Menschen bei einer Tätigkeit unterstützt. Insbesondere unter Berücksichtigung einer alternden Gesellschaft und einer längeren Erwerbstätigkeit ist die physische Entlastung des Arbeitnehmers ein Zukunftsthema. Auch im Hinblick auf die zunehmende Individualisierung von Produkten und deren Komplexität stellt die psychische (Entscheidungs-)Unterstützung im Arbeitsprozess eine sich verschärfende Thematik dar. In diesem Kontext sollte auch an die Möglichkeit der Inklusion von Menschen mit speziellen Bedürfnissen in industriellen Feldern durch angepasste Unterstützung gedacht

werden [9]. Die Entwicklung von RAS als innovative Systemlösungen für den kommerziellen Markt stellt also einen erheblichen wirtschaftlichen und gesellschaftlichen Anreiz dar. Die Wahl, für dieses Anwendungsgebiet erfolgte auch zugunsten des Umweltschutzes. Durch die Einführung von RAS wird eine Verbesserung der Wirtschaftlichkeit der zerstörungsfreien Demontage erwartet. Die kostengünstigere Demontage erweitert wiederum das Spektrum von Produkten, für die ein wirtschaftliche Refabrikation stattfinden kann. Die Refabrikation erspart der Umwelt die zur Herstellung notwendige Energie, Arbeit und Rohstoffe. Somit könnte diese Entwicklung in erheblichem Maße zum Schutz der Umwelt beitragen.

1.1 Problemstellung

Die konkrete Problemstellung, die in dieser Arbeit besteht, ist die zerstörungsfreie Demontage von industriellen Produkten. Bei der Demontage des Produktes soll der Mensch durch ein intelligentes RAS unterstützt werden. Die Unterstützung soll dabei nicht punktuell, sondern über den gesamten Demontageprozess erfolgen. Erschwert wird diese Problemstellung durch die Vielfalt des Aufbaus industrieller Produkte, durch die möglichen Zielstellungen hinsichtlich der Demontage sowie durch den unbekannten Produktzustand zum Zeitpunkt der Demontage. Die Vielfalt der vorkommenden Bauteilgeometrien und Verbindungstechniken erschwert weiterhin den Einsatz von Industrierobotern, Werkzeugen und Peripherie. Weiterhin kommt durch die Beteiligung des Nutzers der ‚menschliche Faktor[1]' hinzu (engl.: human factor), der die ganzheitliche Betrachtung des Gesamtsystems von Mensch-und-Maschine erfordert. Klassische Ansätze der Automation sind für diese Herausforderungen nicht geeignet, was zu folgenden Forschungsfrage(n) Dissertation führt.

[1] Unter dem Begriff ‚Menschlicher Faktor' werden psychische, kognitive und soziale Einflussgrößen in Mensch-Maschinen-Systemen zusammengefasst [10].

1.2 Forschungsfrage(n) und -methodik

Als Voraussetzung für die Mensch-Maschine-Zusammenarbeit in komplexen Prozessen ist allgemein anerkannt [10–12], dass hierzu ein geeignetes technisches System und eine für die Problemstellung passende technische Autonomie erforderlich ist – was zu den folgenden Forschungsfragen führt:

Forschungsfrage 1: Welche technischen Anforderungen werden an ein solches Assistenzsystem gestellt?

Forschungsfrage 2: Wie kann eine geeignete Systemarchitektur für das Assistenzsystem aussehen?

Forschungsfrage 3: Wie kann die technische Autonomie des Assistenzsystems gestaltet werden?

Um diese Fragestellungen zu untersuchen, erfolgt eine Sichtung der vorhandenen Literatur, dem Stand der Technik sowie die Betrachtung des Anwendungsgebiets um die Anforderungen an das Assistenzsystem zu ermitteln. Anhand der Anforderungen erfolgt anschließend der Systementwurf sowie die Umsetzung in einem Demonstrator. Um aussagekräftige Ergebnisse zu erhalten erfolgt, in Form eines Laborexperiments, anschließen die Verifikation des Assistenzsystems in realen Demontageanwendungen. Als Methodik für die Entwicklung des RAS wird sich am Prozess des System Engineering nach (ISO/IEC 15288) orientiert und die folgenden technischen Prozesse betrachtet:

- Definition der Systemanforderungen
- Architekturgestaltung und Entwurf
- Systemanalyse
- Umsetzung von Teilsystemen
- Integration der Teilsysteme in das Gesamtsystem
- Verifikation und Validierung des Gesamtsystems

1.3 Zielsetzung und Aufbau dieser Arbeit

Die Zielsetzung dieser Dissertation ist es, ein ganzheitliches Konzept für robotergestützte Demontage-Assistenzsysteme zu entwickeln und in einer geeigneten Systemarchitektur abzubilden, sowie im System eine für das Einsatzgebiet geeignete technische Autonomie zu implementieren. Weiterhin soll im Rahmen dieser Dissertation das, aus mehreren Forschungsgebieten stammende, notwendige Wissen komprimiert zusammengefasst und hinsichtlich der Problemstellung interpretiert werden. Der Aufbau dieser Arbeit gliedert sich weiter wie folgt:

Im Kapitel 2 wird der Stand der Technik bzgl. des Einsatzes von robotergestützten Assistenzsystemen in Demontageprozessen im Bereich der Wartung, Instandsetzung und Refabrikation dargestellt. Weiterhin erfolgt die Betrachtung des Stands der Technik im Themengebiet der Mensch-Roboter-Systeme anhand aktueller Forschungsprojekte mit industriellen Anwendungsfeld.

In Kapitel 3 erfolgt, eine Einführung in das Themengebiet der technischen Assistenzsysteme sowie der Mensch-Roboter-Systeme und der Mensch-Roboter-Interaktion.

Dies erfolgt auch mit Hinblick zur Ermittlung technischer Anforderungen im Einsatzgebiet des Assistenzsystems, das hierzu in Kapitel 4 dargestellt wird.

Für die Umsetzung dieser technischen Anforderungen erfolgt im Kapitel 5.1 die Betrachtung von Architekturen intelligenter Systeme aus dem Bereich der Robotik, Informatik und Kybernetik mit dem Ziel, eine geeignete Systemarchitektur für die Entwicklung des RAS zu bestimmen. Weiterhin erfolgt im Kapitel 5.2 die Darstellung einer Methodik zur Umsetzung technischer Autonomie auf Grundlage von Planungssystemen.

Im Kapitel 6 erfolgt die detaillierte Erläuterung der Umsetzung des RAS durch eine allgemeine Beschreibung der Systemarchitektur sowie der darin gebildeten Abstraktionsebenen für Teilprobleme. Durch die Darstellung der entwickelten Softwarearchitekturen und der darin gebildeten Module, Modelle und Algorithmen zur Lösung der Teilprobleme, kann das ganzheitliche Konzept des RAS dargestellt werden.

In Kapitel 7 wird der umgesetzte Demonstrator für die experimentelle Validierung des Ansatzes beschrieben. Hierzu wird der Aufbau des zu demontierenden industriellen Produkts (ein mechatronisches Antriebssystem) sowie die einzelnen technischen Systeme

und deren Integration in das RAS dargestellt. Weiterhin werden vier unterschiedliche Demontageprozesse beschrieben und bewertet.

Am Ende dieser Arbeit, in Kapitel 8, erfolgen eine Zusammenfassung sowie ein Ausblick auf zukünftige Forschungsaktivitäten.

2 Stand der Technik

In diesem Kapitel soll der Stand der Technik hinsichtlich des Einsatzes von RAS in industriellen Demontageprozessen betrachtet werden. Weiterhin wird der Stand der Technik im Bereich der Mensch-Roboter-Kollaboration beschrieben.

2.1 Stand der Technik in der industriellen Demontage

Demontageprozesse im industriellen Einsatz spalten sich bzgl. des Automatisierungsgrades in zwei Gruppen: in Vollautomatisierung und rein manuelle Arbeit [12]. Eine vollautomatisierte Demontagelinie konnte für Mobiltelefone [13, 14] ermittelt werden. Die in [13] beschriebene vollautomatisierte Demontagelinie besteht aus 29 Robotern, die pro Jahr 1,2 Millionen Mobiltelefone zerlegen. Vollautomatisierte Demontagelinien kommen aus wirtschaftlichen Gründen nur für Massenprodukte in Frage, die keiner Varianz und geringem Verschleiß ausgesetzt sind [15–17]. Liegt ein Produkt nur in kleinen Stückzahlen bzw. in vielen Varianten vor oder weist großen Verschleiß auf, kommen rein manuelle Demontageprozesse zum Einsatz (wenn die Wirtschaftlichkeit gegeben ist [18]). Diese Wirtschaftlichkeit ist zurzeit nur für hochpreisige Produkte, wie LKW-Getriebe [19], gegeben. Für den variantenreichen mittelpreisigen Produktbereich existieren keine technischen Lösungen in der industriellen Anwendung [12]. Dieser Umstand wird auch anhand des geringen Automatisierungsgrades in diesem Industriebereich sichtbar [16, 18]. Die bestehende Lücke könnte durch RAS geschlossen werden. Allerdings konnten keine Demontagearbeitsplätze dieser Art im industriellen Einsatz ermittelt werden. Betrachtet man den Bereich der industriellen Montage, findet sich eine Vielzahl von hybriden Mensch-Roboter-Arbeitsplatzsystemen [20–24] bei namhaften Unternehmen. Große produzierende Unternehmen, zum Beispiel die Automobilindustrie, können die Kosten für den Einsatz solch innovativer Technologie aufwenden, während die (überwiegend) kleinen oder (seltener) mittelständigen Unternehmen, die sich im Bereich der Refabrikation betätigen, nicht die finanziellen Mittel oder das notwendige Knowhow besitzen. Ein weiterer wichtiger Grund für die schwache Automatisierung im Bereich der Demontage ist die wesentlich höhere Komplexität von Demontageprozessen, wie in Kapitel 4 aufgezeigt wird.

Im Bereich der Forschung konnten folgende Tätigkeiten in der robotergestützten Demontage ermittelt werden.

So erfolgten in [25–27] Untersuchungen zur robotergestützten automatisierten Demontage von elektronischen Altgeräten (am Beispiel von Fernsehern) an Sammelstellen der öffentlichen Abfallbetriebe. Die Automatisierungstechnik konnte sich jedoch der Produktvielfalt und den unterschiedlichen Produktzuständen in keiner Weise anpassen und die Demontage in solchen Sammelstellen wurde als nicht automatisierbar bezeichnet. Die robotergestützte Demontage von elektronischen Komponenten für die Entsorgung von Automobilen war Gegenstand von [28]. Darin fand eine überwiegend zerstörende Demontage von elektronischen Steuerungsgeräten aus Automobilen statt [29]. Im Rahmen dieser Arbeit wurde auch ein Konzept für konfigurierbare robotergestützte Recyclingsysteme entworfen [30]. An einer Lösung zwischen Automatisierung und manueller Arbeit, in Form von Mensch-Roboter-Systeme, wird in [31–35], an der robotergestützten Demontage von Lithiumionenbatterien geforscht. Das Konzept sieht einen hybriden Demontagearbeitsplatz vor, in dem Mensch und Roboter an der Demontage eines Batteriesystems arbeiten. Der Roboter soll Schrauben an Batterien für Elektrofahrzeuge lösen [36]. Hierzu lernt der Mensch die unbekannten Schraubenpositionen eines neuen Batterietyps ein, in dem der Mensch den Roboter an die jeweilige Position von Hand führt. Der Roboter kann nachfolgend die Schraubenpositionen dieses Batterietyps anfahren, das Werkzeug kraftgeregelt auf die Schraube fädeln und sie lösen. Ein Konzept für die ontologiebasierte automatisierte Demontage wurde in [35] vorgestellt. Das Konzept beschreibt einen wissensbasierten Ansatz zur vollautomatisierten Demontage unterschiedlicher Produkte. Der Aufbau des zu demontierenden Produkts wurde hierfür durch eine Ontologie beschrieben. Die vollautomatisierte Demontage von Produkten ist auch Gegenstand des Buches [37]. Es thematisiert das Lebensende von Produkten sowie den Demontageprozess allgemein und geht dann auf die Demontageautomatisierung ein. Dabei werden der Einsatz eines Bildverarbeitungssystems und Demontagewerkzeuge für Roboter beschrieben. Außerdem wird auf die kognitive Robotik Bezug genommen. Eine Fallstudie [38] zur automatisierten Demontage von LCD-Bildschirmen mittels einer Trennscheibe wird am Ende des Buches vorgestellt. Auf die Möglichkeit der Zusammenarbeit von Mensch und Roboter bei der Demontage wird nicht eingegangen.

Im Bereich der Demontage konnte weiterhin keine Forschungstätigkeit nachgewiesen werden, die sich mit dem Einsatz eines intelligenten, mit dem Menschen kollaborierenden

RAS für die zerstörungsfreie Demontage von individuellen Produkten befasst. Hier zeigt sich, dass das Themengebiet der MRK in der Demontage noch nicht vertreten ist. Allerdings sind viele gemeinsame Kernthemen solcher MRK-Systeme Gegenstand aktueller Forschung.

2.2 Stand der Forschung im Bereich der Mensch-Roboter-Systeme

Um die Themenschwerpunkte dieser Forschungsdisziplin aufzuzeigen, werden mehrere aktuelle Forschungsprojekte in diesem Bereich behandelt und deren Ziele aufgezeigt.

Im Projekt ‚Valeri' [39, 40] (‚Validation of Advanced, Collaborative Robotics for Industrial Applications') wurde anhand mehrerer konkreter Applikationen (Auftragen und Inspizieren von Dichtungsmasse sowie Inspektion von geflochtenen Fiberglasmatten) im Bereich der Mensch-Roboter-Interaktion geforscht. Das Projekt verfolgte mehrere Ziele, die auf die Entwicklung des Robotersystems ausgerichtet waren:

- Design eines Robotersystems mit erweitertem Arbeitsraum, siehe Abbildung 1
- zentrale Steuerung der gesamten Kinematik
- Integration neuartiger Sicherheitstechnik für die Mensch-Roboter-Interaktion
- Integration einer flexiblen Bildverarbeitung für Navigation, Bauteilerkennung und Inspektion
- Vereinfachen der Programmierung und Interkonnektivität des Systems

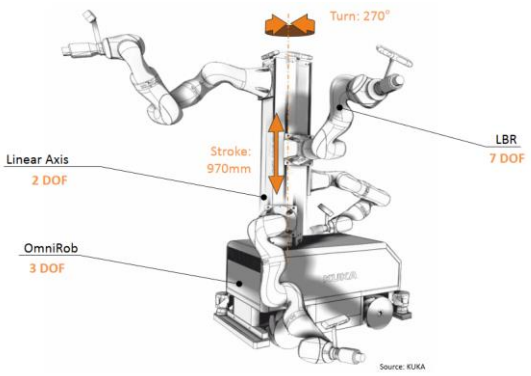

Abbildung 1: Kinematik des mobilen Robotersystems [40].

Gegenstand des ‚HORSE'-Projektes [41] (*smart integrated robotics systems for SMEs controlled by internet of things based on dynamic manufacturing processes*) war die Entwicklung und Anwendung hochflexibler und fast autonomer Robotersysteme für die industrielle Produktion. Konkrete Ergebnisse des Projektes waren:

– Schaffung eines prozessorientierten Management-Modells [42] für die Online-Steuerung von Produktionslinien mit automatisierter Ermittlung und Zuweisung von Ressourcen anhand von festen Prozessablaufmodellen
– Implementierung des Management-Modells unter Adaption des Internet-der-Dinge-Paradigmas und der OSGI-Middleware (siehe [43]) für die Fernsteuerung und Überwachung von Produktionslinien sowie deren Ressourcen – Dabei erfolgte eine nahtlose Integration der Robotersteuerung, der Sensoren und der Mensch-Maschine-Schnittstelle als kontrollierbare, handelnde Agenten in der Produktion.
– Entwurf und Entwicklung von autonomen Systemen, die eine effektive Kooperation zwischen Roboter und Menschen ohne Barrieren ermöglichen und folgende Aspekte inkludieren: multimodale Überwachungs- und Steuerungsmodi neuartiger und kooperativer Roboter für verschiedene industrielle Anwendungen, innovative hybride Positions- und Kraftregelung für inhärent sichere und flexible Robotersysteme und demonstrationsbasierte Roboterprogrammierung für die einfache und intuitive Programmierung durch Leihen
– Entwicklung eines mehrschichtigen Sicherheitsansatzes, der den Roboter sowie das gesamte System miteinbezieht

Das Projekt ‚SYMBOTIC' [44] (‚Symbiotic Human-Robot Collaborative Assembly') sucht Lösungen, um die Anpassungsfähigkeit, Flexibilität und vertikale Integration der industriellen Produktion zu erhöhen. Es stellt sich der damit verbundenen Herausforderung, sichere, dynamische, intuitive und wirtschaftliche Arbeitsumgebungen zu schaffen, in der eine symbiotische Mensch-Roboter-Zusammenarbeit mit großem Nutzen stattfinden könnte, aber durch ihre als zu komplex geltende Aufgabenstellung als nicht automatisierbar angesehen wird. Das Projekt definiert genauer folgende Ziele:

- Entwicklung eines aktiven kollisionsvermeidenden Subsystems, um den menschlichen Arbeiter zu schützen
- Generierung adaptiver Montagearbeitspläne für Mensch und Roboter [45]
- Adaption an dynamische Veränderungen durch intuitive und multimodale Programmierung
- Demonstration und Validierung der Konzepte und Lösungen

Das Projekt ‚CRO-INSPECT' (‚Collaborative Robotic Solution for Advanced Inspection of Complex Composite Parts') [46] soll effiziente und flexible Assistenzroboter für die Inspektion hervorbringen, die die menschliche Adaptionsfähigkeit, die Lage- und Kraftregelung sowie Wiederholgenauigkeit von Robotern verbindet – mit dem Ziel, einen systematischen, reproduzierbaren und durchgängigen Inspektionsprozess für komplexe Bauteile aus Verbundmaterialen zu ermöglichen.

Das EU-geförderte Projekt ‚CogIMon' [47] (‚Cognitive Interaction in Motion') möchte die Mensch-Roboter-Interaktion verändern, indem flexible, nachgiebige, lernende und nutzerfreundliche Roboter entwickelt werden. Die Konzentration liegt auf der Interaktion, die eine aktive und adaptive Regulation der Bewegung und des Verhaltens sowohl des Menschen als auch des Roboters erfordert und eine variable Ganzkörper-Impedanz-Regelung sowie Anpassungsfähigkeit, Prädiktion und Flexibilität beinhaltet. Applikationen, die das Fangen eines geworfenen Balls oder die gemeinsame Handhabung einer Last (Tisch, Karton) in gemischten Mensch-Roboter-Teams zeigten, wurden innerhalb dieses Projektes verwirklicht.

Das Projekt ‚BEYOND SPAI' [48] ist das Nachfolgeprojekt von ‚SPAI' (‚Sichere Personendetektion im Arbeitsbereich von Industrierobotern durch ein aktives NIR-Kamerasystem') und wurde gefördert durch die BMBF-Förderlinie ‚FHprofUnt'. Ziel des Projektes ist es, durch die Nutzung von Ultraschall-Sensoren und Nah-Infrarot-Kamerasystemen

den Menschen im Nahbereich des Roboters sicher zu erkennen und somit dessen Sicherheit bei der Kollaboration mit dem Roboter zu gewährleisten.

Ein BMWi[2]-gefördertes Projekt ist ‚MANUSERV' (‚vom manuellen Prozess zum industriellen Serviceroboter'). Das angestrebte Ziel des Projektes war die Entwicklung *„[…] eines Planungs- und Entscheidungsunterstützungssystems, mit dem ein bisher manuell durchgeführter Prozess mit Hilfe von Servicerobotern technologisch und ökonomisch sinnvoll (teil-)automatisiert werden kann"*[49].

Ein weiteres, im Verbundprojekt ‚Industrie 4.0' vom BMBF[3] gefördertes Projekt ist ‚Hybr-iT' [50] (‚Hybride und intelligente Mensch-Roboter-Kollaboration – Hybride Teams in wandlungsfähigen cyber-physischen Produktionsumgebungen'). Plan des Projekts ist die Erprobung hybrider Teams im industriellen Umfeld mit einer als einmalig bezeichneten ganzheitlichen Betrachtungsweise der MRI-Einzeldisziplinen: intelligente Planungsumgebung, Assistenzsysteme und wissensbasierte Robotik. Ziel ist, dass der Mensch die Zusammenarbeit mit dem Roboter als angenehm, sicher und effizient wahrnimmt. Erforscht und erprobt werden die hierfür notwendigen Technologien, mit Hinblick auf Planung und Optimierung hybrider Teams sowie deren Implementierung in eine IT-Produktionsstruktur.

Das österreichische außeruniversitäre Forschungsunternehmen Profactor forscht intensiv im Bereich der Mensch-Roboter-Interaktion. Das Projekt ‚AssistME' [51] entwickelt und evaluiert innovative Interaktionskonzepte zur Programmierung und Bedienung von robotischen Assistenzsystemen. Projekt ‚KoMoProd' [52] zielt auf die Entwicklung und Evaluierung eines kognitiven Assistenzsystems ab. In diesem Projekt galt es, geeignete Repräsentationen, Methoden und Konzepte zur Parametrierung und Adaptierung des Systems zu finden und zu implementieren. Innerhalb des Projekts ‚FlexRoP' [53] sollen Systeme, die flexibler, einfacher zu programmieren und lernfähig sind, entwickelt werden. Die Ziele im genauen Wortlaut sind [53]:

– Integration einer Assistenzroboterplattform mit erweiterten sensorischen Fähigkeiten

[2]Die Abkürzung steht für das Bundesministerium für Wirtschaft und Energie.

[3]Die Abkürzung steht für das Bundesministerium für Bildung und Forschung.

- Definition einer universellen Darstellung für die Fähigkeit (Skill), eine Montageaufgabe zu bewältigen
- Implementierung automatischer und halbautomatischer Fähigkeiten zur Parametrierung der Skills über visuelle und kinästhetische Beobachtung von Menschen
- Techniken zur Verallgemeinerung, um die in der Anlernphase erworbenen Fähigkeiten in unterschiedlichen Situationen anwenden zu können
- Implementierung von Algorithmen zur Aktionssynthese, um Bearbeitungsprogramme aus Sensordaten ableiten zu können

Im Projekt ‚LERN4MRK: Modellieren, Erlernen und Abstrahieren von Prozessen für die Mensch-Roboter-Kooperation' [54] steht das ‚Transfer Lernen' – von manuell ausgeführten Tätigkeiten auf den Roboter – mittels maschinellen Lernens im Fokus. Die wesentlichen Ziele lauten:

- Mapping der Bewegung des Menschen auf den Roboter
- ‚Verstehen' temporaler Task-Zusammenhänge und Prozessparameter durch den Roboter
- Adaptierbarkeit auf ähnliche Prozesse mit möglichst wenigen neuen Beispielen

Eine weitere führende Forschungsinstitution, die sich mit der Entwicklung innovativer Produktionstechnologien widmet, ist das Zentrum für Mechatronik und Automatisierungstechnik (ZeMA). Im Forschungsprojekt (FourByThree) [55] erfolgte die Entwicklung von sichereren, frei konfigurierbaren, MRK-Robotersystemen und deren Einsatz in verschiedenen MRK-Pilotprojekten. Im Forschungsprojekt (AUTO INB^2) [56] war die Entwicklung von effizienten Montage- und Inbetriebnahmeprozessen in der Automobilproduktion auf Grundlage von MRK-Systemen das Ziel. Im Forschungsprojekt (TRSE) [57] fand die Entwicklung eines teilautomatisierten Mensch-Roboter-Schweißarbeitsplatzes für die Einzelteilfertigung statt. Im Forschungsprojekt ‚Cyber-physische IT-Systeme zur Komplexitätsbeherrschung einer neuen Generation multiadaptiver Fabriken' (SmartF-IT) [58] wurden Softwarewerkzeuge und IT-Infrastrukturen für die Planung und den Betrieb hochflexibler modulbasierter Montagesysteme geschaffen, die die Montage von variantenreichen Produkten ermöglicht. Weiterhin führt das ZeMA den grenzüberschreitenden Forschungscluster für industrielle Robotik und Mensch-Roboter-Kooperation ‚Robotix-Academy' [59].

Die University of British Columbia und weitere namhafte Universitäten erforschten im Projekt ‚CHARM' [60, 61] (‚Collaborative, Human-focused, Assistive Robotics for Manufacturing), wie intelligente Robotersysteme den Menschen in der industriellen Automobil-Produktion unterstützen oder seine Fähigkeiten/Fertigkeiten erweitern können. Hierzu wurden im Bereich der Robotik die Themengebiete Kommunikation, Steuerung und maschinelle Wahrnehmung bearbeitet sowie im Bereich des Systemdesigns die Themen Interaktionsdesign, Situationsbewusstsein und Integration. Ziele des CHARM-Projekts sind (frei übersetzt):

- Identifizieren der Interaktionsanforderungen bestimmter Montageaufgaben im Kontext eines Roboterassistenzszenarios unter Berücksichtigung der Bedürfnisse, Verhaltensweisen und Erwartungen der Fließbandarbeiter
- Entwickeln neuer Interaktionsparadigmen sowie der hierzu benötigten Technologie für Kommunikation und der Mensch-Roboter-Zusammenarbeit, die die identifizierten Interaktionsanforderungen eines menschlichen Mitarbeiters und der Montageaufgabe erfüllen
- Entwickeln von Wahrnehmungs- und Datenintegrationsmethoden, die den Roboterassistenten dazu befähigen, den Arbeiter zu unterstützen oder dessen Handlungsfreiraum erweitert, um seine Aufgabe zu erfüllen
- Entwicklung von neuen Steuerungsmethoden, die für die Integration in sichere und effektive Roboterassistenten geeignet sind, und deren Implementierung in Prototypen
- Erstellung eines integrierten Prototyps für ein gemeinsames Software-Framework, um die technologische Anwendbarkeit anhand einer realen Anwendung zu demonstrieren
- hochqualifiziertes Personal in schnell aufstrebenden Gebieten ausbilden, die für die Entwicklung der kanadischen Hightech-Industrien von entscheidender Bedeutung sind und das internationale Profil Kanadas als weltweit führendes Unternehmen in der Robotik-Technologie verbessern

Weitere, länger zurückliegende Projekte waren PHRIDOM [62] (‘Physical Human-Robot Interaction in Anthropic Domains’) und PHRIENDS [63] (‘Physical Human-Robot Inter-

action: deENDability an Safety'). Schwerpunkte der Forschung waren die Kollisionserkennung mit propriozeptiven Sensoren[4], die Entwicklung von Kollisions-Reaktionsstrategien, die Regelung von Leichtbaurobotern mit Gelenkelastizität sowie die Entwicklung von passiver mechanischer Nachgiebigkeit sowie aktiver Nachgiebigkeit durch Impedanzregelung.

Diese Forschungsprojekte geben eine Übersicht über die Forschungsthemen im Bereich der MRI- und MRK-Systeme, wobei noch keine Spezialisierung auf den Stand der Wissenschaft in den für diese Arbeit wichtigen Spezialgebieten erfolgte – Dies wird im späteren Verlauf dieser Arbeit an den hierzu notwendigen Stellen erfolgen.

2.3 Zusammenfassung

In der Wissenschaft existieren nur wenige Veröffentlichungen hinsichtlich des Einsatzes von MRK-Systemen oder eines RAS für die Demontage von industriellen Produkten. Insbesondere fehlen Erkenntnisse darüber, wie ein RAS den Menschen in der Demontage von variantenreichen oder unterschiedlichen Produkten mit unterschiedlichen Demontage Zielstellungen effizient assistieren kann. Im Bereich der MRK-Systeme und der MRI könnte eine Vielzahl von Veröffentlichungen und internationalen Forschungsprojekten ermittelt werden, meist mit Bezug zur Montage oder Inspektion. In der Regel ist der Beitrag dieser Veröffentlichungen kleinteilig und bezieht sich auf spezialisierte Themen der MRI oder der MRK-Systeme. Nur in manchen Forschungsprojekten konnten ganze Systemarchitekturen entworfen und in Demonstratoren umgesetzt werden. Jedoch zeigte keines dieser Systeme die notwendige technische Autonomie auf, um sich an variable Demontagezielstellungen und variantenreiche Produkte anzupassen.

[4] Propriozeptive Sensoren dienen der Wahrnehmung der eigenen Lage und Bewegung im Raum, bei Industrierobotern sind dies Sensoren zur Erfassung der Achswinkellage, der Achswinkelgeschwindigkeiten und der Antriebsdrehmomente.

3 Einführung in das Themengebiet

Im Rahmen dieses Kapitels erfolgt zuerst die Definition technischer Systeme, um darauf aufbauend das Themengebiet der technischen Assistenzsysteme zu erörtern. In Kapitel 3.2 soll durch die Beschreibung verschiedener Charakteristika ein Verständnis für technische Assistenzsysteme geschaffen werden. In Kapitel 3.3 wird der Fokus auf die Mensch-Roboter-Systeme gelenkt, da die Integration von Industrierobotertechnologie in das Assistenzsystem ein Kernthema dieser Arbeit darstellt.

3.1 Technische Systeme

In dieser Arbeit wird häufig das Wort ‚System' verwendet, zum Beispiel in Form von ‚Assistenzsystem' oder ‚Robotersystem'. Aus diesem Grund soll der Begriff definiert werden. Da der System-Begriff vielfältig verwendet wird und sich daher unterschiedliche Assoziationen gebildet haben, erfolgt die Definition anhand der allgemeinen Systemtheorie [64]. In dieser wird ein System durch drei Aspekte, den strukturellen, den funktionalen und den hierarchischen, charakterisiert. Der strukturelle Aspekt eines Systems beschreibt

> *die Elemente (Teile, Bausteine, Komponenten, Objekte, Artefakte usw.) und dazwischen bestehende Relationen (Beziehungen, Verbindungen, Kopplungen usw.), die zusammen als ‚Ganzes' eine Struktur (Menge, Anordnung, Komplex, Gesamtheit, Gruppe, Ordnung usw.) bilden und die eine oder mehrere Funktionen (Sinn, Zweck, Absicht usw.) erfüllen. [65]*

Die Struktur eines Systems, insbesondere der Elemente und deren Relationen untereinander, kann in einer sogenannten Systemarchitektur [66] dargestellt werden, siehe Abbildung 2. Der funktionale Aspekt eines Systems beschreibt, dessen

> *[...] Funktion […], Stoff (Material), Energie und/oder Information umzuwandeln, zu transportieren und/oder zu speichern [67].*

Dieser Ansatz beschreibt ein System als ‚Black Box', das lediglich anhand seines Ein- und Ausgangsverhaltens charakterisiert wird. Dieser Ansatz ist in der Regelungstechnik weit verbreitet. Der hierarchische Aspekt eines Systems beschreibt den

> *[…] Umstand, dass die Teile eines Systems wiederum als Systeme, das System selbst aber seinerseits als Teil eines umfassenderen Systems angesehen werden können [64].*

Dieses Konzept ist insbesondere bei komplexen Systemen, also Systemen von Systemen, von Bedeutung. Bei einem technischen System handelt es sich letztendlich um

> *[…] industriell (zum Teil auch handwerklich) produzierte, also künstliche, materielle Gebilde (Werkzeuge, Maschinen, Apparate, Bauwerke u. a.) [67].*

Im Zuge dieser Arbeit werden alle drei Systemaspekte des entwickelten RAS detailliert beschrieben. Nun soll jedoch die spezielle Ausprägung des RAS als technisches Assistenzsystem näher erläutert werden.

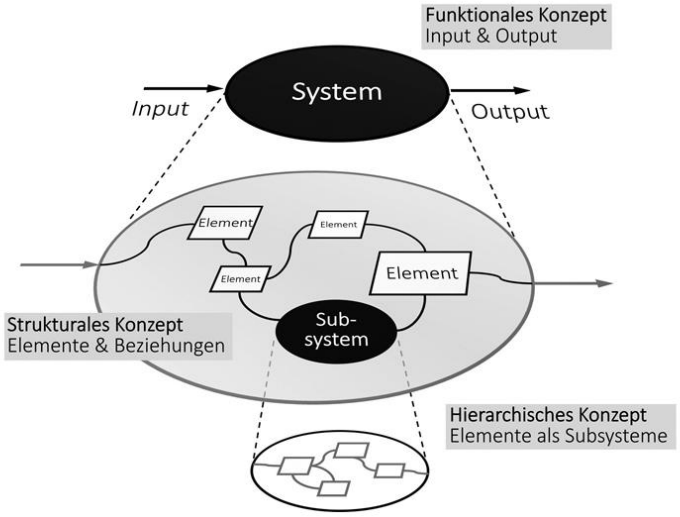

Abbildung 2: Die drei Systemaspekte aus [65].

3.2 Technische Assistenzsysteme

Zur Definition eines technischen Assistenzsystems [10, 68] soll der Begriff ‚Assistenz' zuerst definiert werden. In [69] werden Assistenz und Hilfe als eine spezielle Form der Unterstützung (ausgehend von einem technischen System) angesehen und anhand dreier

Determinanten[5] voneinander abgegrenzt und dadurch definiert [69]. So liegt der Unterschied zwischen Unterstützung und Assistenz darin, dass dem Nutzer bei der Assistenz Zugang zu zusätzlichen Maschinenfunktionen [70] gewährt wird, die die Ausführung der Aktivität erleichtern oder dessen Leistung erhöhen, während durch Unterstützung der Nutzer für die Erfüllung seiner Aufgabe erst befähigt wird. Als Beispiel für Unterstützung und eines technischen Unterstützungssystems kann eine Hebehilfe angesehen werden. Diese ermöglicht es dem Nutzer, Gewichte zu bewegen, die er ohne Unterstützung nicht handhaben könnte. Der Nutzer wird somit für seine Aufgabe durch das Unterstützungssystem befähigt. Bewegt er hingegen Gewichte, die er selbstständig handhaben könnte, kann man die Hebehilfe auch als ein technisches Assistenzsystem begreifen, die die Aktivität des Nutzers erleichtert und dessen Leistung erhöht. Dabei kann das zumutbare Gewicht vom jeweiligen Menschen oder vom Handhabungszeitraum abhängen, was zeigt, dass keine klare Grenze zwischen Assistenz und Unterstützung besteht und dass die Entscheidung, ob Assistenz oder Unterstützung stattfindet, mit der Aktivität, der Situation sowie dem Kontext zusammenhängt und letztlich durch einen Beobachter getroffen wird [69]. Durch den fließenden Übergang zwischen Assistenz, Hilfe und Unterstützung und die Abhängigkeit von einem Beobachter kann der Begriff der Assistenz nicht allgemeingültig getroffen werden [69, 71, 72]. Daher wird auf eine Definition eines technischen Assistenzsystems verzichtet. Stattdessen werden zwei generische Definitionen eines Unterstützungssystems vorgestellt:

> *Bei einem technischen System handelt es sich um ein Unterstützungssystem, wenn es den Menschen bei Tätigkeiten unterstützt, ohne ihn ganz oder teilweise zu substituieren, es dem Menschen die Hoheit über die Ausführung überlässt, es sich bei dem Menschen um den Systembediener handelt und vom System keine Gefahr für den Bediener und für Dritte ausgeht. [69]*

> *Ein Unterstützungssystem ist ein informationsverarbeitendes technisches Gebilde, das die Aufgabenerfüllung eines Operateurs in einem Mensch-Maschine-System dadurch fördert, dass es bestimmte, für seine Zielerreichung notwendige, Teilauf-*

[5] Diese Determinanten betrachten das Dual „menschliche Aktivität-technische Unterstützung“ anhand der räumliche-zeitlichen Beziehung, der Form der Kopplung von Aktivität und Unterstützung sowie der Verortung von Kontrolle.

> *gaben innerhalb seiner Gesamtaufgabe übernimmt und/oder ausführt. Es ist damit als Oberbegriff für rechnergestützte Technologien wie Assistenzsystem, Hilfssystem o. ä. zu verstehen. [73]*

Zur Vollständigkeit soll noch der Begriff ‚Hilfe' definiert werden. Bei der Hilfe übernimmt das technische System vorübergehend die Kontrolle über eine Aktivität, die der Mensch, zum jeweiligen Zeitpunkt, nicht selbstständig ausführen kann [69]. Die Begriffsdefinition von ‚Assistenz', ‚Unterstützung' und ‚Hilfe' ist wichtig, allerdings bleibt das Problem bestehen, verschiedene Eigenschaften, Charakteristika oder Arten von Assistenzsystemen zu beschreiben oder zumindest voneinander abzugrenzen. Eine genauere Beschreibung eines Assistenzsystems erfolgt in der Literatur [10, 74] oft durch vorangestellte Adjektive wie ‚interaktiv', ‚kognitiv' oder ‚intelligent' oder mit dem Verb des verbundenen Zwecks, z. B. ‚Bremsassistent'. Denn erst diese genauere Beschreibung kann Aufschluss über den Einsatzbereich, die Form der Assistenz und ggf. die Komplexität des Systems geben [10]. Diese Unterscheidung ist wichtig, um zwischen einem einfachen technischen Hilfsmittel, wie einem Akkuschrauber, und einem komplexen Assistenzsystem, z. B. dem in dieser Arbeit entwickelten RAS, zu unterscheiden. Für die Differenzierung unterschiedlicher Assistenzsysteme kann ein Klassifizierungsschema, etwa in Form einer Taxonomie, verwendet werden. Eine Taxonomie definiert anhand bestimmter Kriterien verschiedene Klassen (z. B. von Assistenzsystemen) und ordnet diese in einer hierarchischen Struktur an. In der Literatur existieren mehrere Taxonomien für die Klassifikation von Assistenzsystemen. Sie unterscheiden sich durch die Wahl der Klassifizierungskriterien oder in ihrem Detailierungsgrad. Eine Auswahl von Taxonomien für Assistenzsysteme und den jeweiligen Klassifizierungskriterien wird nachfolgend genannt und erläutert.

Ein Kriterium für die Klassifizierung von Assistenzsystemen stützt sich auf unterschiedliche Automatisierungsgrade, die zwischen einem rein manuell gesteuerten und einem vollständig automatisierten Prozess liegen können. Der Automatisierungsgrad wird in der gesichteten Literatur in drei bis zehn Stufen unterteilt [75–79]. In [75] wird der Automatisierungsgrad eines Systems hinsichtlich der folgenden vier Funktionen betrachtet:

1. Überwachung des Herstellungsprozesses (Monitoring)
2. Generieren von Wahlmöglichkeiten

oder Strategien zur Zielerfüllung (Generating)

3. Auswahl einer Möglichkeit bzw. Strategie (Selecting)
4. Durchführung bzw. Umsetzung der getroffenen Wahl (Implementing)

Je nach Zuordnung des Menschen (H) oder von Maschine/Computer (C) zu den vier Funktionen ergeben sich zehn unterschiedliche Automatisierungsgrade, siehe Tabelle 1.

In [74] erfolgt die Klassifikation von Assistenzsystemen über den Grad der Autonomie, mit der vom System Assistenz ausgeht. Wobei ein System als autonom gilt,

> *„[…] wenn es ohne menschliche Steuerung oder detaillierte Programmierung ein vorgegebenes Ziel selbstständig und an die Situation angepasst erreichen kann. Das System ist fähig, die Umgebung über Sensoren wahrzunehmen, proaktiv und situationsgerecht einen angemessenen Handlungsplan zu generieren und über Aktoren sicher und zuverlässig auszuführen“ [80]*.

Tabelle 1: Die zehn Automatisierungsgrade nach [75]

Level of Control	**Monitoring**	**Generating**	**Selecting**	**Implementing**
Manual Control	H	H	H	H
Action-Support	H/C	H	H	H/C
Batch-Processing	H/C	H	H	C
Shared Control	H/C	H/C	H	H/C
Decision-Support	H/C	H/C	H	C
Blended-Decision-Making	H/C	H/C	H/C	C
Rigid System	H/C	C	H	C
Automated-Decision-Making	H/C	H/C	C	C
Supervisory-Control	H/C	C	C	C
Full Automation	C	C	C	C

Der Autonomiegrad eines Assistenzsystems wurde nach [74] hierzu in drei Formen differenziert:

– Die erste Klasse: Das Assistenzsystem bietet lediglich eine oder mehrere sehr einfache Assistenzfunktionen, die automatisch oder durch Aktivierung des Nutzers ausgelöst werden.

– Die zweite Klasse: Dies sind Assistenzsysteme, die eine Gruppe von Assistenzfunktionen anbieten und dem Nutzer bei der Durchführung eines bekannten Anwendungsfalls assistieren. Die Komplexität der Assistenzsysteme in dieser Klasse kann durch die Komplexität des Anwendungsfalls variieren.
– Die dritte Klasse: Diese Klasse bilden Assistenzsysteme, die aus dem Verhalten des Nutzers auf die Aufgabe oder das Ziel seiner Handlungen schließen lassen. Nachfolgend ermittelt das Assistenzsystem geeignete Schritte zur Problemlösung, schlägt diese dem Nutzer vor und führt sie ggf. aus.

Ein weiteres Klassifizierungskriterium basiert auf der Beschreibung, welche menschlichen Funktionen durch ein Assistenzsystem unterstützt werden können. Menschliche Funktionen sind zum Beispiel die menschliche Wahrnehmung, Unterstützung von sensorisch-motorischen Funktionen oder die Unterstützung bei der Entscheidungsfindung. In [73] werden der Klassifizierung der jeweilig unterstützten menschlichen Funktion zusätzlich das Anwendungsfeld und die für die Umsetzung genutzte Technologie hinzugefügt.

Aufbauend auf den Erkenntnissen der Kognitionswissenschaften, die eine menschliche Handlung in verschiedene Phasen gliedert, besteht ein weiteres Klassifizierungskriterium [70]. Die einzelnen Phasen einer Handlung werden als die sechs Aspekte der kognitiven Ergonomie bezeichnet:

- 1. Motivation, Aktivierung und Zielsetzung: Der Nutzer soll in der Zielsetzung unterstützt und motiviert werden, erste Schritte selbst einzuleiten. Das System soll auch im weiteren Arbeitsablauf den Menschen ‚antreiben' bzw. ‚bremsen', zum Beispiel bei sicherheitstechnischen Aspekten.

- 2. Wahrnehmung: Der entstehende Handlungsablauf in der Zusammenarbeit zwischen Mensch und Maschine basiert auf dem Ziel und dem Motiv des Menschen und der vom Assistenzsystem angebotenen Unterstützung zum jeweiligen Zeitpunkt. Damit der Mensch die technische Umgebung erfassen kann, muss ihm der Zustand dieser bekannt sein. Die Darstellung des Zustandes erfolgt über Signale, die vom Assistenten verstärkt, gespeichert oder in ein anderes Medium umgewandelt werden können.

- 3. Informationsintegration und Situationsbewusstsein: Oft können Informationen dahingehend ausgewertet werden, dem Nutzer ein besseres Bild über die

Situation zu verschaffen. Neben einfachen selbsterklärenden Symbolen zur Beschreibung der Situationen können Assistenzsysteme weitere erklärende Informationen dem Nutzer zur Verfügung stellen.

- 4. Entscheidungsunterstützung zur Auswahl der Assistenz: Hier gibt es grundsätzlich die Möglichkeit, dem Nutzer alle Assistenzsystemfunktionen direkt anzubieten. Da ihn dies wahrscheinlich überfordert, kann die Auswahl der angebotenen Funktionen mithilfe von Filtern reduziert werden. Durch diese werden dem Nutzer nur einige Funktionen repräsentiert. Die Filterung kann anhand unterschiedlicher Möglichkeiten realisiert werden, zum Beispiel nach Nutzenfunktionen, Nutzervorlieben oder Wahrscheinlichkeiten. Wenn das System als Berater agiert, kann es dem Benutzer eine Funktion anbieten. Muss der Nutzer diese ohne Auswahlmöglichkeit akzeptieren, spricht man von einer Anweisung. Bei der Beraterfunktion besteht jedoch die Gefahr, dass der Nutzer sich bevormundet fühlt und die Assistenz kategorisch ablehnt. Neben der Entscheidungsunterstützung kann das System auch die Ausführung der Assistenz übernehmen. Hierzu werden drei verschiedene Stufen beschrieben. Die erste wird als Delegation bezeichnet. Der Assistent bietet dem Nutzer hierzu eine Funktion an. Bestätigt dieser den Vorschlag, führt das System die Assistenz automatisiert aus. Lehnt er den Vorschlag ab, bietet das System eine andere Funktion an oder wartet auf eine Situationsveränderung. Die zweite Stufe übernimmt die Kontrolle über den Prozess. Das System führt seine Funktion aus, solange der Mensch nicht in einem definierten Zeitrahmen widerspricht. Die zuletzt beschriebene Assistenz übernimmt die Kontrolle über den gesamten Prozess; der Mensch kann nicht mehr eingreifen, sondern lediglich die Assistenz deaktivieren.

- 5. Ausführung der Assistenz: Man muss grundsätzlich unterscheiden, wie Assistenz ausgeführt werden kann. Sie kann durch Übernahme der Handlung erfolgen, oder sie kann bei der Durchführung der Handlung unterstützen. Meist erfolgt Letzteres, um den Menschen in kraftintensiven oder sensitiven Handlungen zu unterstützen oder um zeitintensive Handlungen zu verkürzen.

- 6. Bewertung des Erfolgs: Das System und der Nutzer müssen nach einer Handlung die Effekte dieser verarbeiten. Hierzu muss die Fähigkeit der Diagnose vorhanden sein. Ist der gewünschte Effekt nicht eingetreten, kann die Fähigkeit der Korrektur von Nutzen sein. Gibt es keine Möglichkeit zur Korrektur, muss die Option der Reflexion gegeben sein, die eine Änderung der Zielsetzung ermöglicht.

Der Vorteil dieses Ansatzes besteht darin, dass die Phasen unabhängig von jeder Aufgabe, Handlung oder Aktivität sind. Durch diese Aspekte kann nicht nur ein Assistenzsystem klassifiziert, sondern auch eine menschliche Handlung untersucht werden. Betrachtet man eine konkrete Aufgabe anhand der einzelnen Aspekte, kann für jede Handlungsphase eine angepasste Unterstützung entworfen werden.

Alle diese Klassifizierungsschemata leiden jedoch am generischen Konzept der Assistenz, die so vielfältige Ausprägungen annehmen, dass kaum auf spezielle Charakteristika der technischen Systeme Bezug genommen werden kann. Erst durch eine weitere technologische Spezialisierung des Assistenzsystems kann auf technische Details eingegangen werden. Hierzu soll jedoch zuerst das Kernmerkmal eines Assistenzsystems herausgestellt werden, die intensive Mensch-Maschine-Interaktion. Diese erfordert, dass eine Betrachtung von Mensch und Maschine als ein Gesamtsystem stattfindet [81, 82]. Diese Betrachtung des Mensch-Maschine-Systems beschreibt

> *[…] eine zweckmäßige Abstraktion des zielgerichteten Zusammenwirkens von Personen mit technischen Systemen zur Erfüllung eines fremd- oder selbstgestellten Auftrages [73].*

Unter dieser Betrachtung wurde auch das RAS entwickelt. Da der Schwerpunkt im Rahmen dieser Arbeit auf der Integration von Industrierobotertechnologie liegt, kann auf eine speziellere technologische Ausprägung einer Maschine, d. h. eines Industrieroboters, und damit auf die Betrachtung eines Mensch-Roboter-Systems Bezug genommen werden [10]. In der gesichteten Fachliteratur wird der Begriff ‚Mensch-Roboter-System' jedoch nicht verwendet, sondern die Bezeichnung ‚MRK-System'.

3.3 Mensch-Roboter-Systeme

Bezüglich der Abkürzung ‚MRK' gibt es verschiedene Interpretationen, wofür der Buchstabe ‚K' steht. Dieser könnte ‚Kollaboration', ‚Kooperation', ‚Kollaboration und Kooperation' oder alle drei Formen der Mensch-Roboter-Interaktion (inklusive der Ko-Existenz) bedeuten. In dieser Arbeit soll die letzte Auffassung gelten. Die Mensch-Roboter-Interaktion (engl.: Human-Robot Interaction) ist Kernthematik der Betrachtung des MRK-Systems und stellt eine eigenständige Forschungsdisziplin dar, deren Gegenstand es ist, die Interaktionen zwischen einem oder mehreren Menschen und einem oder mehreren Robotern zu verstehen und zu gestalten [83]. Die folgenden Unterkapitel stellen eine Einführung in das Themengebiet der MRI und der MRK-Systeme dar. Insbesondere soll der Einfluss von verschiedenen Charakteristika eines MRK-Systems auf die MRI gezeigt werden. Denn für die Entwicklung eines RAS muss bekannt sein, welche Charakteristika eines Systems Einfluss – und in welcher Form – auf die MRI ausüben.

3.3.1 Formen der Mensch-Roboter-Interaktion

Allgemein beschreibt der Begriff ‚Interaktion' das wechselseitige Einwirken von Akteuren oder Systemen aufeinander. Im Fall der MRI ist dies demnach das wechselseitige Einwirken des Menschen auf den Roboter und umgekehrt. Innerhalb der MRI werden die bereits genannten Formen der Interaktion, die Kollaboration, Kooperation und Ko-Existenz, unterschieden. Diese sollen nun ausführlich beschrieben werden.

Die Kooperation, oft auch als ‚einfache Kollaboration' bezeichnet, beschreibt ein Konzept, in dem Akteure einem gemeinsamen übergeordneten Ziel dienen [84]. Um das Ziel zu erreichen, werden die Arbeitsinhalte auf einzelne Akteure verteilt und unabhängig voneinander bearbeitet [85]. Durch Zusammenführen der Ergebnisse wird das übergeordnete Ziel letztendlich erfüllt. Mit Bezug zur Mensch-Roboter-Kooperation bedeutet dies eine Verteilung[6] einzelner Arbeitsinhalte auf den Roboter oder Menschen. Die Bestimmung der Arbeitsinhalte sowie deren Verteilung erfolgt i. d. R. einmalig beim Entwurf

[6] Die Aufteilung von Arbeitsinhalten auf Akteure wird auch als Aufgabenallokation (engl. *task allocation*) bezeichnet.

des MRK-Systems und orientiert sich meist an den jeweiligen Fähigkeiten und Fertigkeiten sowie an der Wirtschaftlichkeit der Gesamtlösung [86]. Der Ansatz der Mensch-Roboter-Kooperation findet im industriellen Umfeld, z. B. in Form von hybriden Montagesystemen, bereits breite Anwendung (siehe Abbildung 3). Ein Unterschied zur klassischen robotergestützten Automatisierung besteht darin, die Tätigkeit des Menschen nicht (vollständig) zu ersetzen, sondern diesen durch Übernahme von Teiltätigkeiten zu entlasten. Ein weiteres Unterscheidungsmerkmal ist in Abbildung 3 zu erkennen: Die Arbeitsräume von Mensch und Roboter überschneiden sich. Der gemeinsam genutzte Arbeitsraum wird auch als ‚Kollaborationsraum' bezeichnet.

Abbildung 3: Hybrides Montagesystem für Achsgetriebe [87].

Die Kollaboration beschreibt das zielgerichtete Zusammenwirken von Akteuren für ein gemeinsames Ziel [84]. Im Gegensatz zur Kooperation sind die Aufgaben in der Kollaboration voneinander abhängig und werden gemeinsam bearbeitet. Bei der Kollaboration kann direkter physischer Kontakt zwischen Menschen und Robotern bestehen. Wichtig bei dieser Form der Zusammenarbeit ist, dass die Handlungen von Mensch und Roboter koordiniert werden. Prozesse mit kollaborierenden Akteuren sind, gegenüber Prozessen mit kooperierenden Akteuren, wesentlich komplexer hinsichtlich der Planung und Koordination von Handlungen sowie der hierzu notwendigen Kommunikation.

Dass Mensch und Roboter keine gemeinsame Zielstellung verfolgen, ist das Merkmal der Ko-Existenz. Mensch und Roboter agieren unabhängig voneinander, um ihren eigenen

Auftrag zu erfüllen. Lokal und vorübergehend kann es zu einer Interaktion kommen, zum Beispiel, wenn eine Person in den Fahrweg eines mobilen Roboters tritt und der Roboter stoppen muss. Diese Art der Interaktion wird auch als Obstruktion bezeichnet. Sie tritt auf, wenn Akteure zur Erfüllung ihrer Aufgabe in Konkurrenz bzgl. Ressourcen treten.

Kernelement eines MRK-Systems ist das Robotersystem, dessen zentrale Eigenschaften im Folgenden beschrieben werden.

3.3.2 Charakteristik des Robotersystems

Die Charakteristiken eines Robotersystems sollen auf Grundlage von [84] in diesem Kapitel aufgezeigt werden. In [84] werden die folgenden Klassifizierungskriterien definiert: Einsatzgebiet, Aufgabe(n), Gestalt und Autonomie des Robotersystems.

Die in der Klassifizierung vorgegebenen Einsatzorte werden in industrielle, kommerzielle und persönliche Bereiche getrennt. Diese Unterteilung ist notwendig, da jede Umgebung andere Anforderungen an das Robotersystem stellt. Zum Beispiel sind die Anforderungen im industriellen Umfeld hinsichtlich der Zuverlässigkeit wesentlich höher als im privaten Bereich. Diese grobe Einteilung in drei Einsatzorte kann beliebig weiter detailliert werden, wie für den industriellen Bereich in Produktion, Intralogistik oder Service. Der Einsatzort des in dieser Arbeit entwickelten Assistenzsystems liegt im Bereich industrieller Demontageprozesse in der Wartung, Instandsetzung oder Refabrikation von Produkten. Der Einsatzort des RAS und die durch die Umgebung gestellten Anforderungen an das Assistenz- und Robotersystem werden hierzu in Kapitel 4 gesondert dargestellt.

Die Aufgaben des Roboters werden in [84] unterteilt in Informationsaustausch, Präzision, Entlastung, Transport und Manipulation. Diese allgemeine Einordnung lässt jedoch wenige oder keine Schlüsse über die konkreten Aufgaben eines Robotersystems in seinem Einsatzbereich zu. Eine detaillierte Beschreibung der Aufgabe(n) ist essenziell, da sie konkrete Anforderungen an das Robotersystem stellen. Diese Anforderungen bestimmen wiederum technische Ausprägungen des Systems und sind für dessen Entwurf und Umsetzung von Bedeutung. Die Aufgaben, die dem in dieser Arbeit entwickelten RAS zugeordnet wurden, werden aus diesem Grund in Kapitel 6.3 sowie in Kapitel 7.3 gesondert und detailliert beschrieben.

Die Gestalt (Morphologie) eines Roboters wird in den Literaturquellen [83, 88–90] in drei Klassen unterteilt: humanoid, zoomorph und funktional. Humanoide oder zoomorphe Roboter sind der Gestalt des Menschen bzw. der von Tieren nachempfunden. Für Industrieroboter deren Gestalt an ihre Funktion angepasst ist, steht die funktionale Gestalt. Die Gestalt des Roboters spielt in der MRI eine besondere Rolle, die zum Beispiel am Phänomen des unheimlichen Tals (engl.: *uncanny valley*) deutlich wird [88, 91]. Die Hypothese beschreibt, dass die menschliche Akzeptanz nicht linear mit der Annäherung eines Roboters zur menschlichen Gestalt zunimmt. Vielmehr zeigt sich, dass ein Tal von geringer Akzeptanz bei zunehmender Annäherung an die menschliche Gestalt [88] durchschritten werden muss, wie in Abbildung 4 ersichtlich ist.

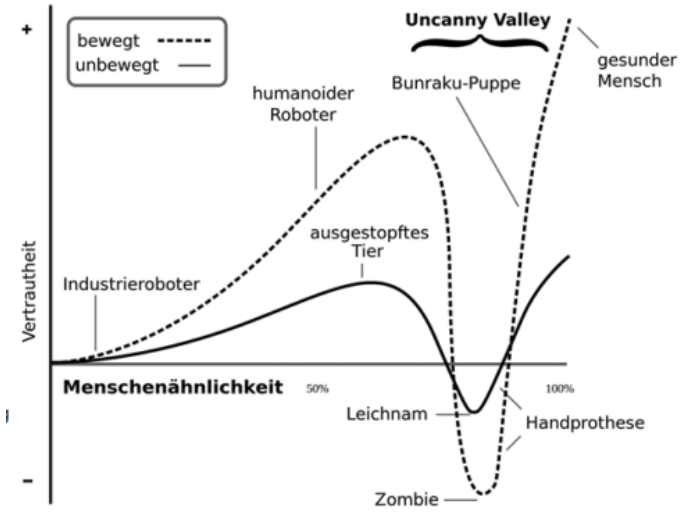

Abbildung 4: Phänomen des unheimlichen Tals [91].

Grund sind kleine Imperfektionen, zum Beispiel in der Mimik, die vom Menschen als verstörend empfunden werden und die Akzeptanz sinken lassen. Menschen verbinden mit der Gestalt eines Roboters auch bestimmte Erwartungen und passen entsprechend ihrer Erwartung ihre Interaktionen an. So erwartet ein Mensch, dass ein Roboter mit Gesicht, Mund und Ohren sprechen, sehen und hören kann. Für den industriellen Einsatzbereich muss die Beschreibung der funktionalen Gestalt wesentlich detaillierter erfolgen. Industrieroboter werden i. d. R. anhand ihrer Kinematik, des Arbeitsraums und der Nutzlast klassifiziert. Auch Klassifizierungen hinsichtlich Aufgabe, Einsatzgebiet sowie Branche

sind üblich. Eine besondere Klasse von Industrierobotern stellen MRK-fähige Robotersysteme dar, da in diesem Bereich höhere Anforderungen bzgl. der Sicherheit bestehen. Industrierobotersysteme unterscheiden sich weiter anhand der verwendeten Antriebssysteme, Steuerung sowie Programmierung. Die Auswahl eines Industrieroboters muss aus diesem Grund immer anwendungspezifisch getroffen werden.

Das letzte und wohl entscheidendste Einteilungskriterium stellt die Autonomie des Robotersystems dar. Bei der Autonomie eines Systems muss zwischen zwei Arten unterschieden werden: Autonomie im Sinne der klassischen Automatisierung, die ein System dazu befähigt, eine Aufgabe selbstständig zu lösen, oder Autonomie im Sinne von Kooperation oder Kollaboration, die ein System dazu befähigt, mit anderen (Systemen oder Menschen) zusammen eine Aufgabe zu lösen [78]. Die letztere Form der Autonomie wird auch als Teamautonomie bezeichnet. Diese Form der Autonomie beeinflusst zwei wesentliche Aspekte der MRI, die Frequenz der Interaktion und die Art der Interaktion . In der MRI-Forschung wird einerseits davon ausgegangen, dass eine höhere Autonomie dazu führt, dass eine Interaktion seltener stattfinden muss. Andererseits wird angenommen, dass sie eine höhere Form der MRI ermöglicht. Hierzu kann folgendes Beispiel angeführt werden. Ein Operateur steuert einen teleoperierten Roboter, indem er die Endlage des Effektors vorgibt und den Greifer öffnet und schließt mit dem Ziel, ein Werkstück von A nach B zu transportieren. Die Frequenz der Interaktion ist durch die direkte Steuerung hoch und die Art der Interaktion ist primitiv. Beides zeigt, dass das System über eine geringe Autonomie verfügt. Ein Robotersystem mit höherer Autonomie könnte zum Beispiel vom Nutzer die Aufgabe erhalten, das Werkstück nach B zu transportieren, und diese Aufgabe selbstständig ausführen. An diesem Beispiel zeigt sich klar, dass die Interaktion auf einer höheren Form stattfindet und dass die Frequenz der Interaktion, bis auf die Erteilung des Auftrags, sich reduziert. Um die Aufgabe selbstständig auszuführen, muss das System jedoch über eine höhere Autonomie verfügen. So muss es etwa wissen, oder sensorisch erfassen, wo das Werkstück liegt. Es muss planen, wie es das Werkstück von A nach B transportiert und wie es einzelne Handlungen ausführt. Eine zu diesem Beispiel passende Definition der Autonomie ist:

> *the extent to which a robot can **sense** its environment, **plan** based on that environment, and **act** upon that environment with the intent of reaching some **task-specific goal** (either given or created by the robot) without external control [78].*

Diese Definition der Autonomie ist passend, da sie auf das sogenannte Wahrnehmen-Planen-Handeln-Paradigma (engl.: *sense-plan-act paradigm*) Bezug nimmt. Dieses Konzept beschreibt eine von mehreren Möglichkeiten zur Umsetzung von Autonomie in einem technischen System und wird im Kapitel 5.1.1 näher beschrieben. Auf Basis eines ähnlichen Konzepts wurde auch die Autonomie des RAS in dieser Arbeit umgesetzt. Wie die einzelnen wesentlichen Aspekte (Wissen, Wahrnehmung, Planen und Handeln) der Autonomie im Rahmen dieser Arbeit umgesetzt wurden, wird in Kapitel 6.3 detailliert erläutert. Zwei weitere wichtige Aspekte, die obiger Definition entnommen werden können, sind, dass technische Autonomie nur für eine klar bestimmte Aufgabe definiert werden kann und dass eine explizite Angabe eines Ziels benötigt wird.

In Bezug zum vorherigen Beispiel wurde nicht genau beschrieben, wie der Mensch dem Robotersystem die Aufgabe erteilt. Zum Beispiel hätte der Mensch über ein Terminal ein Befehl eingegeben, eine Kombination von Knöpfen oder Reglern betätigen oder den Befehl per Sprache erteilen können. Wie die Kommunikation zwischen Mensch und Roboter erfolgt, besitzt somit ebenfalls einen großen Einfluss auf die MRI und stellt deshalb ein weiteres entscheidendes Klassifizierungskriterium für MRK-Systeme dar.

3.3.3 Mensch-Roboter-Kommunikation

Generell handelt es sich bei der Kommunikation um eine Sonderform der Interaktion, die sich auf den Austausch von Informationen zwischen Mitgliedern einer Gruppe begrenzt. Durch Kommunikation kann zwischen Gruppenmitgliedern Aufmerksamkeit (im Sinne eines situationsabhängigen Bewusstseins (engl.: *situation awareness*) hergestellt, Absichten (Intensionen) können geteilt und Informationen zur Situation ausgetauscht werden. Der Informationsaustausch kann dabei unidirektional, etwa vom Roboter zum Menschen, oder bidirektional, d. h. in beide Richtungen, stattfinden. Um die Kommunikationsbarriere zwischen Mensch und Maschine zu überbrücken, werden sogenannte Mensch-Maschine-Schnittstellen (MMS) (engl.: *human machine interfaces*) zwischengeschaltet. Heutige MMS bieten in der Regel eine grafische Benutzeroberfläche, die mit Maus und Tastatur oder direkt über Touch-Bildschirme durch den Menschen bedient werden. Moderne MMS können aber auch natürliche Sprache, Gesten, Emotionen oder die Blickrichtung des Menschen erkennen und diese Modalitäten für die Kommunikation verwenden. Weiterhin bietet der Einsatz von Geräten wie der Mircosoft Hololens, die dem

Nutzer eine computergestützte Erweiterung der Realitätswahrnehmung (engl.: *augmented reality*) ermöglichen, hohes Potenzial in der Mensch-Roboter-Kommunikation [92]. Robotersysteme, die mit Kraft- oder Drehmomentsensoren ausgerüstet oder mit sensitiven Häuten ummantelt sind, können dem Menschen weiterhin eine haptische Schnittstelle über ihre Robotermechanik bieten. Diese Art der MMS wird auch als ‚anfassbare Benutzerschnittstelle' (engl.: tangible user interfaces) bezeichnet. Die Kombination von verschiedenen MMS mit unterschiedlichen Modalitäten ermöglicht die multimodale Kommunikation, bei der Informationen redundant über verschiedene Modalitäten parallel übermittelt werden, mit dem Ziel, die Informationsübertragung robuster zu gestalten. In der Robotik und in anderen Feldern wird zurzeit intensiv daran gearbeitet, die Mensch-Roboter-Kommunikation robuster und natürlicher zu gestalten. Dabei ist das Design der MMS entscheidend: Nicht intuitiv gestaltete Mensch-Maschine-Schnittstellen oder ungünstige Modalitäten erschweren die Kommunikation oder verhindern sogar eine zielgerichtete Interaktion [90, 93–95].

In dem entwickelten RAS wurden verschiedenen Mensch-Roboter bzw. Mensch-Computer-Benutzerschnittstellen erstellt und getestet. Darunter sind mehrere grafische Benutzeroberflächen, die mit Maus und Tastatur oder über ein Touch-Screen die Bedienung unterschiedlicher technischer Systeme sowie des Roboters ermöglichen. Die Kommunikation mittels natürlicher Sprache konnte durch die Integration eines Sprachassistenten [96] in das RAS realisiert werden. Auch eine Applikation für die Teleoperation eines Roboters durch Gestensteuerung war Gegenstand der Untersuchung. Für die Synchronisierung der Tätigkeiten von Mensch und Roboter wurde außerdem eine haptische MRS über die Robotermechanik umgesetzt. Die entwickelten MRS sind nicht Kernthematik dieser Arbeit. Sie waren im Rahmen dieser Arbeit notwendig, um die Handlungen von Mensch und RAS zu koordinieren, denn ohne Kommunikation kann auch keine Interaktion stattfinden – insbesondere, wenn mehrere Roboter und Menschen zusammenarbeiten.

3.3.4 Individuen, Gruppen und Teams in MRK-Systemen

Ein MRK-System besteht zumindest aus einem Menschen und einem Roboter, aber es können auch andere Gruppierungen in einem MRK-System existieren. Die Anzahl von Menschen und Robotern und das dadurch bestimmte Verhältnis ist damit ein wichtiges

Kriterium zur Beschreibung eines MRK-Systems. Von Bedeutung ist weiter, ob die Mitglieder der Gruppen eigene Ziele verfolgen (im Sinne von Ko-Existenz) oder heterogene oder homogene Teams bilden, die ein gemeinsames Ziel verfolgen (im Sinne von Kooperation oder Kollaboration). Mensch-Roboter-Teams sind per Definition

> *[…] groupings of humans and robotic systems who communicate, coordinate and collaborate together to perform a joint activity [90].*

Je nach Zusammensetzung der Gruppe können nun unterschiedliche Interaktionen stattfinden. So können einzelne oder mehrere Menschen und Roboter miteinander oder mit Teams interagieren, wobei auch Interaktionen von Teams möglich sind. In Abbildung 5 ist eine Auswahl von möglichen MRK-Gruppenzusammensetzungen dargestellt; darin werden die Interaktionen durch Linien, einzelne Menschen (M) oder Roboter (R) als Kreise und Teams durch ein umschließendes Rechteck dargestellt. In diesen Gruppen können komplementäre, konkurrierende oder indifferente Zielstellungen entstehen, die die Interaktion positiv oder negativ beeinflussen. In diesem Kontext ist auch von Interesse, ob Teams statisch definiert werden oder ob sie sich dynamisch durch einen Mechanismus bilden (engl. *teaming*). Zum Beispiel könnte sich ein Team anhand einer Aufgabenstellung strukturieren. Weiterhin muss berücksichtigt werden, dass Teams aus einzelnen Mitgliedern bestehen, die anhand ihrer speziellen Fachkenntnisse (im Sinne von Wissen), Fähigkeiten und Fertigkeiten sowie Ressourcen unterschieden werden müssen. Diese Charakteristik eines Mitglieds bestimmt i. d. R., welche Rollen es im Team übernimmt, wobei die zugewiesene Rolle sich auch mit der Zeit ändern kann. In Teams kann es dabei vorkommen, dass mehrere Mitglieder die gleichen (komplementären) Fertigkeiten besitzen. Dies kann etwa bei der Aufgabenallokation, in der Aufgaben entsprechend der Fähigkeiten verteilt werden, eine Konfliktlösungsstrategie erfordern, die aus mehreren möglichen Kandidaten einen Kandidaten für die Aufgabenübernahme bestimmt. Ein weiterer wichtiger Aspekt, der aus der Teamdefinition entnommen werden kann, ist die Koordination der Teammitglieder. Koordination ist die Voraussetzung für das harmonische Funktionieren einer Gruppe bei der Ausführung ihrer Aufgaben und erfordert Voraussicht, Planung, Organisation und Überwachung der Aktivität. Für eine wirksame Koordination [90] ist erforderlich, dass:

– Ein gemeinsamer (abstrakter) Wissensraum besteht (engl.: *common ground*) [97].
– Ein gemeinsames Ziel definiert wurde.

– Kenntnis über die Zusammensetzung des Teams und der Fähigkeiten oder Fertigkeiten der einzelnen Teammitglieder besteht.
– Das eine Planung der Handlungen der einzelnen Teammitglieder erfolgt.
– Eine ausreichende Kommunikation stattfindet.
– Eine Überwachung, Bewertung und ggf. Korrektur der Handlungen erfolgt.

Auf die Aspekte der Koordination wird in der Softwarearchitektur des RAS in Kapitel 6.3.2 eingegangen. In Kapitel 6.3.2 wird zum Beispiel beschrieben, wie die Teamzusammensetzung, die Charakteristik der Teammitglieder sowie die notwendigen Ressourcen im einem aufgabenspezifischen Prozessmodell repräsentiert werden, wie auf Basis dieses Prozessmodells die Planung der einzelnen Handlungen der Teammitglieder erfolgt und wie die Ausführung dieser Handlungen durch Kommunikation koordiniert, überwacht, bewertet und ggf. korrigiert werden.

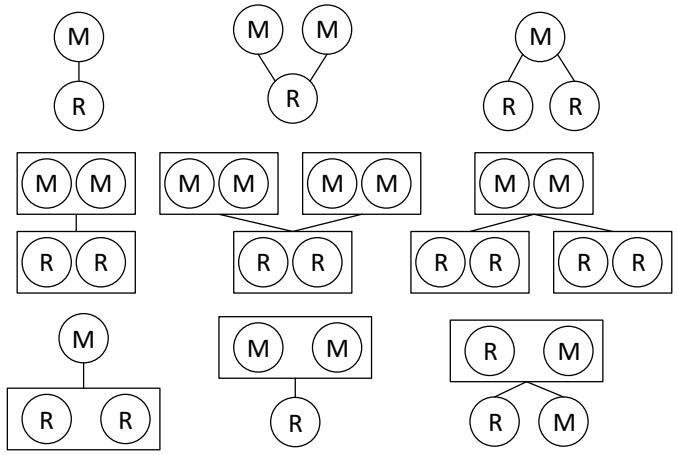

Abbildung 5: Auswahl von möglichen Gruppen Interaktionen nach [89].

Neben der Zusammensetzung und Organisation ist relevant, wie die Zusammenarbeit hinsichtlich Zeit und Raum stattfindet. Aktivitäten können zum Beispiel zeitlich synchron oder asynchron ausgeführt werden; auch können temporale Zusammenhänge bzw. Abhängigkeiten zwischen Handlungen bestehen. So kann es sein, dass eine Handlung vor, nach, parallel oder synchronisiert zu einer anderen Handlung stattfinden muss. Die räumliche Interaktion von Mensch und Roboter wird danach unterschieden, ob Handlungen von Mensch und Roboter in den gleichen oder unterschiedlichen Arbeitsräumen stattfin-

den (siehe Abbildung 6). Weiterhin ist es wichtig, das Annäherungsverhalten des Roboters zu unterscheiden, zum Beispiel, ob der Roboter dem Menschen folgt, ihm aktiv ausweicht, ihn weiträumig umfährt, sich annähert oder den Menschen berührt. Bei der Annäherung spielt auch die Richtung, seine Gestalt und die Annäherungsgeschwindigkeit des Roboters eine entscheidende Rolle für die menschliche Akzeptanz [98–100].

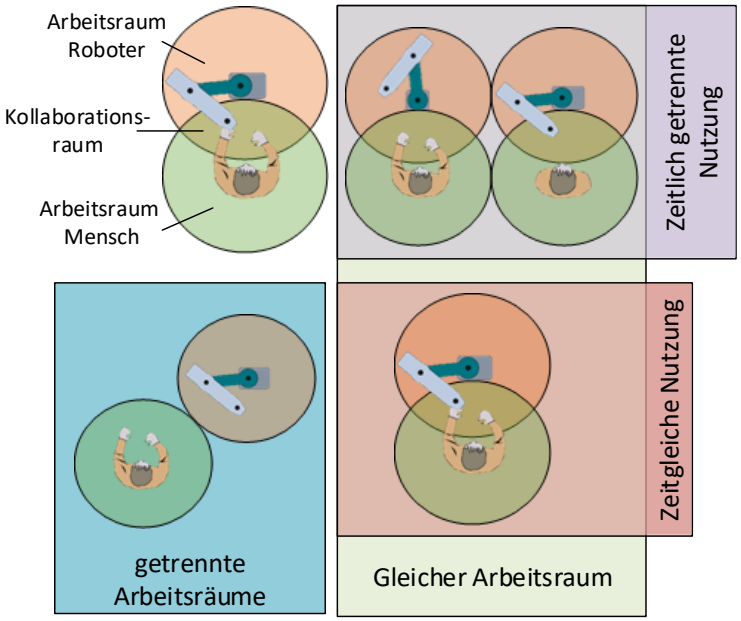

Abbildung 6: Zeitliche und räumliche Ausprägungen der MRI nach [79].

3.3.5 Interaktionsrollen in der Mensch-Roboter-Interaktion

Generell kann ein Mensch in einem MRK-System verschiedene Rollen einnehmen. In der gesichteten Literatur [83, 84, 101] werden Rollenbilder aus dem Bereich der Mensch-Computer- oder Mensch-Maschine-Interaktion übernommen und hinsichtlich der MRI angepasst. Die einzelnen Rollen sollen nun folgend genannt und beschrieben werden.

Die Rolle des Supervisors besteht darin, die längerfristigen Zielstellungen zu definieren und die Gesamtsituation hinsichtlich der Zielerreichung zu kontrollieren. Hierzu überwacht der Supervisor die Robotersysteme und greift ggf. korrigierend ein (vgl. Supervisory Control Tabelle 1). Wenn die untergeordneten Robotersysteme auf Planungssystemen[7] beruhen, die Zielvorgaben selbstständig verfolgen können, dann werden diese Ziele durch den Supervisor definiert.

Die Rolle des Operateurs kann die direkte Steuerung des Robotersystems beinhalten oder darin bestehen, automatisierte Abläufe anzupassen oder neue zu definieren. Der Operator ist dabei auch für die Funktion des Systems verantwortlich.

Die Rolle des Mechanikers (engl.: *mechanic*) erhalten Menschen, die konstruktive Anpassungen, Reparaturen oder die Wartung des Robotersystems verantworten.

Die Rolle des Teamkollegen (engl.: *team member*/*peer*) erhält in der MRI eine weitere Untergliederung in Kollaborateur und Kooperateur. Die Rolle des Kollaborateurs oder Kooperateurs ergibt sich hierbei analog zur jeweiligen Interaktionsform. Der Teamkollege arbeitet mit dem Robotersystem zusammen und kann im Rahmen seiner Tätigkeit dem Robotersystem Befehle erteilen. In der Rolle des Teamkollegen wird in [101] auch der Übergang zu anderen Rollen, zum Beispiel des Operateurs oder des Supervisors, vorgesehen. Die Rolle des Operateurs kann etwa erforderlich werden, wenn der Roboter eine Kollision verursacht, stehenbleibt und anschließend von Hand aus der Kollisionslage geführt werden muss. Der Übergang zur Rolle des Supervisors kann notwendig werden, wenn Ziele neu definiert oder Aufgaben neu (dynamisch) verteilt werden müssen.

Die letzte Rolle ist die des Nichtbeteiligten (engl.: *by-stander*). Auch wenn eine Person nicht mit einem Robotersystem zusammenarbeitet, kann eine Interaktion stattfinden und

[7] Der Begriff des Planungssystems wird später erläutert.

eine Rolle erforderlich sein. Zum Beispiel kann eine mobile Plattform einem Nichtbeteiligten den Weg versperren. In dieser Situation wäre es sinnvoll, der Person Zugang zur Steuerung des Systems (in einem beschränkten Maße) zu geben, um es aus dem Weg zu fahren.

In [83] werden zusätzlich die Rollen des Mentors und des Informationsbereitstellers für den Roboter geschaffen. Als Mentor agiert das Robotersystem als Lehrer oder Führer, zum Beispiel, um den Menschen durch einen komplizierten Prozess zu führen. Als Informationsbereitsteller bietet er dem Menschen Zugang zu Informationen.

Weist man dem Menschen eine Rolle in einem MRK-System zu, wird indirekt auch die Rolle des Roboters festgelegt. Mit der zugewiesenen Rolle verändern sich die Anforderungen an das Robotersystem und an dessen Steuerung.

3.3.6 Sicherheit

An MRK-Systeme im industriellen Umfeld werden hohe Anforderungen hinsichtlich der Sicherheit gestellt, welche in der Maschinenrichtlinie 2006/42/EG sowie weiteren Normen bzw. technische Spezifikation definiert sind. Sicherheitsanforderung für den Roboter, als unvollständige Maschine, werden in der Norm EN ISO 10218-1:2011 festgelegt. Zum Beispiel werden in dieser Norm technische Anforderungen an den Roboter für den kollaborativen Betrieb in einer der folgenden Ausprägungen gefordert: sicherheitsbewerteter überwachter Halt, vorhanden sein einer Zustimmungseinrichtung für die Handführung, Geschwindigkeits- und Abstandsüberwachung oder Leistungs- und Kraftbegrenzung durch inhärente Konstruktion oder Steuerung. Weitere Sicherheitsanforderungen werden an die Steuerung des Roboters durch die Normen EN ISO 13849-1 und 2, z. B. bzgl. der sicherheitsbezogenen Leistungsfähigkeit der Steuerung (*Performance Level*) und Hardware-Fehlertoleranz (*Safety Integrity*), gestellt. Für das Robotersystem und dessen Integration gilt die EN ISO 10218-2:2011. Hier werden Sicherheitsanforderungen für das Gesamtsystem, z. B. bzgl. der Endeffektoren sowie der gehandhabten Werkstücke, der Definition geschützter Bereiche und der Beherrschung gespeicherter Energien (z. B. Druckluft), vorgegeben. Für MRK-Anwendungen werden weitere Vorgaben durch die technische Spezifikation ISO/TS 15066:2016 beschrieben, z. B. werden Grenzwerte für verschiedene Körperbereiche des Menschen hinsichtlich Klemmen oder Kollisionen bei

der Bewegung mit dem Roboter festgelegt. Für den Betrieb eines MRK-Systems ist letztendlich die CE-Zertifizierung erforderlich. Für Details zu diesem komplexen mehrstufigen Prozess, wird auf weiterführende Literatur [11, 102, 103] verwiesen.

4 Einsatzgebiet Demontage

Die Demontage kommt als Prozess in der Produktion, der Instandsetzung, der Wartung, in der Refabrikation von Produkten und Komponenten sowie bei der Entsorgung zur Anwendung. Jeder Bereich stellt eine andere Umgebung dar, wodurch die Anforderungen variieren. Folgend sollen die Einsatzgebiete der Instandsetzung, Wartung und Refabrikation detailliert betrachtet werden. Dabei sollen insbesondere die Unterschiede zwischen Montage und Demontage aufgezeigt werden.

4.1 Demontage im Bereich der Instandsetzung

Die DIN 31051 beschreibt die Instandsetzung als

> *„Maßnahmen zur Rückführung in den funktionsfähigen Zustand (nach Ausfall), mit Ausnahme von Verbesserungen“ [104, 105]*.

Die Rückführung in den funktionsfähigen Zustand erfolgt (bei Bauteilversagen) durch Austausch des defekten Bauteils. Hierzu muss dieses zuerst demontiert, ersetzt und danach das Produkt wieder zusammengesetzt werden. Die zerstörungsfreie Demontage des defekten Bauteils erfolgt zielgerichtet, wenn die Ursache (das defekte Bauteil) und der Produktaufbau bekannt sind. Im anderen Fall muss die Fachkraft die Ursache des Ausfalls ermitteln und evtl. durch die schrittweise Demontage des Produktes sich dem Problem nähern. Die Instandsetzung zeigt an diesem Punkt deutlich ihre höhere Komplexität gegenüber der Montage. Während in der Montage ein Produkt ‚nur‘ zusammengesetzt wird, muss in der Instandsetzung auch die zielgerichtete Demontage von einzelnen Bauteilen betrachtet werden. Einem Montageprozess steht damit unter Umständen eine Vielzahl von möglichen Demontageprozessen gegenüber. Exponentiell verstärkt wird diese Problematik durch variantenreiche oder individuelle Produkte (Losgröße 1). Beim Zerlegen eines Produktes müssen weiterhin unterschiedliche Bauteile demontiert und gehandhabt werden. Der Demontageprozess eines jeden Bauteils wird dabei maßgeblich durch die Verbindungstechnik(en), die mit der Baugruppe bestehen, beeinflusst. Die Art der Verbindungstechnik und das Bauteil bestimmen weiterhin Ressourcen, wie Werkzeuge oder Maschinen, und die Ausprägungen von

Demontagehandlungen. Innerhalb der Instandsetzung muss durch den unvorhersehbaren Ausfall von Komponenten außerdem mit einem nicht kontinuierlichen Materialfluss unterschiedlicher Produkte bzw. Varianten von Produkten gerechnet werden. Ein weiterer Unterschied zur Montage von Produkten ist, dass der Produktzustand unvorhersehbar vom Ideal abweichen kann, zum Beispiel durch defekte oder fehlende Bauteile, aber auch Verschmutzung oder Verschleiß wirkt auf die Bauteile ein. Darüber hinaus sind Manipulationen an den Produkten zu erwarten, etwa in Form von improvisierten Reparaturen oder Funktionserweiterungen. Dieser Produkt- oder Bauteilzustand kann es erfordern dass der ‚normale' Demontageprozess anhand der vorliegenden Situation angepasst werden muss. Hierzu muss der Produktzustand richtig interpretiert und antizipiert werden, ob der ‚normale' Demontageprozess zum gewünschten Ziel führt. Ist dies nicht der Fall, muss ein neuer Demontageprozess ermittelt werden – wofür ein bestimmtes Maß an Kreativität, Fachkenntnis sowie ein breites Spektrum von Fähigkeiten und Fertigkeiten notwendig sind. Innerhalb der Instandsetzung herrscht zudem Zeitdruck. Um den eventuellen Ausfall wichtiger Infrastruktur zu minimieren, muss entschlossen und schnell gehandelt werden. Im Gegensatz zur Montage sind zudem meist keine an die Aufgabestellung perfekt zugeschnittenen Arbeitsplätze vorhanden.

4.2 Demontage im Bereich der Wartung

Unter Wartung wird nach DIN 31051 die

> *„Verzögerung des Abbaus von Abnutzungsvorrat durch Verminderung der Abnutzungsgeschwindigkeit einer Betriebseinheit" [104, 105]*

verstanden. Für die Wartung von Produkten sind meist Wartungspläne vorhanden, die beschreiben, wann, wie und welche Wartungstätigkeiten erfolgen müssen. Einzelne Wartungstätigkeiten können hier unterschiedliche Aufgabenstellungen beschreiben, zum Beispiel das Wechseln des Getriebeöls, das Wechseln einer Bremse oder das Reinigen des Produktes. Die Wartung erfolgt i. d. R. deshalb durch speziell geschultes Personal. Da die Wartung im besten Fall bei der Konstruktion des Produktes berücksichtigt wird, sind zudem keine aufwendigen Demontageprozesse anzunehmen. Der Zeitpunkt der Wartung ist außerdem meist planbar und der Zeitdruck ist im Vergleich zur Instandsetzung geringer. Gegenüber der Instandsetzung muss bei der Wartung nur mit Bauteilen gerechnet werden,

die eine bestimmte Abnutzung vorweisen und noch funktionsfähig sind. Der Produktzustand ist somit gegenüber der Instandsetzung als ‚besser' anzusehen.

4.3 Demontage im Bereich der Refabrikation

Der Begriff ‚Refabrikation' ist besser unter dem englischen Namen ‚Remanufacturing' bekannt. Bei der Refabrikation handelt es sich um ein höherwertiges Produktrecycling. Beim sogenannten Produktrecycling wird das ganze Produkt, oder Komponenten des Produkts, wiederaufbereitet und einer neuen Verwendung zugeführt [106]. Die Refabrikation eines Produktes gliedert sich in fünf Schritte [107]:

- Demontage,
- Reinigung,
- Prüfen und Sortieren,
- Aufarbeiten oder Ersetzen,
- Montage und Endkontrolle.

Die wiederaufbereiteten Komponenten können als Ersatzteile verkauft oder zum Bau neuer Produkte verwendet werden. Daher muss die Demontage innerhalb der Refabrikation zerstörungsfrei und möglichst beschädigungsfrei erfolgen. Als Standort für die Refabrikation bieten sich die Produktionswerke der Hersteller an, da dort die Produktion direkt angeschlossen ist. Sammelstellen in den Werken könnten dazu dienen, einen gleichmäßigeren Materialfluss gleicher Produkte dem Demontageprozess zuzuführen. Hier muss jedoch im Hinblick auf individuelle Produkte oder solche mit hoher Variantenzahl erwähnt werden, dass der Refabrikationsprozess an jedes Produkt und dessen Zustand angepasst werden muss. Der Produktzustand kann auch hier variieren und alle bereits beschriebenen Zustandsformen annehmen. Durch die Demontage erfolgt die Vereinzelung der Komponenten, um sie nachfolgend dem Wiederaufbereitungsprozess zuzuführen. Dabei ist die Logistik entscheidend, denn für jedes einzelne Bauteil muss bekannt sein, ob und welche Wiederaufbereitungsprozesse das Bauteil durchläuft und wo es in der Produktion benötigt wird. Aus Sicht der Materialflusssteuerung ist dies eine Herausforderung, die bereits am Demontagearbeitsplatz beginnt. Hier müssen die Bauteile demontiert und in passende Transportbehälter vereinzelt werden. Bei der Vereinzelung muss dabei protokolliert werden, welches Bauteil in welcher Transportbox liegt, denn jedes

Bauteil kann an eine gesonderte, örtlich getrennte Wiederaufarbeitungs- oder Entsorgungsstation transportiert werden. Außerdem kann sich der Wiederaufarbeitungsprozess am Zustand des Bauteils orientieren, was eine weitere Komplizierung des Prozesses nach sich zieht.

4.4 Zusammenfassung der Anforderungen

Insbesondere der Bereich der Instandsetzung stellt hohe Anforderungen an das RAS. So muss sich das RAS an unterschiedliche Produkte und an variable Zielstellungen hinsichtlich der Demontage anpassen und den Menschen in unterschiedlichen Aktivitäten unterstützen. Die vom RAS ausgehende Unterstützung und Verhalten sollte für den Menschen hervorsehbar, aber sich auch an unterschiedliche Situationen anpassen können. Auch sollte das Assistenzverhalten des RAS aufgabenspezifisch und auf Basis einer einheitlichen Methodologie entworfen und umgesetzt werden. Durch schrittweises Hinzufügen von Assistenzfunktionen für einzelne Aufgabenstellungen der Demontage könnte das RAS dem Menschen immer breitere Unterstützung bieten. Hierzu ist insbesondere die einfache Einbindung von weiteren Geräten und Peripherie in RAS zu berücksichtigen. Physisch unterstützen könnte das RAS den Menschen in der Demontage durch die Übernahme von Demontagehandlungen, wie vom Lösen von Schraubenverbindungen, oder durch direkte Kollaboration, zum Beispiel bei der gemeinsamen Handhabung von demontierbaren schweren Unterbaugruppen. Generell gilt, dass praktisch jede menschliche Aktivität bei der Demontage in irgendeiner Art durch ein technisches System unterstützbar ist. Nur technische Gegebenheiten oder Aspekte bzgl. der Sicherheit sowie das Kosten-Nutzen-Verhältnis bestimmen die Sinnhaftigkeit der Umsetzung. Die vom RAS ausgehende Unterstützung sollte sich dabei nicht nur auf die physische Unterstützung beschränken, sondern kann auch psychischer Natur sein. Versetzt man sich zum Beispiel in die Lage des Mitarbeiters, dem die Ausfallursache und evtl. das defekte Produkt unbekannt ist, versteht man, worauf der erste Aspekt der kognitiven Ergonomie [70] „Motivation, Aktivierung und Zielsetzung“ abzielt. Der Mitarbeiter kann in dieser Situation zuerst handlungsunfähig sein, weil er nicht weiß, was zum Ausfall des Produktes geführt hat. Unterstützt das System den Nutzer durch eine Zielvorgabe, kann dieser ohne Zögern die Arbeit aufnehmen. Zum Beispiel könnte das Assistenzsystem vorschlagen, welche Bauteile den Ausfall verursacht haben könnten. Der Vorschlag kann dabei auf gesammeltem

Wissen durch eine Zustandsüberwachung (engl.: Condition Monitoring) oder auf vorherigen Instandsetzungen gleicher Produkte beruhen. Ist dem Menschen das Produkt dazu unbekannt, könnte das Assistenzsystem den Menschen durch die einzelnen Demontageschritte führen und ihn anleiten. Für diese Fähigkeit muss das RAS in der Lage sein, anhand einer Zielvorgabe, etwa der Demontage eines gewissen Bauteils, die notwendigen Demontageschritte automatisiert zu bestimmen. Hierbei muss berücksichtigt werden, dass der Produktzustand – und damit der Demontageprozess – unvorhersehbare Formen annehmen kann. Da nur durch den Menschen der Produktzustand und die Situation in ihrer Gesamtheit vollständig erfasst und der Demontageprozess richtig antizipiert werden kann, muss der Mensch die Rolle des Supervisors einnehmen, den Demontageprozess ggf. anpassen und dem RAS vorgeben, wann und welche Art von Assistenz gewünscht bzw. möglich ist.

Einige aufwendig zu realisierende Anforderungen werden an das Robotersystem gestellt; hierzu zählt das Greifen und Handhaben von Bauteilen mit unterschiedlichen Geometrien. Hier muss man sich der Problematik bewusst sein, dass für die sichere Handhabung eines Bauteils i. d. R. speziell angepasste Greifer verwendet werden. Für die Handhabung von vielen unterschiedlich geformten Bauteilen kann dies bedeuten, dass vielfältige Greifer verwendet und in einem Werkzeugmagazin bevorratet werden müssen. Um den menschlichen Nutzer nicht zusätzlich mit dem Wechseln der Greifer zu belasten, sollte das Robotersystem diese automatisiert wechseln können. Diese Problematik kann durch den Einsatz von Magnet- und Vakuumgreifern oder durch flexible Greifsysteme, die sich der Form des Bauteils anpassen können (engl.: *soft grippers*), jedoch unter günstigen Umständen entschärft werden. Neben Werkzeugen für die Handhabungen müssen ebenfalls Werkzeuge (Effektoren) für die Manipulation von Bauteilen bereitstehen. Hierzu müssen spezielle Roboterwerkzeuge für das Lösen von verschiedenen Verbindungstechniken und die zugehörigen Roboterfertigkeiten (Skills) entwickelt werden. Bei der Handhabung und Manipulation von Bauteilen besteht generell die Problematik des beschränkten Arbeitsraums sowie der begrenzten Nutzlast MRK-fähiger Industrieroboter. Eine weitere Herausforderung ist die Programmierung des Roboters für diese Art der Problemstellung. Ständig wechselnde Aktivitäten, in denen unterschiedlichen Bauteile zwischen unterschiedlichen Orten bewegt oder unter Zuhilfenahme von verschiedenen Werkzeugen manipuliert werden, verhindern die Erstellung und Verwendung eines üblichen Roboterprogramms. Auch moderne Ansätze wie das Programmieren durch Demonstration (engl.:

Programming by Demonstration) verlieren hinsichtlich der ständig wechselnden Tätigkeiten ihre Vorteile. Eine für diesen Fall geeignete Form der Roboterprogrammierung wäre die instruktive Programmierung [108]. Bei dieser Form der Programmierung werden dem Robotersystem zur Laufzeit Instruktionen gesendet. Diese Instruktionen veranlassen entweder die Ausführung grundlegender Funktionen[8], wie Bewegungsbefehlen, oder die Ausführung von Aufgaben[9], für die der Roboter im Vorhinein programmiert wurde, zum Beispiel das Rüsten eines Werkzeugs. Damit sich das Robotersystem den ändernden Aufgabenstellungen der Demontage anpassen kann, müssen diese Instruktionen geeignete parametrierbare Variablen für die entsprechenden Aktionen oder Aufgaben enthalten, zum Beispiel eine Variable, die den Namen des zu rüstenden Werkzeugs definiert. Die Steuerung des Roboters könnte nun durch Instruktionen direkt durch den Menschen erfolgen. Wobei das Instruieren des Robotersystems auf Basis einzelner parametrierbarer Aktionen und Aufgaben den Menschen schnell überfordern würde. Denn allein für die Ausführung einer linearen Bewegung in kartesischen Raum benötigt ein Roboter die Angabe eines Werkzeugs (TCP), eines Bezugskoordinatensystems, die Zielposition und Orientierung des TCPs, eine kartesische Geschwindigkeit und, um evtl. die Roboterpose eindeutig festzulegen, die Angabe von Konfigurationsparametern. Diese Informationen über die Mensch-Maschine-Schnittstelle zu übertragen, wäre zeitaufwendig und würde zu einer uneffektiven Zusammenarbeit führen. Außerdem muss bedacht werden, dass der Mensch viele dieser Informationen gar nicht besitzt und keinen Zugang zu diesen hätte. Um den Menschen zu entlasten, könnte das RAS zwischen Mensch und Roboter geschaltet werden. Für den Menschen würde das RAS die zugrunde liegende Komplexität verdecken und Instruktionen auf einer höheren Abstraktionsebene bereitstellen, zum Beispiel Instruktionen wie „Löse diese Schraube.“. Das RAS würde dann die entsprechenden Instruktionen mit den entsprechenden Variablenwerten an den Roboter senden. Hier stellt sich allerdings die Frage, woher das RAS diese Variablenwerte besitzt. Eine Möglichkeit bezieht sich auf maschinelle Wahrnehmung, eine andere auf explizites Wissen. Wobei die maschinelle Wahrnehmung nur eingeschränkt fähig ist, diese Variablenwerte zu liefern. So können einzelne Variablenwerte gar nicht oder nur mit erheblichem Aufwand

[8] Im späteren Verlauf dieser Arbeit werden diese grundlegenden Funktionen als Aktionen bezeichnet.

[9] Der entsprechende engl. Fachbegriff wäre „skills“. Ein Skill beschreibt die Fähigkeit und Fertigkeit eines Roboters für eine spezielle Aufgabe [109]. Ein Skill besteht i.d.R. aus einer Sequenz von Aktionen.

aus der Umgebung des Demontageprozesses ermittelt werden. Im Bereich der künstlichen Intelligenz wird dieser Umstand als partiell wahrnehmbare Umwelt bezeichnet. Aus diesem Grund ist die Repräsentation von explizitem Wissen bzgl. der Demontage notwendig und vorteilhaft.

Eine weitere Anforderung an das RAS bezieht sich auf die Koordination der Teammitglieder, denn das RAS muss während des gesamten Demontageprozesses nicht nur die Handlungen des Menschen und eines Roboters, sondern ggf. auch die Handlungen von mehreren Robotern sowie weiterer Peripherie, wie Greifern und anderen Maschinen oder Geräten, planen, koordinieren und überwachen. Innerhalb der Demontage muss auch immer damit gerechnet werden, dass einzelne Handlungen nicht das erwartete Ergebnis hervorbringen. So kann z. B. das Lösen einer festgerösteten Schraube nicht gelingen. Die Erkennung solcher Fehler und die daraus folgende Adaption des Demontageprozesses sind somit wichtige Voraussetzungen für das RAS.

Die in diesem Kapitel formulierten Anforderungen an das RAS erfordern einen ganzheitlichen Systementwurf, der insbesondere die Umsetzung einer geeigneten technischen Autonomie unterstützt. Aus diesem Grund erfolgt im folgenden Kapitel die Betrachtung möglicher Systemarchitekturen und Methoden zur Umsetzung einer geeigneten technischen Autonomie.

5 Theoretische und methodische Grundlagen zur Umsetzung von Autonomie

Innerhalb dieses Kapitels soll zuerst ein Überblick über unterschiedliche Paradigmen[10] und Konzepte zur Umsetzung von Autonomie in technischen Systemen betrachtet werden. Neben den Paradigmen aus dem Bereich der Robotik sollen auch Konzepte aus der Informatik (aus dem Bereich der künstlichen Intelligenz) und der Kybernetik betrachtet werden. Für bestimmte Paradigmen oder Konzepte werden hierzu repräsentative Systemarchitekturen dargestellt, die Entwurfsmuster für deren Implementierung zeigen. Ziel dieses Kapitels ist es, ein geeignetes Paradigma oder Konzept sowie eine Architektur für die Entwicklung des RAS zu bestimmen.

Im darauffolgenden Kapitel 5.2 soll das Grundlagenwissen für die Methoden zur automatisierten Demontageplanung sowie zur Koordinierung der Handlungen durch ein Planungssystem geschaffen werden. Hierzu werden zuerst generische Planungsmethoden der künstlichen Intelligenz besprochen und dann wird auf spezielle Methoden für die Demontageplanung eingegangen.

5.1 Paradigmen, Konzepte und Architekturen autonomer Systeme

Zunächst erfolgt eine Betrachtung der Paradigmen, die sich im Bereich der Robotik für die Umsetzung von intelligentem Verhalten gebildet haben. Danach werden Konzepte erörtert, die aus der Informatik stammen. Am Ende dieses Kapitels wird ein Ansatz aus der Kybernetik beschrieben.

[10] Ein Paradigma beschreibt eine grundlegende Denkweise, die in einem Fachgebiet als anerkannter Konsens über Annahmen und Vorstellungen einer Problemstellung gilt.

5.1.1 Paradigmen und Architekturen aus der Robotik

Für die Umsetzung von Autonomie in einem Robotersystem haben sich die drei folgende Paradigmen herausgebildet: das hierarchische, das reaktive und das hybride Paradigma [110–114]. Das hierarchische und das reaktive Paradigma sind widersprüchliche Denkweisen und beschreiben unterschiedliche Ansätze zur Umsetzung von Autonomie. Das hybride Paradigma hingegen kombiniert die beiden Ansätze. Die einzelnen Paradigmen sollen nun beschrieben werden.

5.1.1.1 Hierarchisches Paradigma

Bei diesem Paradigma werden die drei grundlegenden Aspekte der Robotik, das Wahrnehmen (engl.: *sense*), Planen (engl.: *plan*) und Handeln (engl.: *act*) in einem Kreislauf verbunden (siehe Abbildung 7). Grundgedanke dieser auch als ‚deliberative Architektur' (engl.: *deliberative architecture*) bezeichneten Systeme ist die physikalische Symbolsystemhypothese[11] (PSSH) [115] (engl.: *physical symbol system hypothesis*). In dieser Annahme werden die Dinge der realen Umwelt durch elementare und eindeutig identifizierbare Symbole in einem Symbolsystem dargestellt, z. B. {Hund, Katze}. Anhand syntaktischer Regeln können diese Symbole zu (logischen) Ausdrücken oder Symbolstrukturen, wie Relationen jagd(x, y), zusammengesetzt werden und dadurch den Zustand der Umwelt des Roboters repräsentieren, z. B. jagd(Hund, Katze). In der PSSH wird Kognition als Informationsverarbeitung angesehen, die auf Transformation der Symbolstrukturen, durch Anwendung sogenannter Operatoren, beruht. Besitzt ein System eine adäquate Repräsentation, in Form eines Symbolsystems sowie Operatoren, um deren Struktur zu manipulieren, kann es zur Problemlösung dienen und damit kognitive Funktionen umsetzen. Eine technische Umsetzung diese Paradigmas fand auf Basis des generischen Planungssystems STRIPS[12] in dem Roboter Shakey [116] statt. Die Performanz des Systems war, auch aufgrund der damals fehlenden Rechenleistung, eingeschränkt und führte zu einer Gegenbewegung in Form des reaktiven Paradigmas [110]. Die Nachteile dieses Ansatzes

[11] In der KI existieren zwei wesentliche Ansätze zur Umsetzung von intelligentem Verhalten, der Top-Down Entwurf entspricht der PSSH und der Bottom-Up Entwurf entspricht dem Konnektionismus. Beide Ansätze spalten die KI in symbolisch KI und subsymbolische KI.

[12] STRIPS steht für „Stanford Research Institute General Problem Solver".

waren die exzessive Nutzung von rechenaufwendigen Planungsalgorithmen und die Verwendung eines globalen Zustandsmodells. Letzteres ist ungünstig, da die Entwicklung von adäquaten, alles beschreibenden Repräsentationen durch das sogenannte Rahmenproblem (engl.: Frame-Problem) [117], der geschlossenen Weltannahme (engl.: *closed world assumption*) [118] und das Deutungsproblem von Symbolen (engl.: *symbol grounding problem*) [119] von Teilen der wissenschaftlichen Gemeinschaft als unmöglich angesehen wird.

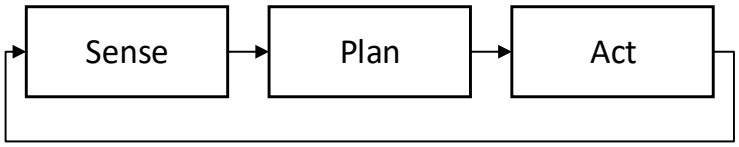

Abbildung 7: Sense-Plan-Act-Zyklus [109].

5.1.1.2 Reaktives Paradigma

Der Ansatz des reaktiven Paradigmas steht im Kontrast zum hierarchischen Paradigma. Zum Beispiel verzichtet der Ansatz vollständig auf ein Modell der Umwelt. Dahinter steht die Kernthese, dass intelligentes Verhalten ohne explizite Repräsentation und ohne Schlussfolgerungen entstehen kann. Intelligenz sei eine sich herausbildende (emergente) Eigenschaft, die durch Interaktion von Individuen in komplexen Systemen entsteht [120]. Umgesetzt wird dieses Paradigma durch die direkte Kopplung von Wahrnehmung und Handeln in Form von Verhaltensformen (engl.: *behaviors*), siehe Abbildung 8. Dieses Paradigma ist biologisch inspiriert, durch frühe Verhaltensuntersuchungen an Menschen und Tieren, die in reflexive, reaktive oder bewusste Formen unterschieden wurden. Das reaktive Paradigma steht für reflexive und reaktive Verhaltensformen, während das hierarchische Paradigma für bewusstes Verhalten steht. Reflexives Verhalten (engl.: *reflexive behaviors*) meint angeborene Reiz-Reaktionen (engl.: *stimulus-response*) wie den Patellarsehnenreflex oder den Lidschlussreflex. Reaktives Verhalten (engl.: *reactive behaviors*) bezieht sich auf angeborenes oder erlerntes Verhalten, das ohne bewusstes Denken ausgeführt wird, z. B. Fahrrad fahren oder Schwimmen. In letzteren Sinne wird eine Verhaltensform als direkte Abbildung von Sensorinformationen zu Bewegungsmustern (Motor-Schemata) verstanden [110].

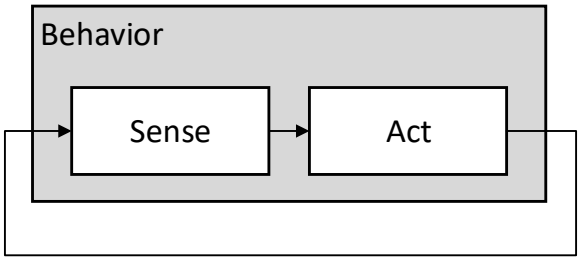

Abbildung 8: Sense-Act-Zyklus [110].

Diese Art der verhaltensbasierten Steuerung (engl.: *behavior-based control*) führte in der frühen Phase zu großen Erfolgen [121], zum Beispiel in Form der „subsumption architecture“ [120], siehe Abbildung 9. In dieser Architektur existieren mehrere Verhaltensformen parallel zueinander, wobei höhere Verhaltensformen untergeordnete Verhaltensformen unterdrücken bzw. zurücksetzen können [110]. Später zeigte sich jedoch, dass komplexe Verhaltensformen nur schwer auf diese Art zu realisieren sind.

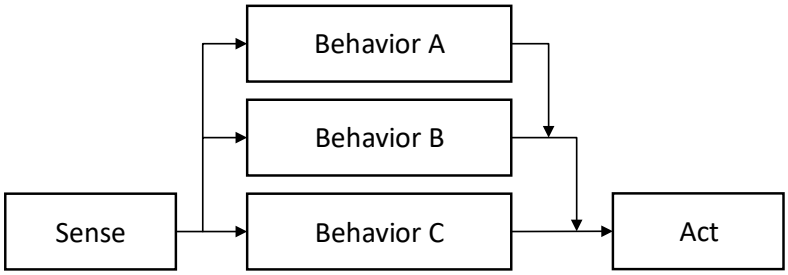

Abbildung 9: Grundlegende Form der Subsumption-Architektur [110].

5.1.1.3 Hybrides Paradigma

Die Kombination der beiden vorherigen Paradigmen führte zum heutigen vorherrschenden hybriden Paradigma. Hier erfolgt das Planen getrennt vom reaktiven Verhalten (Wahrnehmen und Handeln), wie in Abbildung 10 gezeigt. Umgesetzt wurde dieses Paradigma in einer Vielzahl von unterschiedlichen Architekturen, zum Beispiel in der „Autonomous Robot Architecture (AuRA)“, der „Task Control Architecture (TCA)“ oder der „three tier architecture (3T)“, siehe [111].

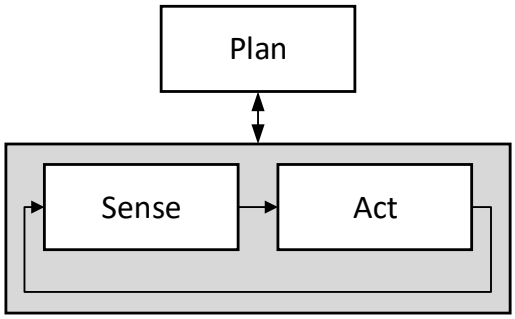

Abbildung 10: Plan, Sense-Act-Zyklus .

Letztere ist Gegenstand aktueller Forschung [122, 123] im Bereich der Service-Robotik. Ihre grundlegende Architektur ist in Abbildung 11 dargestellt. Wichtigstes Element dieser Architektur ist die vermittelnde Schicht, die die schnell arbeitende reaktive Ebene mit der langsam arbeitenden Planungsebene verbindet. Für eine detaillierte Beschreibung dieser Architektur muss auf [124] verwiesen werden.

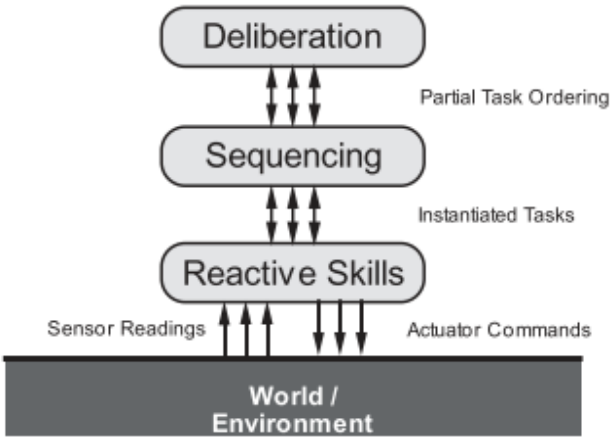

Abbildung 11: Three-Tier(3T)-Architecture aus [122].

Eine weitere umgesetzte Architektur für einen mobilen Manipulator konnte in [114] gefunden werden. Diese Architektur implementiert eine Wissensbasis auf Grundlage einer Ontologie (OWL-DL[13]) in der Planungsebene. Diese Ontologie wird zusammen mit den

[13] OWL steht für Web Ontology Language, eine Spezifikation die für den Austausch von Ontologien dient und durch das World Wide Web Consortiums (W3C) veröffentlicht wurde. DL steht weiterhin für Description Logic d. h. Beschreibungslogik.

Nutzereingaben dazu verwendet, Planungsprobleme für ein Planungssystem zu generieren. Damit prägt diese Architektur eine moderne Form des hybriden Paradigmas, die durch einen modellbasierten Stil seitens der Informatik geprägt wurde.

5.1.2 Konzepte und Architekturen aus der Informatik

Eine von der Informatik ausgehende Sichtweise ist, dass intelligentes Verhalten grundlegend von sogenannten rationalen Agenten ausgeht. Daher soll das Konzept rationaler Agenten und Systeme mit mehreren interagierenden Agenten, sogenannten Multiagentensystemen, in diesem Kapitel betrachtet werden. Ein weiteres Konzept aus der Informatik, das von den Kognitionswissenschaften geprägt wurde, beschreibt kognitive Systeme. Dieses Konzept soll ebenfalls erörtert werden.

5.1.2.1 Konzept rationaler Agenten

Die Definition eines Agenten kann breit gefasst werden, so lautet eine Definition:

> *An Agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.[125]*

Nach dieser Definition kann ein Mensch, ein Roboter und eine Vielzahl anderer technischer Systeme grundsätzlich als Agent aufgefasst werden. An rationale Agenten wird weiterhin die Forderung gestellt, dass ihr Handeln rationales Verhalten aufzeigt, d. h., dass ihr Handeln in keinem Widerspruch zu ihren Zielen steht [125]. In der Literatur findet sich eine Vielzahl von Konzepten und Architekturen für verschiedene Arten von Agenten. Dazu gehören einfache reflexive Agenten sowie modellbasierte reflexive Agenten, die zusätzlich über ein Modell ihrer Umwelt verfügen. Modellbasierte Agenten wissen, zu welchen Handlungen (Aktionen) sie fähig sind, wann eine Handlung anwendbar ist und welche Effekte diese Aktion auf den Zustand ihrer Umwelt hat. Eine daraus resultierende zentrale Fragestellung ist, wie geeignete Modelle der Umwelt und der Handlungen umgesetzt werden. Hierzu wird in Kapitel 5.2.2 eine Möglichkeit aufgezeigt. Dieses Wissen kann in ziel- und nutzenbasierten Agenten effektiv genutzt werden. Zielbasierte Agenten können das ihnen gestellte Ziel langfristig selbstständig verfolgen; nutzenbasierte Agenten versuchen weiterhin, dieses Ziel in einer optimalen Weise, bewertet anhand einer Nutzenfunktion, zu erreichen. Um sein Ziel zu erreichen, muss der Agent hierzu seine Handlungen zielgerichtet auszuwählen, wozu einfache Regeln nach der Art

‚wenn Zustand A, dann Aktion B' in komplexen Umgebungen i. d. R. nicht ausreichen. Hier kommen Methoden der künstlichen Intelligenz wie Suchverfahren oder Planungsverfahren zum Einsatz [126–128], die in Kapitel 5.2.2 beschrieben werden. Nutzenbasierte Agenten weisen zwei weitere Vorteile auf. So können Kompromisse im Fall konkurrierender und teilerreichbarer Ziele gefunden sowie Entscheidungen auf Basis des Verhältnisses von Wahrscheinlichkeit und Nutzen getroffen werden. Nutzenbasierte Agenten sind damit für Umgebungen besser geeignet, in denen Unsicherheit bzgl. des Wissens, des Zustands der Umwelt oder der Effekte von Handlungen herrscht [125]. Erweitert wird das Konzept durch lernende Agenten. Lernen gibt einem Agenten die Möglichkeit, Vorhersagen zu bilden, eine beobachtete Situation zu klassifizieren und Cluster von ähnlichen Situationen zu bilden sowie sein Verhalten entsprechend der Situation zu ändern oder zu optimieren.

Eine weitere Form einer Agentenarchitektur, die auf Grundlage von drei mentalen Attributen des Menschen entwickelt wurde, ist der BDI-Agent [129]. Jeder BDI-Agent besitzt eine eigene Auffassung über den Zustand seiner Umwelt (engl.: *beliefs*) sowie Wünsche und Ziele (engl.: *desires*) und folgt Intentionen (engl.: *intentions*). Das Verhalten des BDI-Agenten wird durch den Interpreter festgelegt (siehe Abbildung 12).

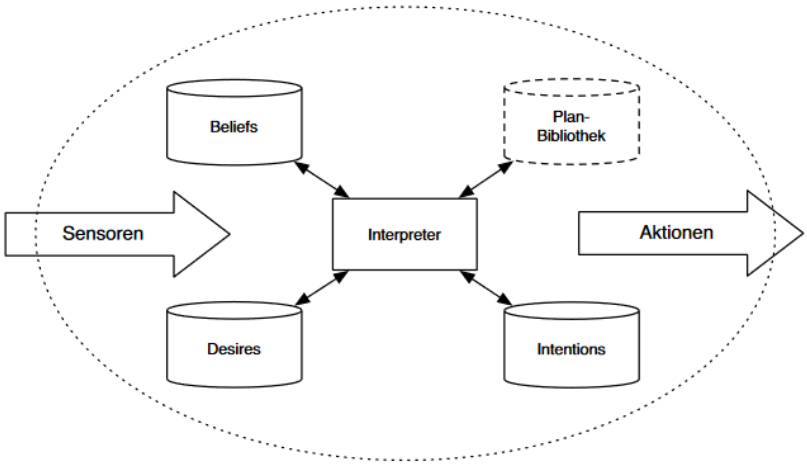

Abbildung 12: Architektur eines BDI-Agenten [130].

Wenn mentale Attribute verändert werden, zum Beispiel, wenn sich der Zustand der Umgebung durch die Wahrnehmung verändert (engl.: *belief revision*), bestimmt der Interpreter, ob er weiter seiner Intention und seinem Ziel folgen soll oder ob er einen der beiden Aspekte verändert. Für die Verfolgung einer Intension wird ein passender Plan aus der Planbibliothek des Agenten ausgewählt und die darin beschriebene Folge von Handlungen ausgeführt. Durch die strikte Methodik und das einfache Konzept der BDI-Agenten erlangte dieses Konzept großes Interesse, insbesondere in Multiagentensystemen (MAS).

5.1.2.2 Konzept von Multiagentensystemen

Nach [128] wird ein MAS durch eine Umgebung beschrieben, die aus einer Menge von Objekten sowie Agenten besteht. Objekte können durch Agenten auf Basis von Operationen wahrgenommen, hergestellt, zerstört, modifiziert und manipuliert werden. Zwischen Agenten und Objekten können weitere Beziehungen oder Abhängigkeiten bestehen, die durch Relationen dargestellt werden. Entsprechend der individuellen Zielstellung der Agenten innerhalb dieser Umgebung entsteht eine bestimmte Art der Interaktion, die zu Unabhängigkeit, Kooperation oder aber zu Wettbewerb führt. Die Methoden, die zur Realisierung von Kooperation zwischen Agenten dienen, sollen kurz thematisiert werden. Grundlegend hierfür ist die Bildung von organisationalen Strukturen wie Teams. In Teams können Aufgaben sowie Ressourcen effektiv verteilt und Handlungen koordiniert werden. Auch eine Spezialisierung von einzelnen Individuen für bestimmte Aufgabenstellungen ist möglich. Grundlegend hierfür ist die Fähigkeit eines jeden Agenten zur Kommunikation, was zu einer veränderten Definition eines Agenten nach [128] führt.

Ein Agent wird als virtuelle oder physische Entität beschrieben und ist dazu fähig, seine Umgebung in einem beschränkten Maße wahrzunehmen, in dieser Umgebung zu handeln und mit anderen Agenten zu kommunizieren. Weiterhin verfügt jeder Agent über ein unvollständiges Modell seiner Umgebung, stellt Ressourcen sowie Dienste bereit, besitzt Fähigkeiten und Fertigkeiten. Das Verhalten eines Agenten soll auf die Erfüllung seiner Ziele abzielen – unter Berücksichtigung der verfügbaren Ressourcen, seiner Fähigkeiten und Fertigkeiten, seiner Wahrnehmung und Kommunikation [128].

Fragestellungen, die in MAS und Organisationen von Agenten auftreten, sind vielfältig und können in Rahmen dieses Kapitels oder dieser Arbeit nur ansatzweise dargestellt werden. Für eine allgemeine Einführung in MAS muss auf [128, 131, 132] verwiesen werden. Jedoch soll auf die drei wesentlichen Elemente, die Aufgabendekomposition, die

Aufgabenallokation und die Koordination von Handlungen, die für die Zusammenarbeit von Agenten in einem MAS notwendig sind, kurz eingegangen werden. Die einzelnen Schritte sind in Abbildung 13 präsentiert.

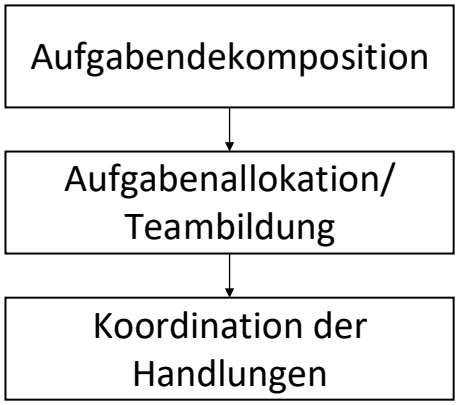

Abbildung 13: Schritte zur Kooperation oder Kollaboration in MAS.

In Fällen, in denen die Fähigkeiten eines Agenten für eine komplexe Aufgabenstellung nicht ausreichen, ist es zwingend erforderlich, diese in Teilaufgaben zu zergliedern (Aufgabendekomposition) und entsprechend den Fähigkeiten der Agenten zu verteilen. Wobei die Aufgabendekomposition grundsätzlich die Möglichkeit bietet, komplexe Aufgaben zu zergliedern, aufzuteilen und parallel auszuführen. Eine Methode für die Aufgabendekomposition in der Demontage wird in Kapitel 5.2.3 getrennt behandelt.

Für den nachfolgenden Schritt der Aufgabenverteilung, d. h. die Zuordnung von Aufgaben zu Agenten (Aufgabenallokation), können in MAS zentrale oder dezentrale Ansätze verwendet werden. Eine Ausprägung des zentralen Ansatzes kann durch eine feste Zuordnung von Aufgaben zu einzelnen Agenten erfolgen. Eine weitere Möglichkeit des zentralen Ansatzes stützt sich auf einen Händler (engl.: *trader*), der zwischen einem Agenten (Kunden), der eine Aufgabe stellt, und anderen Agenten (Zulieferern), die diese Aufgabe übernehmen könnten, vermittelt. Gängige Ansätze für den verteilten Ansatz sind sogenannte Bekanntschaftsnetze (engl.: *acquaintance network*) oder Vetragsnetze (engl.: *contract net*). Innerhalb von Bekanntschaftsnetzen besitzen die einzelnen Agenten Wissen über die Fähigkeiten anderer Agenten. Dadurch wissen sie, welche Aufgaben sie lösen können. Besitzt ein Agent nicht die Fähigkeiten für eine Aufgabe, stellt er Anfragen an Agenten in seinem Netzwerk, die dazu fähig sind. Hat einer dieser angefragten Agen-

ten freie Kapazitäten, wird er die Aufgabe annehmen. Dieser Vorgang wird direkte Zuweisung genannt. Darüber hinaus gibt es die Möglichkeit der Delegation. Findet der aufgabenvergebende Agent keinen Agenten in seinem Netzwerk mit den notwendigen Fähigkeiten, kann er diese Aufgabe an einen oder mehrere Agenten delegieren. Die Delegierten suchen wiederum in ihrem Bekanntschaftsnetz nach fähigen Agenten. Durch diesen Mechanismus können Auftraggeber und -nehmer zusammengeführt werden, die sich nicht direkt kennen. In sogenannten Vertragsnetzen erfolgt die Vergabe von Aufgaben nach dem folgenden Muster. Der sogenannte Manager verteilt eine Ausschreibung mit der zu leistenden Aufgabe an alle Agenten in seinem Vertragsnetz. Die zur Ausführung der Aufgabe fähigen Agenten antworten durch Übermittlung eines Angebots an den Manager. Dieser vergleicht alle Angebote und erteilt einem Agenten den Auftrag. Der letzte Schritt ist die Annahme des Auftrags durch den ausgewählten Agenten, der sich verpflichtet, die Aufgabe zu übernehmen. Diese dynamischen Methoden der Aufgabenallokation erlauben damit, auch aufgaben- oder anforderungsspezifische Teams zu bilden.

Nach der Aufgabendekomposition und -allokation muss die Ausführung der Aufgaben erfolgen. Hierzu ist es in einem MAS essenziell, die Handlungen der einzelnen Agenten zu koordinieren. Diese Koordination hat das Ziel, unnötige oder widersprüchliche Handlungen zu verhindern, Konflikte um Ziele sowie Ressourcen zu vermeiden, Reaktionszeiten zu verkürzen und dadurch die Zusammenarbeit zu optimieren. Für die Koordinierung der Handlungen gibt es verschiedene Ansätze.

Der in der Automatisierung vorherrschende Ansatz ist die direkte Synchronisation. Die Synchronisation der Handlungen findet auf der untersten Ebene zwischen Geräten statt oder durch eine zwischengeschaltete Steuerung, i. d. R. eine Speicherprogrammierbare Steuerung (SPS). Neben den gängigen Fachsprachen für die Programmierung von SPS können auch PC-basierte Ablaufsteuerungen auf Basis von Automaten sowie steuerungstechnisch interpretierte Petri-Netze zum Einsatz kommen [133, 134]. Vorteile der direkten Synchronisation sind schnelle Reaktionszeiten, eine gute Koordination und Konfliktvermeidung sowie eine hohe Anzahl von Agenten, die koordiniert werden können. Die Koordinierung kann dezentralisiert, zwischen einzelnen Agenten, oder zentralisiert, zum Beispiel über eine übergeordnete Prozesssteuerung, stattfinden. Allerdings sind die abgebildeten Prozesse starr, nicht adaptiv und damit fehleranfällig.

Ein anderer Ansatz stammt aus dem Bereich der künstlichen Intelligenz und zielt auf die Koordination durch Planen ab. Die Umsetzung von Planungssystemen für MAS ist aufwendig, ermöglicht jedoch eine optimale und effiziente Koordination. Die Reaktionszeiten des Systems sind größer, aber variable Prozesse können realisiert werden. Durch dynamisches Planen steigen zudem die Adaptivität und die Robustheit des Systems gegenüber Fehlern. Auch hier ist eine zentrale sowie dezentrale Koordinierung möglich. In letzterem Fall müssen die einzelnen Pläne der Agenten zusammengeführt, auf Konflikte untersucht und ggf. auf Basis von Konfliktlösungsstrategien behoben werden. Durch die kombinatorische Problemstellung, die in Planungssystemen für Multiagentensysteme immer besteht, ist die Anzahl der koordinierbaren Agenten wesentlich geringer. Ein weiterer Nachteil dieses Ansatzes sind die aufwendigen Modelle[14], die in jedem Agenten und ggf. zentral hinterlegt werden müssen.

Ein weiterer Ansatz besteht in der Koordination der Agenten auf Basis ihres reaktiven Verhaltens. Hier bestimmt nur das reaktive Verhalten eines Agenten über dessen Handlungen. Die Vorteile sind schnelle Reaktionszeiten und eine hohe Anpassungsfähigkeit. Nachteilig ist, dass ein kaum vorhersehbarer Prozess entsteht, wodurch sich die Koordination verschlechtert und Konfliktsituationen auftreten. Durch diesen Ansatz können MAS mit einer hohen Anzahl von Agenten umgesetzt werden, die auf Basis einfacher reaktiver Verhaltensmuster ein vermeintlich komplexes kollektives Verhalten herausbilden [135].

Abschließend, nach [128], erfolgt die Forschung im Bereich der Multiagentensysteme zwei Zielstellungen. Die erste dient der theoretischen und experimentellen Analyse von Mechanismen zur Selbstorganisation in Umgebungen mit mehreren interagierenden autonomen Agenten. Die zweite Zielstellung zielt auf die Entwicklung verteilter Entitäten mit der Fähigkeit, komplexe Aufgaben durch Kooperation oder Kollaboration gemeinsam zu lösen. Dadurch können MAS als eine Methode oder Technik betrachtet werden, um komplexe Systeme auf Basis von Agenten, Kommunikation, Kooperation und Koordination von Handlungen umzusetzen. Letztere ist insbesondere für die Umsetzung von adaptiven und sich selbst organisierenden Produktionssystemen von großem Vorteil [132].

[14] Diese Modelle werden in Kapitel 4.3.2.1 detailliert behandelt

5.1.2.3 Konzept kognitiver Systeme

In diesem Paradigma [136] werden (kognitive) Architekturen entworfen, die unabhängig von bestimmten Domänen und Aufgabenstellungen sind. Hierzu wird auf die grundlegenden Aspekte menschlicher Kognition der Fokus gelegt, wie Lernen, Schlussfolgern und das Gedächtnis[15] (engl.: *memory*). Kognitive Systeme sind generische Berechnungsmodelle, die für ihre Einsatzumgebung mit Wissen über die Domäne und Aufgaben angereichert werden. Ähnlich dem hybriden Paradigma aus der Robotik implementieren kognitive Systeme reaktive und deliberative Prozesse. Darüber hinaus implementieren kognitive Systeme einen reflektierenden Prozess, die Metakognition. Diese beschäftigt sich mit dem Denken über das Denken, d. h., dieser kognitive Prozess überwacht, verändert und reguliert das reaktive sowie bewusste Verhalten des Systems. Langfristiges Ziel dieser Forschung ist es, eine menschenähnliche Intelligenz in technischen Systemen umzusetzen. Realisierte Architekturen dieses Paradigmas sind zum Beispiel Soar [137] (siehe Abbildung 14). Eingesetzt wurde Soar überwiegend in Simulationsumgebungen, wie Microsoft Robotic Studio, und in Enterprise-Resource-Planning Systemen, aber auch in Robotersystemen, wie dem iRobot Packbot für Katastrophenszenarien, sowie auf der mobilen Plattform Pioneer 2 [137].

5.1.3 Ein Konzept aus der Kybernetik

Cyber-physische Systeme (CPS) sind ein Produkt der Mechatronik und der technischen Kybernetik. Eine Definition des Verbands Deutscher Ingenieure lautet wie folgt:

> *Cyber-Physical Systems (CPS) sind gekennzeichnet durch eine Verknüpfung von realen (physischen) Objekten und Prozessen mit informationsverarbeitenden (virtuellen) Objekten und Prozessen über offene, teilweise globale und jederzeit miteinander verbundene Informationsnetze. [138]*

Als Bestandteil der realen Objekte werden hierin die mechatronischen Systeme (MS) verstanden, die ein Verbund von Mechanik, Elektronik und eingebetteter Software sind, z. B. in Form von einfachen Sensoren und Aktoren oder komplexeren Systemen wie einem Industrieroboter. Einfache CPS sind i. d. R. Bestandteil eines komplexeren CPS, zum

[15] Das Gedächtnis kann in Kurzzeit-, Arbeits- und Langzeitgedächtnis gegliedert werden und jeweils unterschiedliche Arten von Wissen, z.B. deklaratives oder prozedurales Wissen, speichern.

Beispiel eines cyber-physischen Produktionssystems, das aus mehreren Industrierobotern und weiterer Peripherie besteht.

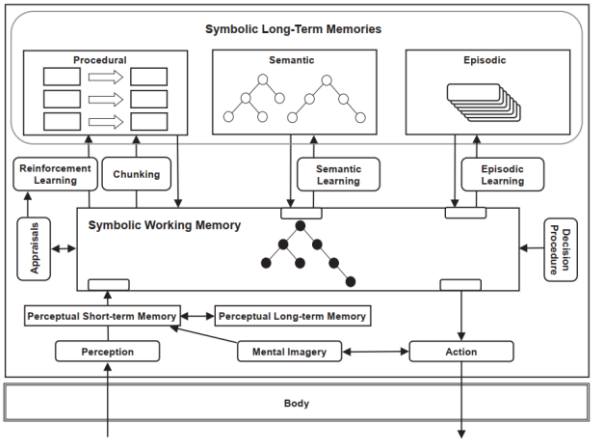

Abbildung 14: Die Architektur von Soar [136].

Im Gegensatz zu normalen MS werden diese jedoch durch offene und teilweise globale Kommunikationsnetzwerke, wie das Internet, mit virtuellen Objekten und Prozessen verbunden. Die virtuellen Objekte sind informationstechnische Abbildungen (semantische Modelle) der realen Objekte (der mechatronischen Systeme oder des Produktionssystems). Sie beschreiben zum Beispiel, um welche Art von System es sich handelt sowie dessen Funktionen und Beziehungen mit anderen Systemen. Darüber hinaus speichern sie Informationen wie Sensordaten. Die Entwicklung solcher standardisierten, offenen semantischen Modelle, die in der Lage sind, die Eigenschaften und Funktionsmerkmale von CPS vollständig zu beschreiben, sind Gegenstand intensiver Forschung [139–141], worunter auch agentenbasierte Modellierungsansätze zu finden sind [142]. Das große Potenzial von CPS liegt in den virtuellen Prozessen. Diese ermöglichen die einfache Einbindung von Steuerungssystemen, Middleware und Applikationsprogrammen, zum Beispiel für die einfache Integration von CPS nach dem Plug-and-Play-Prinzip, Anwendungen für eine bessere Komplexitätsbeherrschung, dezentrale Steuerungskonzepte sowie die Bildung von Sensornetzwerken u. v. m. (siehe [138, 143]). Virtuelle Prozesse können auch im sogenannten Cyber-Raum (engl.: *cyberspace*) umgesetzt werden, d. h. auf IT-Infrastruktur, die über das Internet erreichbar ist, z. B. mit Cloud-Diensten wie Amazons Web Services (AWS) oder Microsofts Azur. So können beispielsweise Applikationsanwendungen wie Zustandsüberwachungssysteme als Dienste realisiert werden, die auf Basis

der gespeicherten oder abrufbaren Sensordaten den Zustand einzelner Systeme oder Prozesse erfassen, vergleichen, Fehler erkennen und Ursachen diagnostizieren. An diesen neuen Geschäftsmodellen zeigt die Industrie derzeit großes Interesse. Die langfristige Vision wird durch Schlagwörter wie ‚intelligente Stromnetze' (Smart Grids), ‚intelligente Städte' (Smart Cities) und ‚intelligente Fabriken'[16] (Smart Factories) [144] beschrieben. Für letzteres ist die Schaffung von hochflexiblen selbstorganisierenden vernetzten Produktionssystemen, die die herkömmlichen festen System-, Organisations- und Domänengrenzen auflösen (siehe Abbildung 15), ein Ziel. Komplexe CPS besitzen dadurch einen Bezug zu den Problemfeldern von Multiagentensystemen.

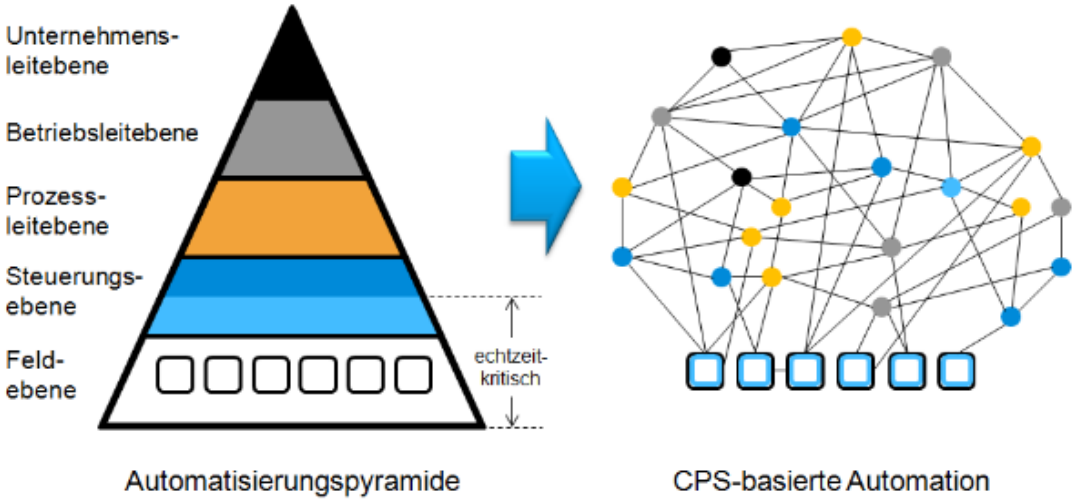

Abbildung 15: Auflösung der Automatiserungspyramide aus [138].

[16] Die intelligente Fabrik ist aktueller Forschungsgegenstand der Forschungsagenda Industrie 4.0 der deutschen Bundesregierung.

Die Kommunikation von CPS baut dabei auf den Ansatz des sogenannten Internets der Dinge (engl.: Internet of Things (IoT)) [145] bzw. dem Internet vom Allem (engl.: Internet of Everything (IoE)) [146] auf. Das IoT kann als Kommunikationsinfrastruktur angesehen werden, die physische Geräte (engl.: *devices*) miteinander verbindet und den Datenaustausch unter diesen ermöglicht. Ursprünglich wurden unter dem Begriff ‚Dinge' diejenigen Geräte verstanden, die zuvor nicht fähig waren, Daten zu erzeugen, zu senden und zu empfangen. Ein interessantes Beispiel hierfür sind intelligente Steckdosen (engl.: Smart Plugs). Der Unterschied zwischen IoT und Maschine-zu-Maschine-Kommunikation (M2M) besteht darin, dass IoT offene Kommunikationsnetzwerke zwischen losen gekoppelten Geräten umsetzt, während M2M meist auf Basis geschlossener und fest gekoppelter Kommunikation zwischen Geräten (Peer to Peer) umgesetzt wurde. Das IoE geht einen bedeutenden Schritt weiter als der IoT-Ansatz: Es soll Menschen, Prozesse, Daten und Dinge miteinander verbinden [146]. Um den höheren Sicherheitsanforderungen bzgl. Cyber-Sicherheit der Industrie gerecht zu werden, hat sich das Industrielle Internet der Dinge (Industrial Internet of Things (IIoT)) vom IoT des Verbrauchermarktes abgegrenzt, wobei beim IIoT auch vom Internet der Dinge und Dienste gesprochen wird, siehe [147].

5.1.4 Zusammenfassung

Wie in diesem Kapitel dargestellt, gibt es vielfältige Systemarchitekturen, die für die Umsetzung von Autonomie in einem technischen System in Frage kommen. Rein reaktive oder deliberative Paradigmen aus dem Bereich der Robotik können, durch die bereits in der Literatur beschriebenen Probleme, für die Umsetzung des RAS ausgeschlossen werden. Die Umsetzung des RAS nach dem hybriden Paradigma der Robotik scheint möglich, wirkt im großen Zusammenhang jedoch ungünstig und würde einen zu roboterzentrierten Blickpunkt darstellen. Außerdem besitzen kommerziell verfügbare Industrieroboterssysteme i. d. R. nicht die notwendigen Voraussetzungen hinsichtlich der Steuerung solch komplexer Prozesse. Weiterhin wäre das entwickelte RAS an die verwendete Hardware gebunden. Aus diesem Grund wird die Umsetzung des RAS in einem externen Steuerungssystem in Software favorisiert. Die Umsetzung des RAS in Form eines rationalen Softwareagenten oder eines kognitiven Systems würde zu einem monolithischen System mit hoher Komplexität führen. Vor diesem Hintergrund erscheint die Umsetzung des RAS auf Basis eines Multiagentensystems oder eines komplexen cyber-physischen

Systems durch den verteilten Systemansatz vorteilhaft, insbesondere hinsichtlich der einfacheren Integration weiterer technischer Systeme in das RAS. Die Implementierung des RAS als komplexes CPS erscheint sinnvoll und zukunftsfähig, allerdings sind die hierzu notwendigen cyber-physischen Teilsysteme noch nicht am Markt verfügbar. Aus diesem Grund wurde für die Umsetzung des RAS eine Architektur in Form eines Multiagentensystems gewählt, auch mit dem langfristigeren Ziel, cyber-physische Teilsysteme sowie komplexe CPS durch die Verknüpfung von mechatronischen Systemen und Softwareagenten auf Basis einer geeigneten Kommunikationsinfrastruktur nach dem Ansatz des IoT bzw. IoE umzusetzen. Die umgesetzte Architektur des RAS wird in Kapitel 6 detailliert beschrieben. Für die Aufgabendekomposition und Koordinierung der Handlungen des Menschen und des RAS wurde der Einsatz von Planungssystemen gewählt, da diese eine bestmögliche Koordination bieten. Die Aufgabendekomposition erfolgt auf Basis automatisierter Demontageplanung. Die Aufgabenallokation wird durch den Menschen vorgenommen und die Koordinierung und Adaption des Prozesses erfolgt durch ein dynamisches Planungssystem. Das folgende Kapitel soll hierzu die notwendigen theoretischen Grundlagen vermitteln und die aus der Literatur bekannten Ansätze darstellen.

5.2 Planungssysteme

Planungssysteme können für die Steuerung von Prozessen verwendet werden, deren Prozessabläufe stark variieren, d. h. in Bereichen, in denen keine festen Abläufe existieren [127]. In diesen Bereichen können Planungssysteme anhand einer Zielbeschreibung, einer Beschreibung des zu steuernden Systems und dessen initialen Zustands, geeignete Prozessablaufe (Pläne) erzeugen, die zur zielgerichteten Steuerung des Systems verwendet werden. Erfolgt die Ausführung der Pläne, ohne deren Effekte zu überwachen, spricht man von einem einfachen Planungssystem. Wird die Ausführung der Pläne überwacht und bei Abweichungen eingeschritten, zum Beispiel durch eine Adaption des Plans, spricht man von dynamischen Planungssystemen (siehe Abbildung 16).

Das Themengebiet des automatisierten Planens ist ein eigenes Forschungsgebiet im Bereich der künstlichen Intelligenz.

> *Planning is the reasoning side of acting. It is an abstract, explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes. This deliberation aims at achieving as best as possible some pre-stated objectives. Automated planning is an area of Artificial Intelligence (AI) that studies this deliberation process computationally. [126]*

Planungssysteme können in domänenspezifische und domänenunspezifische Planungssysteme unterschieden werden. Der Begriff ‚Domäne' meint dabei ein abgetrenntes Anwendungs- oder Wissensgebiet.

Domänenunspezifische Planer (DUP) nutzen standardisierte und generische Wissensrepräsentationen und Wissensverarbeitungstechniken. Ihre Wissensrepräsentation erfolgt durch die Beschreibung der Problemstellung sowie ihrer Anwendungsdomäne. Anhand des Domänenmodells und der Problemstellung kann eine generische Planungsmaschine (engl.: Planning Engine) verwendet werden, um das Planungsproblem zu lösen.

Domänenspezifische Planungssysteme (DSP) nutzen eigene, auf die Problemstellung angepasste Wissensrepräsentationen und Wissensverarbeitungstechniken. Durch ihre Anpassung muss kein zusätzliches Domänenmodell zur Problembeschreibung erzeugt werden, da es in die Wissensverarbeitung integriert wurde. Aufgrund ihrer Spezialisierung sind DSP performanter, haben jedoch einen spezialisierten Anwendungsbereich. Dagegen können DUP für unterschiedlichste Anwendungsbereiche verwendet werden, was

Entwicklungszeit und -kosten reduziert. Der Unterschied zwischen DUP und DSP wird in Abbildung 17 dargestellt.

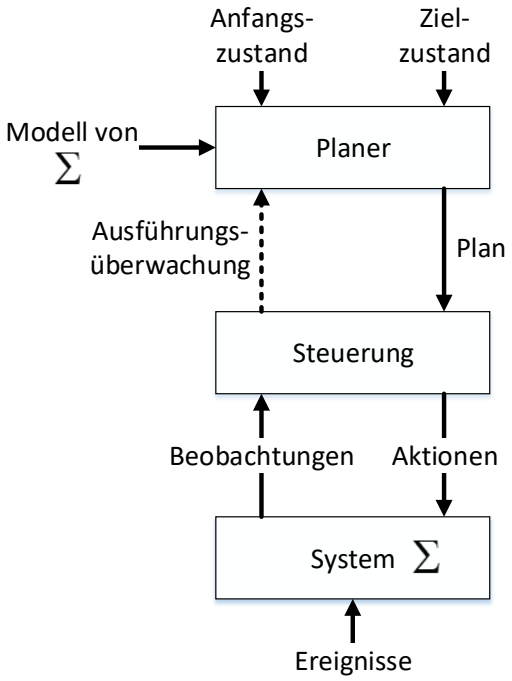

Abbildung 16: Einfaches bzw. dynamisches Planungssystem [127].

Im Rahmen dieser Arbeit wurden ein domänenspezifisches Planungssystem für die automatisierte Demontageplanung sowie ein domänenunspezifisches Planungssystem für die Koordination der Handlungen von Mensch und RAS umgesetzt. Die theoretischen Grundlagen und Ansätze des Planungssystems für die Demontageplanung werden in Kapitel 5.2.3 erörtert und dessen Umsetzung im RAS wird in Kapitel 6.3.1 beschrieben. Die theoretischen Grundlagen für das Planungssystem, das für die Koordination und Adaption der Demontagehandlungen verwendet wird, folgen im Anschluss dieses Kapitels. Die Umsetzung dieses Planungssystems im RAS, mit den notwendigen Anpassungen, wird in Kapitel 6.3.2 dargestellt. Zuvor soll kurz auf generische Planungssysteme eingegangen werden.

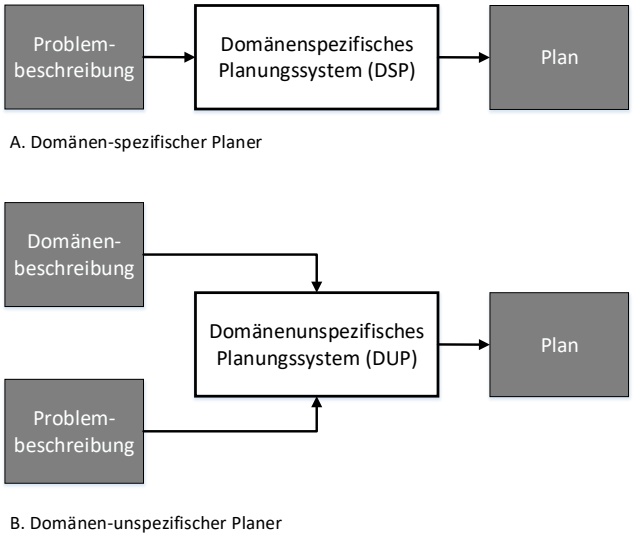

Abbildung 17: Domänenspezifische und -unspezifische Planungssysteme [127].

Die Entwicklung von generischen domänenunspezifischen Planungssystemen wurde seit 1999 durch den jährlich stattfindenden internationalen Planungswettbewerb (International Planning Competition) [148] vorangetrieben. Im Rahmen dieser Veranstaltung wurde auch eine weithin anerkannte Planungssprache, die Planning Domain Definition Language (PDDL) [149] entworfen, die ständig weiterentwickelt wird, um weitere Anforderungen zu erfüllen. Im Bereich der generischen Planungssysteme haben sich spezialisierte Themengebiete gebildet, z. B.:

– deterministisches Planen (engl.: *deterministic planning*)
– nicht deterministisches Planen (engl.: *non-deterministic planning*)
– Planen durch Verfeinerung (engl.: *refinement planning*)
– probabilistisches Planen (engl.: *probabilistic planning*)
– temporales Planen (engl.: *temporal planning*)
– verteiltes und Multiagenten-Planen (engl.: *distributed and multi-agent planning*)

Jedes Themengebiet kann weitere spezialisierte Felder aufweisen. Beispielsweise können im Bereich des deterministischen Planens Planungssysteme für optimale, nicht optimale oder kostenbeschränkte Pläne unterschieden werden und für kontinuierliche sowie diskrete Systeme erfolgen. Im Rahmen dieser Arbeit können die oben genannten Themengebiete nicht vollständig beschrieben werden, daher muss auf entsprechende Literatur verwiesen werden [126, 127, 150, 151]. Die grundlegende Theorie des in dieser Arbeit verwendeten deterministischen Planungssystems soll nun erläutert werden.

5.2.2 Deterministisches Planungssystem

Das deterministische Planen wird auch als klassisches Planen bezeichnet. Die mathematische Grundlage zur Beschreibung eines diskreten Systems Σ erfolgt auf Basis eines vereinfachten Zustandstransitionssystems (engl.: *state transition system*).

5.2.2.1 Zustandstransitionssystem

Das vereinfachte Zustandstransitionssystem wird definiert durch

$$\Sigma = (S, A, \gamma) \tag{1}$$

Darin beschreibt $S = \{s_1, s_2, ..\}$ eine Menge von Zuständen s_i, die das System annehmen kann, und $A = \{a_1, a_2, ..\}$ eine Menge von Aktionen a_i. Die Zustandstransitionsfunktion $\gamma : S \times A \to S$ beschreibt alle möglichen Zustandstransitionen, die durch die Anwendung von Aktionen erfolgen können. Zum Beispiel könnte darin die folgende Zustandstransition $\gamma(s_1, a_1) \to s_2$ definiert sein, die eine Transition vom Zustand s_1 zu s_2 durch Anwendung der Aktion a_1 beschreibt. Wobei eine Aktion a im Zustand s nur anwendbar ist, wenn eine Zustandstransition $\gamma(s, a)$ definiert ist.

Die Vereinfachungen, die in diesem Zustandstransitionssystem getroffen wurden, sind:

- Das System besitzt eine endliche Anzahl von Zuständen.
- Es handelt sich um ein deterministisches System, d. h., der Systemzustand wird durch eine anwendbare Aktion in den erwarteten Nachfolgezustand $\gamma(s, a) \to s'$ überführt.
- Das System ist statisch, d. h., der Zustand des Systems verändert sich nur durch die Ausführung von Aktionen.
- Aktionen haben keine zeitliche Ausdehnung.
- Aktionen können nur nacheinander und nicht parallel ausgeführt werden.

Dieses Zustandstransitionssystem kann grafisch als gerichteter Graph mit Kantenlabels (engl.: *directed labeled graph*) dargestellt werden, wie in Abbildung 18 dargestellt ist.

Ein gerichteter Graph $G = (N, E)$ beschreibt durch seine Knoten (engl.: *nodes*) die möglichen Zustände des Zustandstransitionssystems und durch seine Kanten die möglichen Zustandstransitionen. Jeder Zustand s in S entspricht somit einem Knoten n in N. Eine gerichtete Kante e (engl.: *edge*), definiert durch e = (n_e,n_k,l), zeigt vom Elternknoten n_e zum Kindknoten n_k mit dem Label l und repräsentiert somit eine Zustandstransition $\gamma(s,a) = s'$ vom Zustand s in den Zustand s‘ durch die Aktion a, indem s = n_e, s‘ = n_k und l = a entspricht.

Beispiel: Gegeben sind das Zustandstransitionssystem

$\Sigma = (S, A, \gamma)$ mit $S = \{s_1, s_2, s_3\}$ und $A = \{a_1, a_2, a_3, a_4\}$ sowie die Zustandstransitionen $\{\gamma(s_1, a_1) \to s_2, \gamma(s_2, a_2) \to s_3, \gamma(s_3, a_3) \to s_2, \gamma(s_2, a_4) \to s_1, \gamma(s_3, a_4) \to s_1\}$.

Der entsprechende gerichtete Graph $G = (N, E)$ würde die Knoten $N = \{s_1, s_2, s_3\}$ sowie die Kanten $E = \{(s_1, s_2, a_1), (s_2, s_3, a_2), (s_3, s_2, a_3), (s_2, s_1, a_4), (s_3, s_1, a_4)\}$ besitzen und der Abbildung 18 entsprechen.

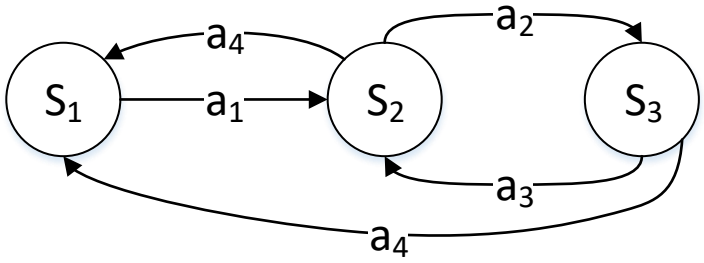

Abbildung 18: Zustandstransitionssystems als gerichteter Graph.

Auf Basis eines solchen Zustandstransitionssystems kann beschrieben werden, wie sich der Zustand eines technischen Systems, zum Beispiel eines Robotergreifers, durch verschiedene Aktionen verändert. So könnte der Zustand s_1 beschreiben, dass der Greifer initial deaktiviert ist. Durch die Aktion a_1 wird er aktiviert und wechselt vom Zustand s_1 in den Zustand s_2, indem die Greiferfinger geöffnet sind. Die Aktion a_2 schließt die Greiferfinger, was durch den Zustand s_3 beschrieben wird. Durch die Aktion a_3 werden die Greiferfinger wieder geöffnet und der Zustand s_2 wird wieder eingenommen. Aus den Zuständen s_2 sowie s_3 kann durch die Aktion a_4 der Greifer wieder deaktiviert und auf den Zustand s_1 zurückgesetzt werden. Für einfache Systeme, die einen kleinen Zustandsraum

besitzen, kann die Beschreibung der einzelnen Zustände und Zustandstransitionen von Hand erfolgen; bei komplexeren Systemen ist dies nicht mehr möglich. Könnte der Greifer zum Beispiel seine Finger in 1-mm-Schritten von 0 bis 85 mm öffnen, müssten 87 einzelne Zustände und 7 397 Transitionen beschrieben werden. Aus diesem Grund werden generative Prozeduren verwendet, die den Zustandsraum und Zustandstransitionen automatisiert herleiten können. Hierzu wird jedoch eine andere Art der Beschreibung (Repräsentation) der Zustände und Aktionen notwendig.

5.2.2.2 Repräsentation von Zuständen und Aktionen in klassischen Planungssystemen

Üblich sind im Bereich des klassischen Planens Repräsentationen auf Basis der Aussagenlogik, der Prädikatenlogik erster Stufe oder die Zustandsvariablen-Repräsentation (engl.: *state variables*). Die Form der Repräsentation beeinflusst die Art, wie Zustände und Aktionen durch eine innere Struktur beschrieben werden. In dieser Arbeit erfolgt die Repräsentation durch Zustandsvariablen, die nachfolgend betrachtet werden. Für eine vollständige Definition der anderen Repräsentationsformen wird auf [126, 127] verwiesen.

Darstellung von Zuständen durch Zustandsvariablen

In dieser Repräsentationsform erhält jeder Zustand s des Zustandstransitionssystems eine innere Struktur, die aus einer Menge von verschiedenen Objekten und deren Eigenschaften besteht. Welche Objekte und welche Eigenschaften der Objekte in einem Zustand beschrieben werden, hängt davon ab, welches Wissen das Planungssystem für seine Funktion benötigt. Bei den Eigenschaften der Objekte wird dabei zwischen statischen und variablen Eigenschaften unterschieden. Variable Eigenschaften eines Objektes können sich durch Aktionen verändern, statische Eigenschaften hingegen bleiben konstant. Die variablen Eigenschaften werden durch sogenannte Zustandsvariablen und statische Eigenschaften durch Relationen beschrieben.

Jedes Objekt wird durch ein eindeutiges Objektsymbol gekennzeichnet und einer Klasse von Objekten zugeordnet. Besteht zum Beispiel eine Roboterzelle aus einem Roboter, einem Greifer, drei Werkstücken sowie einem Tisch, dann könnten die Objektklassen, Roboter D_R, Greifer D_G, Werkstück D_W und Tisch D_T, gebildet werden. Den einzelnen Objekten dieser Klassen könnten eindeutige Symbole zugewiesen werden, wie $D_R = \{r_1\}$,

$D_G = \{g_1\}$, $D_w = \{w_1,w_2,w_3\}$ und $D_T = \{t_1\}$. All diese Objektklassen werden mit mathematischen Konstanten, die benötigt werden, z. B. die Wahrheitswerte D_B = {Wahr, Falsch} in einer Menge D zusammengefasst.

$$D = \bigcup_{i=1}^{k} D_i$$

Um die festen Eigenschaften der Objekte zu beschreiben, werden Relationen zwischen diesen Objekten definiert. Jede feste n-stellige Relation r wird hierzu in der folgenden Form beschrieben: $r = rs(v_1, v_2, ..., v_n)$. Darin ist rs ein eindeutiges Symbol, das die Relation identifiziert und jedes v_i ist ein Objektsymbol aus D. Wäre der Greifer g_1 fest am Roboter r_1 gerüstet, könnte dies durch eine binäre Relation zwischen diesen beiden Objekten in der Form $r_1 = gerüstet(r_1, g_1)$ beschrieben werden. Alle festen Relationen werden in der Menge R zusammengefasst.

$$R = \{r_1, r_2, ..., r_n\}$$

Veränderliche Eigenschaften der Objekte werden durch Zustandsvariablen beschrieben. Jede k-stellige Zustandsvariable x wird hierzu in der Form $x = sv(v_1, v_2, ..., v_k)$ beschrieben. Darin kennzeichnet sv ein eindeutiges Symbol, das die Zustandsvariable identifiziert und jedes v_i bildet ein Objektsymbol aus D ab. Jeder Zustandsvariablen wird zudem eine Domäne zugeordnet. Die Domäne einer Zustandsvariablen beschreibt die Menge von Ausprägungen, die sie annehmen kann. Die Domäne einer Zustandsvariablen ist immer eine Teilmenge von D. Zum Beispiel kann eine Zustandsvariable x_1 definiert werden, die beschreibt, dass ein Roboter r ausgerüstet mit einem Greifer g unterschiedliche Werkstücke gegriffen haben kann. Hierzu wird eine Zustandsvariable in der Form $x_1 = \text{gegriffen}(r, g \,|\, r \in D_R, g \in D_G)$ definiert und ihr die Domäne D_W zugeordnet. Soll weiterhin beschrieben werden, ob ein Werkstück w auf dem Tisch t liegt oder nicht, kann eine Zustandsvariable nach der Form $x_2 = auf(t, w \,|\, t \in D_T, w \in D_W)$ definiert werden und ihr kann die Domäne D_B der Wahrheitswerte zugeordnet werden. Alle definierten Zustandsvariablen werden in der Menge X zusammengefasst.

$$X = \{x_1, x_2, ..., x_n\}$$

Ein konkreter Zustand s der Umgebung wird beschrieben, indem für jede Zustandsvariable x_i in X die möglichen variablen Kombinationen erzeugt und ihnen ein entsprechender Wert z_i ihrer Domäne zugewiesen wird. Ein Zustand s besitzt dann die folgende Form:

$$s = \{(x_1, z_1), (x_2, z_2), ..., (x_n, z_n)\} \text{ bzw. } s = \{x_1 = z_1, x_2 = z_2, ..., x_n = z_n\}$$

Zum Beispiel könnte der initiale Zustand s_0 der Umgebung wie folgt sein:

$$s_0 = \{gegriffen(r_1, g_1) = w_1, auf(t_1, w_1) = Falsch, auf(t_1, w_2) = Wahr, auf(t_1, w_3) = Wahr\}.$$

Damit wird beschrieben, dass der Roboter r_1, gerüstet mit dem Greifer g_1, das Werkstück w_1 gegriffen hat und dass die Werkstücke w_2 und w_3 auf dem Tisch t_1 liegen.

Der Zustandsraum S beschreibt dann alle möglichen variablen Belegungen und Wertezuordnungen der Zustandsvariablen x in X und kennzeichnet somit alle Zustände der Umgebung. Um den Zustand zu verändern, müssen Aktionen bzw. Aktionsvorlagen definiert werden.

Darstellung von Aktionsvorlagen und Aktionen

Zur Beschreibung von Aktionen werden sogenannte Aktionsvorlagen[17] definiert. Die Vorlage einer Aktion in der Zustandsvariablen-Repräsentation ist ein Tupel der Form:

$$\alpha = (head(\alpha), \text{pre}(\alpha), \text{eff}(\alpha), \text{cost}(a)).$$

Wobei $head(\alpha)$ ein Ausdruck folgender Form ist:

$$\text{act}(z_1, z_2, ..., z_k).$$

In dem Ausdruck ist *act* das eindeutige Symbol, das die Aktion repräsentiert. Die Variablen z_i kennzeichnen Objekte aus der Menge D und werden auch als Parameter der Aktion bezeichnet. Im Kopf (*head*) einer Aktion müssen alle Variablen, die in $\text{pre}(\alpha)$ oder $\text{eff}(\alpha)$ vorkommen, definiert sein. Durch $\text{pre}(\alpha) = \{p_1, p_2, ..., p_m\}$ werden die Vorbedingungen einer Aktion beschrieben.

[17] Die Aktionsvorlage besitzt im engl. Sprachgebrauch verschieden Bezeichnungen: action models, action template, action schema, oder planning operator.

Jedes Element p seiner Menge beschreibt ein sogenanntes Literal, das entweder eine Relation r, z. B. $gerüstet(r,g)$, oder eine Zustandsvariable x, z. B. $\text{gegriffen}(r,g)=w$, abbildet.

Der Ausdruck $\text{eff}(\alpha)=\{e_1,e_2,...,e_n\}$ beschreibt den Effekt einer Aktion. Jedes Element e dieser Menge besitzt einen Ausdruck der Form $sv(v_1,v_2,...,v_j)\leftarrow z$, der einer Zustandsvariablen x einen neuen Wert seiner Domäne zuweist, z. B. $\text{gegriffen}(r,g)\leftarrow w$.

Weiterhin können einer Aktion mit $\text{cost}(a)$ die Kosten c zugeordnet werden. Die Kosten einer Aktion müssen immer größer als null sein. Wie später gezeigt wird, sind Aktionskosten entscheidend, um optimale Pläne zu ermitteln.

Kompakt dargestellt besitzt eine Aktionsvorlage die folgende Form:

$\text{act}(z_1,z_2,...,z_k)$
pre : $p_1, p_2,..., p_m,$
eff : $e_1,e_2,...,e_n$
cost : c

Zwei konkrete Aktionsvorlagen α_1,α_2 für das Beispiel können folgende Ausprägung besitzen. Die Aktionsvorlage α_1 beschreibt das Aufheben eines Werkstücks w. Als Vorbedingung muss der Roboter r mit einem Greifer g gerüstet sein, darf kein Werkstück gegriffen haben und das Werkstück muss auf dem Tisch t liegen. Der Effekt der Aktion ist, dass der Roboter r und Greifer g das Werkstück gegriffen haben und dieses somit nicht mehr auf dem Tisch t liegt.

$aufheben(r,g,w,t)$
pre : $gerüstet(r,g), auf(t,w)=Wahr, gegriffen(r,g)=nil$
eff : $auf(t,w)=Falsch, gegriffen(r,g)=w$
cost : 2

Die Aktionsvorlage α_2 beschreibt den umgekehrten Fall, das Ablegen eines gegriffenen Werkstücks w. Als Vorbedingung muss der Roboter r mit einem Greifer g gerüstet sein und ein Werkstück w gegriffen haben. Der Effekt der Aktion ist, dass der Roboter r und Greifer g das Werkstück ablegen und dieses somit wieder auf dem Tisch t liegt.

$ablegen(r, g, w, t)$
pre : $gerüstet(r, g), gegriffen(r, g) = w$
eff : $auf(t, w) = Wahr, gegriffen(r, g) = nil$
cost : 2

Alle Aktionsvorlagen α werden in einer Menge $\mathcal{A}$ zusammengefasst. Damit kann nun die Planungsdomäne definiert werden.

5.2.2.3 Definition einer Planungsdomäne und eines Planungsproblems

Mit den Definitionen des letzten Kapitels kann nun die Planungsdomäne als folgendes Triple dargestellt werden:

$$\Sigma = (S, \mathcal{A}, \gamma)$$

Ein Planungsproblem in der Zustandsvariablen-Repräsentation liegt in der Form

$\mathcal{P} = (\Sigma, s_0, g)$ jedoch mit $\Sigma = (S, A, \gamma)$ vor.

Darin beschreibt s_0 den initialen Zustand der Umgebung und g den Zielzustand der Umgebung, z. B., dass der Roboter anstatt des Werkstücks w_1 das Werkstück w_2 gegriffen hat.

$$g = \{gegriffen(r_1, g_1) = w_2, auf(t_1, w_1) = Wahr, auf(t_1, w_2) = Falsch, auf(t_1, w_3) = Wahr\}$$

Die Menge A beschreibt darin Aktionen a, die durch eine Variablenzuordnung auf Basis der Aktionsvorlagen α definiert in $\mathcal{A}$ abgeleitet werden können, d. h. den Aktionsparameter in den Aktionsvorlagen wurden konkrete Objekte zugeordnet, zum Beispiel $r = r_1$. Beispielsweise könnten damit die beiden folgenden Aktionen a_1 und a_2 gebildet werden, die das Ablegen des Werkstücks w_1 und das Aufheben des Werkstücks w_2 beschreiben.

$ablegen(r_1, g_1, w_1, t_1)$
pre : $gerüstet(r_1, g_1), gegriffen(r_1, g_1) = w_1$
eff : $auf(t_1, w_1) = Wahr, gegriffen(r_1, g_1) = nil$
cost : 2

$aufheben(r_1, g_1, w_2, t_1)$
pre : $gerüstet(r_1, g_1), auf(t_1, w_2) = Wahr, gegriffen(r_1, g_1) = nil$
eff : $auf(t_1, w_2) = Falsch, gegriffen(r_1, g_1) = w_2$
cost : 2

5.2.2.4 Beschreibung der Anwendbarkeit und des Effekts einer Aktion

Die Vorbedingungen einer Aktion bestimmen, ob eine Aktion im momentanen Zustand s anwendbar ist oder nicht. Eine Aktion a gilt als anwendbar im Zustand s, wenn dessen Voraussetzungen definiert in $\text{pre}(a)$ im Zustand s erfüllt sind.

$$anwendbar(s,a) = \begin{cases} Wahr, \text{ wenn } \forall p \in pre(a) \text{ erfüllt in } s \text{ sind} \\ Falsch, \text{ andernfalls} \end{cases}$$

So ist etwa die Aktion a_1 (ablegen) im initialen Zustand s_0 anwendbar, jedoch nicht die Aktion a_2 (aufnehmen).

Die Anwendung einer Aktion a auf einen Zustand s erzeugt einen neuen Zustand s‘ durch die Transition $\gamma(s,a) = s'$ mit

$$s' = \{(x,w) | \text{eff}(a) \text{ enthält den Effekt x} \leftarrow \text{w}\} \cup \{(x,w) \in s | \text{eff}(a) \text{ keinen Effekt für } x \text{ beschreibt}\}$$

Wird zum Beispiel die Aktion a_1 (ablegen) auf den initialen Zustand s_0 angewendet, entsteht über die Transition $\gamma(s_0, a_1) = s_1$ der neue Zustand s_1 mit

$$s_1 = \{gegriffen(r_1, g_1) = nil, auf(t_1, w_1) = Wahr, auf(t_1, w_2) = Wahr, auf(t_1, w_3) = Wahr\}$$

Auf diesen neuen Zustand ist nun die Aktion a_2 (aufnehmen) anwendbar, jedoch nicht mehr die Aktion a_1 (ablegen). Wird die Aktion a_2 (aufnehmen) auf diesen neuen Zustand s_1 angewendet, entsteht über die Transition $\gamma(s_1, a_2) = s_2$ der neue Zustand s_2, der dem Zielzustand g definiert im Planungsproblem entspricht.

$$s_2 = \{gegriffen(r_1, g_1) = w_2, auf(t_1, w_1) = Wahr, auf(t_1, w_2) = Falsch, auf(t_1, w_3) = Wahr\}$$

Dies kann durch eine Zielprüfung nach folgender Form erfolgen:

$$erfüllt(g,s) = \begin{cases} Wahr, \text{ wenn } \forall(\text{x,z}) \in \text{g auch in s definiert sind} \\ Falsch, \text{ andernfalls} \end{cases}$$

Die Aktionsfolge a_1, a_2 beschreibt damit eine Lösung, d. h. einen Plan, für das Planungsproblem $\mathcal{P} = (\Sigma, s_0, g)$. Wie Pläne in klassischen Planungssystem definiert werden, soll im Folgenden beschrieben werden.

5.2.2.5 Definition von Plänen

Ein Plan Π wird als eine Sequenz von Aktionen a in der folgenden Form definiert:

$$\Pi = \langle a_1, a_2, ..., a_k \rangle \quad (2)$$

Eine Sequenz beschreibt eine feste Abfolge, d. h., die Aktion a_1 muss vor Aktion a_2 usw. ausgeführt werden. Die Anzahl von Aktionen a in einem Plan definiert dessen Länge $k = |\Pi|$. Ein Plan ohne Aktionen wird als leerer Plan $\Pi = \langle\ \rangle$ definiert. Wir definieren weiterhin folgende mathematischen Operatoren für Pläne:

Für die beiden Pläne $\Pi = \langle a_1, a_2, ..., a_n \rangle$, $\Pi' = \langle a'_1, a'_2, ..., a'_m \rangle$ und die Aktion a gilt:

$$\Pi + a = \langle a_1, a_2, ..., a_n, a \rangle$$

$$a + \Pi = \langle a, a_1, a_2, ..., a_n \rangle$$

$$\Pi + \Pi' = \langle a_1, a_2, ..., a_n, a'_1, a'_2, ..., a'_m \rangle$$

Ein Plan Π gilt als anwendbar auf den initialen Zustand, wenn jedes $\gamma(s_{k-1}, a_k)$ in der folgenden Formel für alle k definiert ist:

$$\hat{\gamma}(s_0, \Pi) = \hat{\gamma}(\gamma(s_0, a_1), \langle a_2, ..., a_k \rangle)$$

Ein Plan wird als Lösung für ein Planungsproblem $\mathcal{P} = (\Sigma, s_0, g)$ bezeichnet, wenn er angewendet auf den initialen Zustand zum Zielzustand führt: $\hat{\gamma}(s_0, \Pi) = \langle s_0, s_1, ..., g \rangle$.

Die zur automatisierten Plangenerierung verwendeten Algorithmen werden nachfolgend dargestellt.

5.2.2.6 Algorithmen zur Plangenerierung

Die Algorithmen zur Plangenerierung basieren auf klassischen Suchverfahren der künstlichen Intelligenz. Ein wichtiges Unterteilungsmerkmal für Suchverfahren ist die

Einteilung in uninformierte und informierte Suchverfahren. Die uniformierten Suchverfahren sind:

- Breitensuche (*breadth-first search*)
- Tiefensuche (*depth-first search*)
- tiefenbeschränkte Suche (*depth-limited search*)
- iterative vertiefende Tiefensuche (*iterative deepening depth-first search*)
- bidirektionale Suche (*bidirectional search*)
- Suche mit einheitlichen Kosten (*uniformed cost search*)

Die uninformierten Suchverfahren besitzen kein zusätzliches Wissen über den kürzesten Weg zum Zielzustand, daher werden diese Verfahren oft als ‚blinde Suche' bezeichnet. Im Gegensatz bieten die informierten Suchverfahren Heuristiken, die den Weg zum Ziel abschätzen und so die Expansion des Suchraumes in Richtung des Zielzustands beschleunigen. Die in der Literatur bekannten informierten Suchalgorithmen sind:

- Besten-Suche (*greedy/best-first/hill climbing search*)
- rekursive Besten-Suche (*recursive best-first search*)
- A*-Suche (*A* search*)
- gewichtete A*-Suche (*weighted A* search*)

Die Leistungsfähigkeit der informierten Suchverfahren wird durch die verwendeten Heuristiken bestimmt. Aus diesem Grund folgt nach der Darstellung der Suchverfahren ein Abschnitt, der die verschiedenen Möglichkeiten, eine Heuristik zu definieren, aufzeigt. Umgesetzt und verwendet wird in dieser Arbeit die Suche mit einheitlichen Kosten. Nur diese soll detailliert beschrieben werden. Für Informationen zu den anderen genannten Suchverfahren wird auf die Literatur [126, 127, 150–152] verwiesen. Bevor das Suchverfahren erläutert wird, muss eine effiziente Datenstruktur für die Suche beschrieben werden.

Datenstruktur für Suchalgorithmen

Die Darstellung des Suchraumes kann grundsätzlich durch Bäume oder Graphen erfolgen. In dieser Arbeit wird die Darstellung in Suchbäumen verwendet. Ein Suchbaum (*search tree*) Γ besteht aus Knoten (*nodes*) N und gerichteten Kanten (*edges*) E:

$$\Gamma = (N, E)$$

Ein Suchbaum besitzt immer einen Wurzelknoten (*root node*). Aus dem Wurzelknoten entspringen die Äste mit Blattknoten. Bis auf den Wurzelknoten besitzt jeder Blattknoten einen Elternknoten (*parent node*), wobei die Kanten des Suchbaumes vom Elternknoten zum Kindknoten (*child node*) zeigen. Am Ende eines jeden Astes befindet sich das Grenzblatt (*leaf node*). Jeder Blattknoten in diesem Suchbaum besitzt die folgende interne Struktur:

– n.State beschreibt den Zustand des Systems durch Zustandsvariablen.
– n.Parent ist der vorherige Blattknoten im Ast (Elternblattknoten).
– n.Action ist die Aktion, durch die dieser Blattknoten erzeugt wurde.
– n.PathCost sind die akkumulierten Pfadkosten bis zu diesem Blattknoten.
– N.EvalCost sind die geschätzten Kosten bis zum Ziel (nur in der informierten Suche).
– N.Depth kennzeichnet die Tiefe des Knotens.

Die Generierung des Suchbaumes findet durch die schrittweise Expansion der Grenzknoten, beginnend am Wurzelknoten, statt. Um die Expansion des Suchbaumes zu steuern, kommt eine Warteschlange, die als ‚offene Liste' (oList) bezeichnet wird, zum Einsatz. In dieser werden die Grenzknoten des Suchbaumes gespeichert. Die Art der Warteschlange bestimmt das Verhalten des Algorithmus bei der Expansion des Suchraumes. Die drei vorkommenden Arten von Warteschlangen werden nachfolgend kurz beschrieben:

– FIFO (First in, First out): Die Reihenfolge vom Hinzufügen (*enqueue*) und Entnehmen (*dequeue*) von Elementen bleibt gleich; es ist die klassische Form einer Warteschlange (engl.: *queue*).
– LIFO (Last in, First out): Die Reihenfolge vom Hinzufügen (*push*) und Entnehmen (*pop*) von Elementen wird umgedreht, man spricht von einem ‚Stapel' (engl.: *stack*).
– LOFO (Lowest in, First out): Eine priorisierte Warteschlange ordnet nach dem Hinzufügen (*insert*) die Elemente nach ihrem Wert in die Warteschlange. Entnommen (extractMin) wird das Element mit dem niedrigsten Wert. Man spricht in diesem Fall auch von einer ‚Vorrangwarteschlange' (engl.: *heap*).

Weiterhin existiert eine Menge, die als ‚geschlossene Liste' (gList) bezeichnet wird, in der alle bereits expandierten Blattknoten hinzugefügt werden.

Suche mit einheitlichen Kosten

Die Suche mit einheitlichen Kosten generiert einen Suchbaum nach dem in Algorithmus 1 beschriebenen Verfahren. Der Wurzelknoten wird mit dem initialen Zustand s_0 initialisiert (Zeile 1) und zur priorisierten LOFO-Warteschlange hinzugefügt (Zeile 2). Der Algorithmus bleibt in der Schleife (Zeile 3), solange Grenzknoten in der Warteschlange existieren. Wenn sich keine Grenzknoten in der Warteschlange befinden, wurde der Suchbaum vollständig generiert und der Zielzustand wurde nicht gefunden. Somit konnte keine Lösung ermittelt werden; der Algorithmus gibt in diesem Fall einen Fehler zurück (Zeile 12). Die Grenzknoten werden anhand ihrer kumulierten Pfadkosten, vom Wurzelknoten bis zum Knoten n, in der Warteschlange sortiert. Der Grenzknoten mit den niedrigsten Pfadkosten $g(n)$ wird jeweils der Warteschleife entnommen (Zeile 4) und direkt der geschlossenen Liste hinzugefügt (Zeile 5). Dadurch expandiert der Algorithmus immer das Grenzblatt mit den geringsten Kosten. Der Algorithmus ignoriert damit Teile des Baumes mit hohen Kosten. Im Sonderfall, wenn alle Aktionen die gleichen Kosten besitzen, verhält sich der Algorithmus wie die Breitensuche. Danach, in Zeile (Zeile 6), wird überprüft, ob der Zustand des entnommenen Grenzknotens dem des Zielzustandes entspricht. Ist dies der Fall, wird die Aktionsfolge ausgehend vom Zielknoten nach Funktion 1 ermittelt und als Lösung des Suchproblems ausgegeben (Zeile (7)). Entspricht der Grenzknoten nicht dem Ziel, werden in Zeile (8) alle Aktionen ermittelt, die in diesem Grenzblatt anwendbar sind, d. h. deren Vorbedingungen im Grenzblatt-Zustand vollständig erfüllt sind. Im folgenden Schritt (Zeile 9) werden die ermittelten Aktionen auf den Grenzknoten angewendet, wodurch |A‘| neue Kindknoten bzw. Grenzblätter entstehen, siehe Funktion 2. Nicht alle dieser erzeugten Grenzknoten werden der Warteschlange hinzugefügt (siehe Zeile 11), sondern nur solche, deren Knotenzustand nicht bereits in einem Grenzknoten der Warteschlange oder in der geschlossenen Liste vorkommt (siehe Zeile 10). Diese Maßnahme soll verhindern, dass ein bereits bekannter Grenzknotenzustand mehrmals im Suchbaum vorkommt und/oder mehr als einmal expandiert wird. Eine Optimierung dieses Algorithmus kann durch die in Zeile (**) dargestellte Überprüfung stattfinden. Der Zustand der neu erzeugten Grenzblätter wird hierzu mit dem Zielzustand verglichen. Liegt ein Knoten mit dem Zielzustand vor, wird der Plan durch Funktion 1 erzeugt und ausgegeben.

Algorithmus: Suche mit einheitlichen Kosten
Eingabe: A, s_0, g
Ausgabe: Π oder Fehler

(1) n_0.State $\leftarrow s_0, n_0$.Parent $\leftarrow null, n_0$.Action $\leftarrow null, n_0$.PathCost $\leftarrow 0$
(2) oList.insert(n_0), gList $\leftarrow \emptyset$
(3) **while** oList $\neq \emptyset$
(4) $n \leftarrow$ oList.extractMin()
(5) gList $\leftarrow$ gList $\cup \{n\}$
(6) **if** $n.State$ entspricht g
(7) **return** $\Pi \leftarrow$ Aktionsfolge(n)
(8) $A' \leftarrow \{\forall a \in A \mid$ pre(a) ist in n.State erfüllt $\}$
(9) $N' \leftarrow$ expandiereKnoten(n, A')
(10) $N^* \leftarrow \{\forall n' \in N' \mid \neg\exists x \in$ gList $\cup$ oList: x.State = n'.State$\}$
(**) **if** $\{\exists n' \in N' \mid n'$.State entspricht $g\}$ **return** $\Pi \leftarrow$ Aktionsfolge(n')
(11) oList.insert(N^*)
(12) **return** Fehler

Algorithmus 1: Suche mit einheitlichen Kosten.

Die Ermittlung der Aktionsfolge (siehe Funktion 1) vom Wurzelknoten bis zum (Ziel-)Blattknoten ist durch die interne Struktur eines Knotens in Suchbaum trivial. Ausgehend vom letzten (Ziel-)Blattknoten erfolgen die Bewegungen über die Elternknoten (Zeile 6) zurück bis zum Wurzelknoten über eine Schleife (Zeile 2). Bei der Rückwärtsbewegung werden die Aktionen dem anfangs leeren Plan (Zeile 1) vorne angefügt (Zeile 5). Rückgabewert der Funktion ist ein vollständig geordneter Plan (Zeile 7).

```
Funktion: Aktionsfolge
Eingabe: n'
Ausgabe: Π

(1)  Π ← ⟨ ⟩
(2)  while n' ≠ null
(5)        Π ← n'.Action + Π
(6)        n' ← n'.Parent
(7)  return Π
```

Funktion 1:Ermittlung der Aktionsfolge.

Die Expansion eines Knotens erfolgt wie in Funktion 2 dargestellt. Für jede anwendbare Aktion wird ein neuer Blattknoten generiert (Zeile 2). Dieser erhält den durch die Aktion veränderten Zustand des Elternknotens (Zeile 3).

```
Funktion: expandiereKnoten
Eingabe: n, A'
Ausgabe: N'

(1)  N' ← ∅
(2)  foreach a ∈ A'
(3)        n'.State ← γ(n.State, a)
(4)        n'.Parent ← n
(5)        n'.Action ← a
(6)        n'.PathCost ← n.PathCost+cost(a)
(7)        n'.Depth ← n.Depth +1
(8)        N' ← N' ∪ {n'}
(9)  return N'
```

Funktion 2: Expansion eines Grenzknotens.

Weiterhin wird in dem neuen Blattknoten eine Referenz zum Elternknoten (Zeile 4) sowie zur Aktion gespeichert (Zeile 5). Auch die Kosten (Zeile 6) und die Tiefe des neuen Knotens werden aktualisiert (Zeile 7). Jeder durch die Expansion entstehender Blattknoten wird in der Menge N‘ gespeichert (Zeile 8) und am Ende als Rückgabewert übergeben (Zeile 9).

In Abbildung 19 ist dargestellt, wie die Expansion des Suchraumes bei der Suche mit einheitlichen Kosten stattfindet. Der rote Knoten stellt den Wurzelknoten und damit den

initialen Zustand s_0 dar. Der grüne Knoten beschreibt den Zielzustand g. Kanten beschreiben Aktionen und die Zahl neben einem Knoten dessen kumulierte Kosten. Graue Knoten kennzeichnen Knoten, die bei der Suche expandiert wurden. Durch Strichpunkt gekennzeichnete Kanten und Knoten wurden bei der Suche nicht erzeugt.

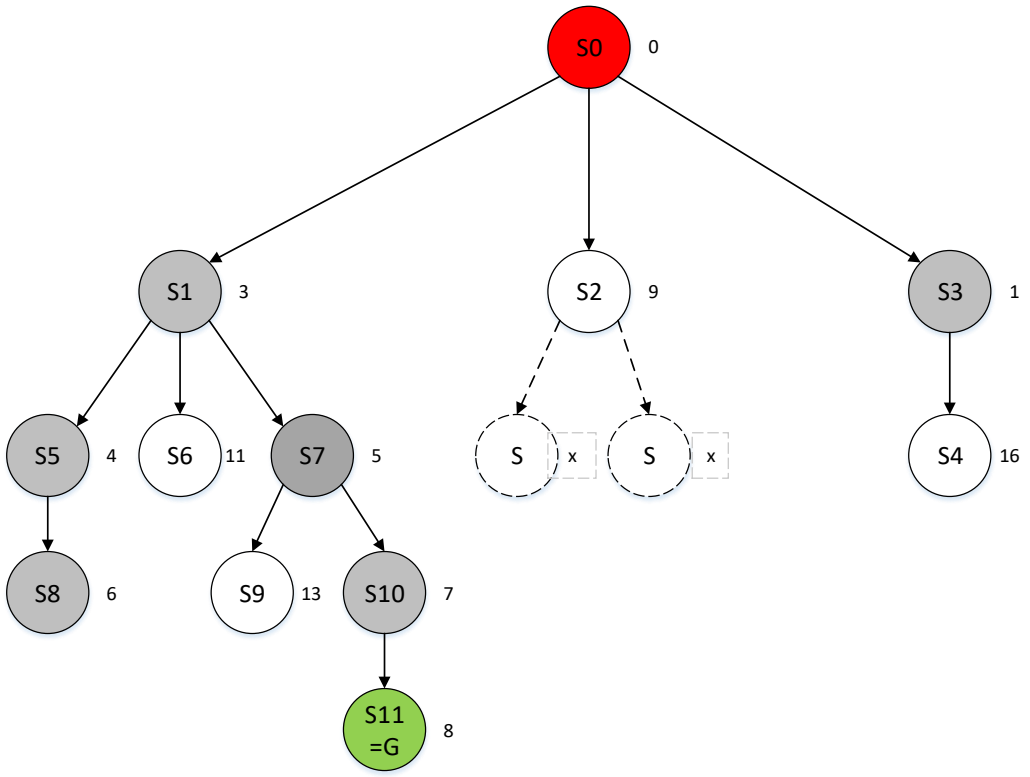

Abbildung 19: Durch den Plannungsalgorithmus erzeugter Suchbaum.

Vergleich der Suchverfahren

Der Vergleich der Suchverfahren soll anhand der folgenden Eigenschaften stattfinden.

- Vollständigkeit: Nachweis, dass eine Lösung, falls sie existiert, gefunden wird
- Optimalität: Nachweis, ob die ermittelte Lösung anhand der gewählten Metrik optimal ist
- Zeitkomplexität: Laufzeitverhalten des Algorithmus
- Speicherkomplexität: Speichernutzung des Algorithmus

Mithilfe dieser Eigenschaften sollen die Suchalgorithmen der uninformierten sowie informierten Suche bewertet werden. Wie in Tabelle 2 und Tabelle 3 zu erkennen, ist die Komplexität der Suchalgorithmen grundsätzlich abhängig vom Verzweigungsfaktor (engl.: *branching factor*) b. Weiterhin haben, je nach Verfahren, die Tiefe d der ersten

Lösung, die maximale Tiefe des Suchraumes m oder die Tiefenbeschränkung l Einfluss auf die Komplexität. Bei der Suche mit einheitlichen Kosten kann die Komplexität anhand der Pfadkosten g der optimalen Lösung und anhand der kleinsten Aktionskosten **ε** beschrieben werden. Die normale oder tiefenbeschränkte Tiefensuche kann nicht garantieren, dass eine Lösung gefunden wird. In diesem Fall würde der Algorithmus nie terminieren. Die Vollständigkeit der Breitensuche, der iterativ vertiefenden Suche, der bidirektionalen Suche sowie der Suche mit einheitlichen Kosten ist in endlichen Zustandsräumen hingegen gegeben. Werden Aktionen mit unterschiedlichen Kosten verwendet, kann eine optimale Lösung nur durch die Suche mit einheitlichen Kosten ermittelt werden. Die Komplexität der Suche mit einheitlichen Kosten kann jedoch bei Aktionen mit kleinen Kosten wesentlich höher als die der anderen Suchverfahren sein. Der Algorithmus wählt Aktion mit niedrigen Kosten vor Aktionen mit hohen Kosten. Das kann dazu führen, dass ‚nützliche' Aktionen mit hohen Kosten spät expandiert werden.

Tabelle 2: Vergleich der uninformierten Algorithmen [125–127].

Kriterium	**Breiten -suche**	**Tiefen -suche**	**Tiefenbe-schränkte Suche**	**Iterativ vertiefende Suche**	**Bi-direktionale Suche**	**Suche mit einheitlichen Kosten**
Vollständig-keit	Ja	Nein	Nein	Ja	Ja	Ja
Zeitkomplexi-tät	$O(b^d)$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$	$O(b^{1+\lfloor g/\varepsilon \rfloor})$
Speicher-kom-plexität	$O(b^d)$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$	$O(b^{1+\lfloor g/\varepsilon \rfloor})$
Optimal	Ja	Nein	Nein	Ja	Ja	Ja

Die exponentielle Komplexität, die durch die stetige Verzweigung des Suchraumes entsteht, schränkt die Anwendbarkeit von uninformierten Suchverfahren erheblich ein. Die informierten Suchverfahren versuchen, diese Einschränkung zu lockern, indem sie die Verzweigung des Suchbaumes durch Verwendung einer Heuristik reduzieren. Die Heuristik h schätzt, welcher Knoten schneller zum Ziel führt und expandiert werden soll.

Das informierte Suchverfahren der Besten-Suche, wie in Tabelle 3 dargestellt, ist unvollständig und besitzt im ungünstigen Fall eine Komplexität, die mit der maximalen Tiefe m des Suchraumes zusammenhängt. Die A*-Suche ist vollständig und optimal mit einer

geeigneten Heuristik. Im einfachsten Fall ist die Zeitkomplexität des Algorithmus abhängig vom absoluten Fehler $\Delta = h^* - h$, zwischen geschätzten h und wahren Kosten $h*$ der Heuristik. Da der A*-Algorithmus jedoch alle Knoten expandiert, deren Knotenkosten geringer sind als die Kosten der optimalen Lösung, und diese in der offenen und geschlossenen Liste verwaltet, ist die Speicherkomplexität der beschränkende Faktor. Die Speicherkomplexität der A*-Suche kann allerdings durch die speicherbegrenzten Varianten reduziert werden. Durch die gewichtete A*-Suche kann zudem der effektive Verzweigungsfaktor verringert werden. Die Performanz der informierten Suchverfahren wird durch die Heuristik bestimmt.

Tabelle 3: Vergleich der informierten Suchverfahren [125].

Kriterium	**Bestensuche**	**Rekursive Bestensuche**	**A*-Suche**
Vollständigkeit	Nein	Nein	Ja
Optimalität	Nein	Nein	Ja
Zeitkomplexität	$O(b^m)$	$O(b^m)$	$O(b^{\Delta})$
Speicherkomplexität	$O(b^m)$	$O(mb)$	$O(b^{\Delta})$

Heuristiken

Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems. Existiert jedoch kein Algorithmus, mit dem eine Problemstellung effizient gelöst werden kann, können Heuristiken zur Effizienzsteigerung verwendet werden. Eine Heuristik basiert auf vagem Wissen, mit dem trotzdem eine wahrscheinliche Aussage zum Verhalten eines Systems getroffen werden kann. Eine Heuristik verkörpert damit spezifisches Wissen, das zu einer schnelleren Lösungsfindung verwendet werden kann. Durch eine Evaluierungsfunktion wird dieses Wissen in eine problemunabhängige Form, d. h. in eine positive reelle Zahl, überführt. Eine Heuristik beschreibt dadurch eine Abbildung, die jedem Zustand eine Zahl zuweist, die den ‚Abstand' zum Zielzustand wiedergibt.

$$h : S \to \mathbb{R}_0^+ \cup \{\infty\}$$

Der unendliche Wert wird benötigt, da per Definition gilt:

$$h(n_0) = \infty, h(g) = 0 \text{ und } h(n^\dagger) = \infty$$

Wobei $n^{\dagger}$ einen Knoten beschreibt, der keinen Nachfolgeknoten besitzt (Sackgasse).

Für manche Problemarten sind geeignete Heuristiken einfach zu definieren, zum Beispiel für räumliche Probleme wie Karten. Hier kann die Heuristik die direkte euklidische oder die Manhattan-Distanz zwischen zwei Punkten der Karte verwenden. Für andere Problemdomänen ist es erheblich schwieriger, geeignete Heuristiken zu ermitteln. ‚Geeignet' (engl. *admissible*) bezeichnet dabei eine fest definierte Eigenschaft einer Heuristik:

$$h(n) \leq h^*(n),$$

worin $h^*(n)$ die wahren Kosten vom Knoten n zum Ziel sind.

Eine Heuristik darf demnach die wahren Kosten von Knoten n zum Ziel niemals überschätzen. Diese Bedingung muss für alle Knoten n im Suchbaum Γ gelten. Nur wenn eine Heuristik diese Eigenschaft erfüllt, kann sie zur Ermittlung von optimalen Lösungen angewendet werden. Für die Optimalität von A* in der Suche in Graphen muss die Heuristik darüber hinaus konsistent sein. Konsistent ist eine Heuristik, wenn sie folgende Eigenschaft erfüllt:

$$h(n) \leq c(n,a,n') + h(n')$$

Darin beschreibt $c(n,a,n')$ die Kosten vom Knoten n zu n' durch die Aktion a. Damit darf die Heuristik die Summe der Kosten von n nach n' und von n' zum Ziel g nicht überschätzen. Man spricht in diesem Fall auch von ‚optimistischen Heuristiken', weil die wahren Kosten immer höher als die vermuteten sind.

Grundsätzlich gibt es die Möglichkeit, eine Heuristik durch ein sogenanntes gelockertes Problem (*relaxed problem*) zu definieren. Bei einem gelockerten Problem werden alle oder einige der Vorbedingungen von Aktionen entfernt. Wird ein Suchgraph auf Basis dieses gelockerten Problems erzeugt, bildet der Pfad zwischen Start und Ziel den absolut kürzesten Weg. Auch wenn dieser Weg im realen Problem aufgrund der Vorbedingungen der Aktionen nicht möglich ist, genügt er der Definition einer anwendbaren Heuristik, die die Kosten von jedem Knoten n zum Ziel g nicht überschätzt.

Weiterhin können Heuristiken auf Basis bekannter Lösungskosten von Unterproblemen beruhen. Besteht ein Problem aus einem oder mehreren bekannten Unterproblemen, können deren Lösungskosten als Heuristik verwendet werden. Die Unterprobleme und deren

Lösungskosten werden hierzu in Musterdatenbanken (*pattern data bases*) gespeichert. Das Lernen aus Erfahrung ist eine weitere Möglichkeit, um Heuristiken zu erstellen. Hierzu kommen maschinelle Lernverfahren zum Einsatz, die anhand bereits gelöster Probleminstanzen eine Heuristikfunktion ermittelt.

Am Ende dieses Kapitels soll noch darauf hingewiesen werden, dass im Rahmen dieser Arbeit ein Konzept entwickelt wurde, wie ein heuristikähnlicher Effekt, auf Basis variabler Aktionskosten, in der Suche mit einheitlichen Kosten umgesetzt wurde, um die Performanz der Suche deutlich zu erhöhen. Das Konzept wird in Kapitel 6.3.2.3 erläutert.

5.2.3 Automatisierte Demontageplanung

Ein technisches Produkt lässt sich als Anordnung von Bauteilen beschreiben, die untereinander durch Verbindungen verknüpft sind. Werden die Verbindungen zwischen den Bauteilen schrittweise gelöst, wird das Produkt zerlegt. Die Schrittfolge dabei ist nicht beliebig, sondern durch den Aufbau des Produkts vorgegeben. Die computerunterstützte Generierung dieser Demontageschrittfolge wird als automatisierte Demontageplanung bezeichnet und ist Gegenstand jahrzehntelanger intensiver Forschung. Die Demontageplanung erhält auch durch den Assembly-by-Disassembly-Ansatz Aufmerksamkeit aus dem Bereich der Montageplanung. Eine Voraussetzung für die automatisierte Demontageplanung ist ein Modell, das den Produktaufbau in einer maschinell interpretierbaren Form beschreibt. In der Literatur stehen hierzu vielfältige Möglichkeiten zur Verfügung, wobei die Planungsalgorithmen vom gewählten Modell abhängig sind. Im Folgenden soll eine Auswahl von Ansätzen kurz dargestellt werden. Die gefundenen Publikationen [153–165] verwenden für die Beschreibung der Produktstruktur gerichtete und ungerichtete Graphen sowie Hypergraphen (auch ‚Und-oder-Bäume' genannt). Andere Ansätze basieren auf Petri-Netzen [153, 158], Beschreibungslogik [166] sowie objektorientierten Modellen [167]. Neuere Ansätze verwenden Ontologien [166, 168–171]. Die meisten Arbeiten legen den Fokus auf die Ermittlung der Demontageschrittfolge und berücksichtigen damit nicht die weiteren Anforderungen für die Prozesssteuerung. Aus diesem Grund werden in diesen Modellen Demontageprozessinformationen wie die Greifposition eines Bauteils, die für das Robotersystem notwendig sind, nicht berücksichtigt. Die Entwicklung eines eigenen Modells zur Beschreibung war somit notwendig. Nach gründlicher Abwägung wurde ein objektorientierter Ansatz gewählt. Entscheidend waren die Vorteile der einfachen Erweiterbarkeit, der Informationskapselung und der einfachen Implementierung [167]. Entstanden ist ein aussagekräftiges Modell, das den Aufbau eines Produktes detailliert beschreiben kann. Aus dem objektorientierten Modell lassen sich zudem gerichtete Graphen und andere Modelle erzeugen [167]. Das Modell wird in Kapitel 6.3.1 ausführlich erläutert.

6 System- und Softwarearchitektur

In diesem Kapitel wird das Demontagearbeitsplatzsystem und die Architektur des RAS dargestellt werden. Im folgenden Kapitel 6.3 wird ausführlich auf die Umsetzung der einzelnen Softwareagenten eingegangen. Hierzu werden die beiden verwendeten Softwarearchitekturen, deren Module und Informationsverarbeitung sowie Kommunikation detailliert erörtert.

6.1 Demontagearbeitsplatzsystem

Der Mensch, das technische Assistenzsystem und alle anderen am Demontageprozess beteiligten Objekte bilden zusammen das Arbeitsplatzsystem, wie es in Abbildung 20 gezeigt ist.

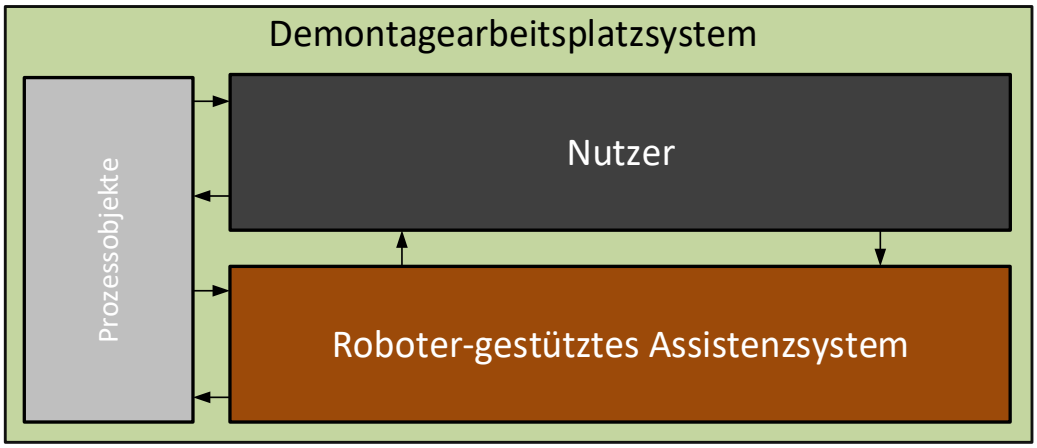

Abbildung 20: Arbeitsplatzsystem.

Der Entwurf des Arbeitsplatzsystems sieht vor, dass lediglich ein Nutzer mit dem technischen Assistenzsystem bei der Demontage von Produkten zusammenwirkt. Die Form der Mensch-Roboter-Interaktion kann dabei zwischen Kooperation und Kollaboration wechseln. Auch die Interaktionsrolle des Menschen wechselt zwischen Supervisor, Kollaborateur und Kooperateur. Als Supervisor bestimmt der Nutzer den Ablauf des Demontageprozesses und die Form der Arbeitsaufteilung bzw. die Form der technischen Unterstützung; bei den einzelnen Demontagetätigkeiten kann er hingegen die Rolle einen Kooperateurs oder Kollaborateurs einnehmen. Unter dem Begriff ‚Prozessobjekte‘ werden alle

am Demontageprozess beteiligten Gegenstände, etwa das zu demontierende Produkt, Handwerkzeuge, Transportkisten usw., zusammengefasst. Alle mit dem Assistenzsystem vernetzten Maschinen und Geräte werden im RAS zusammengefasst.

6.2 Robotergestütztes Assistenzsystem

Wie bereits in Kapitel 5.1.4 angemerkt, soll das RAS nach dem Ansatz verteilter, loser miteinander gekoppelter Systeme im Sinne eines Multiagentensystems umgesetzt werden (siehe Abbildung 21). Ausschlaggebend hierfür ist die schnellere Integration neuer Teilsysteme, die einfachere Modifikation oder Wartung bestehender Systeme und die bessere Beherrschbarkeit der Komplexität durch Zerlegung des Gesamtsystems in Teilsysteme. Der Entwurf des RAS sieht vor, dass jedes technische System mit einem Softwareagenten (gerätesteuernden Agenten (GSA)) verbunden und dadurch im RAS einheitlich repräsentiert wird.

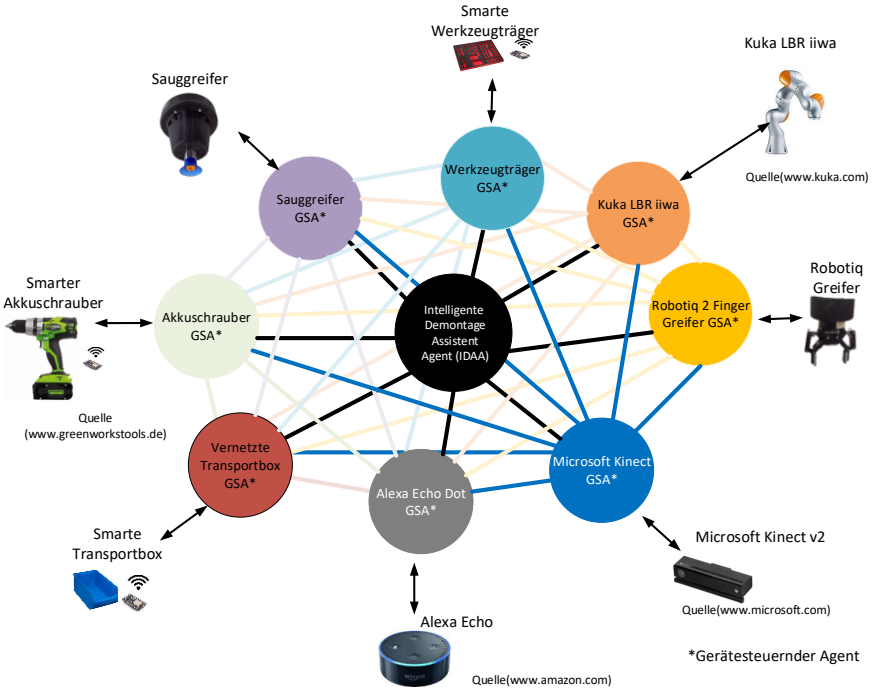

Abbildung 21: Prinzipdarstellung des verteilten, lose gekoppelten RAS.

Ein zweiter Typ von Softwareagenten im RAS, der keine physische Repräsentation besitzt, ist der intelligente Demontage-Assistent-Agent (IDAA). Dieser stellt das zentrale, planende, steuernde, koordinierende sowie überwachende und mit dem Menschen auf höherer Abstraktionsebene kommunizierende System dar. Beide Agententypen werden nun näher anhand der verschiedenen Abstraktionsebenen im RAS erläutert. Die Abstraktionsebenen, die im RAS umgesetzt wurden, sind in Abbildung 22 dargestellt. Um einen Überblick über die Funktionalitäten des Systems zu geben, wird der Gegenstand bzw. Zweck jeder Abstraktionsebene benannt und die dazu relevante Wissensbasis zugeordnet. Auf jede Abstraktionsebene wird in diesem Kapitel kurz Bezug genommen.

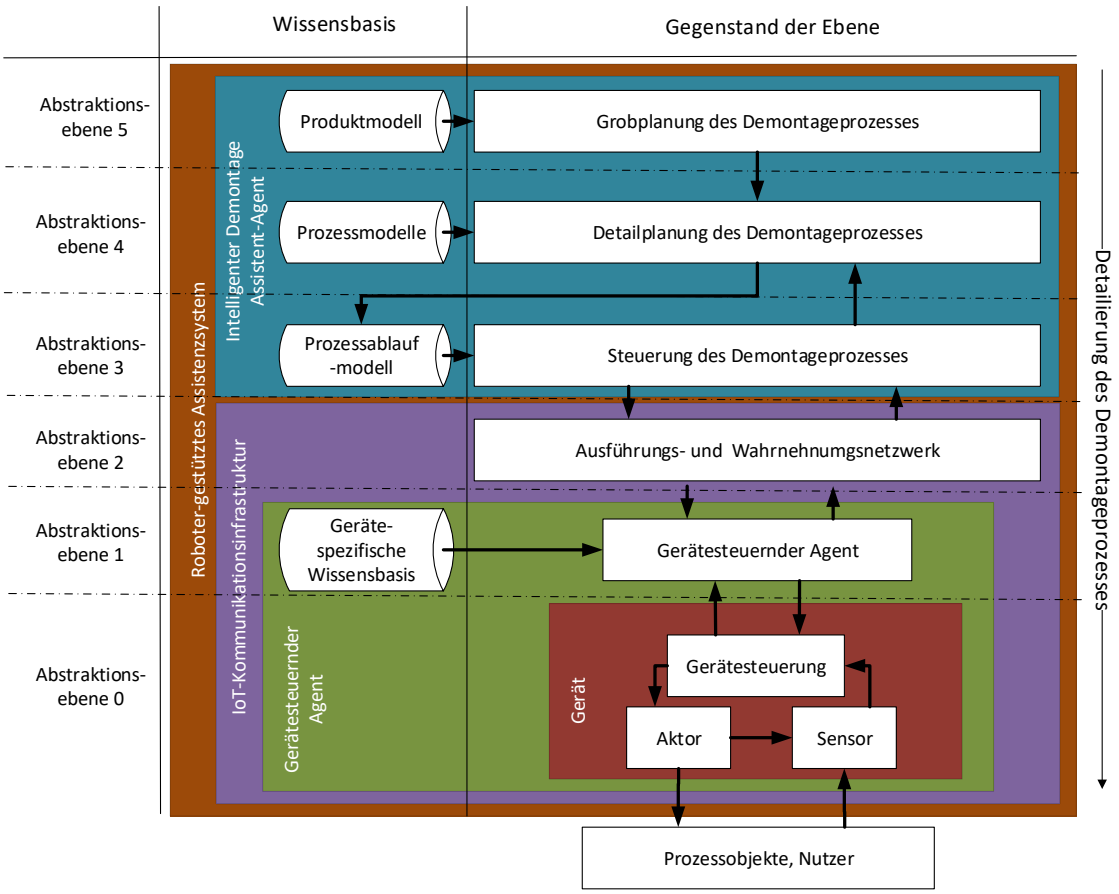

Abbildung 22: Abstraktionsebenen des robotergestützten Assistenzsystems.

Abstraktionsebene 5

Zur Demontage unterschiedlicher Produkte mit variabler Zielstellung sind im IDAA ein Demontageplanungssystem (DPS) und eine Schnittstelle für den Import von Produktmodellen integriert. Durch das DPS erfolgt die Grobplanung des Demontageprozesses. Auf Basis eines Produktmodells[18], das den Aufbau des zu demontierenden Produkts beschreibt, und einer Zielvorgabe des Nutzers, z. B. die Demontage eines bestimmten Bauteils, ermittelt das System die möglichen Demontagereihenfolgen der Bauteile und bildet eine Abfolge von Demontageschritten. Jeder Demontageschritt besteht aus mindestens einer Demontageaufgabe, deren Ziel die Entfernung eines Bauteils aus der Baugruppe ist. Zu jeder Demontageaufgabe ist das zu demontierente Bauteil, die Art der Verbindung, die das Bauteil mit der Baugruppe verbindet, sowie ein entsprechendes Prozessmodell[19] zur Demontage hinterlegt. Die Zuordnung eines Demontageprozesses zu einem Bauteil erfolgt anhand der Verbindungsart, die es mit der Baugruppe herstellt. Das Prozessmodell beschreibt den Demontageprozess des Bauteils hierzu allgemein (implizit). Erst in der nächsten Abstraktionsebene (4) wird der Demontageprozess konkretisiert. Der Nutzer des Assistenzsystems kann für jede Demontageaufgabe festlegen, ob und welche der im Prozessmodell beschriebenen Formen der Unterstützung stattfinden sollen. Weiterhin kann der Nutzer die ermittelte Demontagereihenfolge, z. B. die Reihenfolge von unabhängig voneinander zu demontierenden Bauteilen, nach seinem Wunsch bzw. dem Produktzustand anpassen. Zusammenfassend dient diese Abstraktionsebene der Aufgabendekomposition, die eine komplexe Demontageaufgabe in Teilaufgaben zergliedert, und der Aufgabenallokation, indem der Nutzer die Form der vom RAS ausgehenden technischen Unterstützung definiert.

Abstraktionsebene 4

Die Detailplanung dient der Koordination, Steuerung und Überwachung des Demontageprozesses durch Planung der einzelnen Handlungen der beteiligten technischen Systeme sowie des Menschen. Hierzu muss der im Prozessmodell allgemein beschriebene Demontageprozess anhand der vorliegenden Situation konkretisiert werden. Dies erfolgt

[18] Das Produktmodell wird im Kapitel 6.3.1.1 erläutert.

[19] Das Prozessmodel wird im Kapitel 6.3.2.1 erläutert.

durch ein weiteres in IDAA integriertes Planungssystem. Auf Grundlage des Prozessmodells und des aktuellen Prozesszustands[20] kann ein vollständig definiertes Planungsproblem $\mathcal{P} = (\Sigma, s_0, g)$ automatisiert erzeugt werden. Die Lösung dieses Planungsproblems liefert eine Handlungsfolge der Akteure, die den konkreten Demontageprozess des Bauteils beschreiben. Das in der Planung erzeugte Prozessablaufmodell dient zu Steuerung und Überwachung des Demontageprozesses. Es beschreibt eine Abfolge von parametrierten Handlungsanweisungen sowie die im Demontageprozess zu erwartenden Prozesszustände.

Abstraktionsebene 3

Gegenstand dieser Ebene ist die Steuerung des Demontageprozesses. Die zur Ausführung vorgesehenen Aktionen im Prozessablaufmodell werden schrittweise durch Aktionskommandos über das Ausführungsnetzwerk veranlasst. Durch das Wahrnehmungsnetzwerk und anhand des erwarteten Prozessverlaufs kann durch diese Abstraktionsebene eine Prozessüberwachung erfolgen, die erkennt, ob eine Aktion ausgeführt oder beendet wurde und ob sie den erwarteten Effekt hervorgerufen hat. Dadurch kann der IDAA die Aktionsausführung verschiedener gerätesteuernder Agenten (bzw. technischer Geräte) und des Menschen miteinander koordinieren, Planabweichungen erkennen und im Abweichungsfall durch eine erneute Planung seine Handlungsfolge adaptieren.

Abstraktionsebene 2

Durch eine einheitliche und offene Kommunikationsinfrastruktur (siehe Abbildung 23) nach dem Ansatz des IoT können die gerätesteuernden Agenten stellvertretend für die technischen Systeme, die sie repräsentieren, deren Funktionen und Dienste bereitstellen und somit ein nach außen hin einheitlich wirkendes Ausführungsnetzwerk für Aktionskommandos[21] bilden. Nach demselben Prinzip kann ein Wahrnehmungsnetzwerk geschaffen werden, indem die GSA den Systemzustand[22] ihres zugeordneten Geräts sowie

[20] Der Prozesszustand wird durch das Wahrnehmungsnetzwerk gebildet, siehe Abstraktionsebene 2 und 1.

[21] Aktionskommandos werden über verschiedene Kommunikationskanäle übermittelt, dargestellt durch die grünen Pfeile in Abbildung 23.

[22] Analog zu den Aktionskommandos werden die Gerätestatus über die in Abbildung 23 durch gelbe Pfeile dargestellte Kommunikationskanäle übermittelt.

dessen maschinell wahrgenommenen Zustand der Umgebung in ihrem Netzwerk offen teilen.

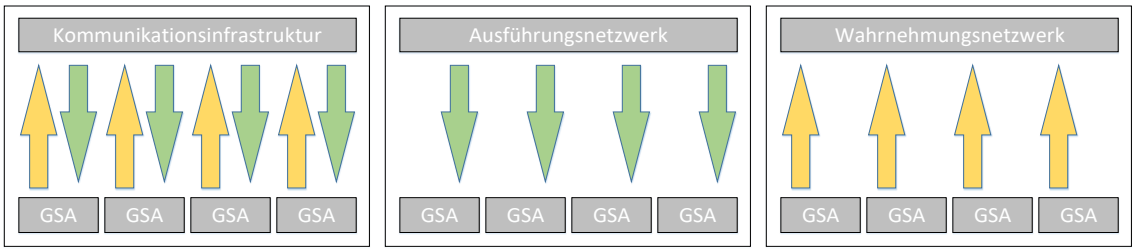

Abbildung 23: Ausführungs- und Wahrnehmungsnetzwerk.

Abstraktionsebene 1

Gegenstand dieser Ebene ist die Bildung einer Geräteabstraktionsschicht. Ein GSA kann unterschiedliche Geräte im RAS einheitlich repräsentieren und dabei die gerätespezifische Komplexität, z. B. bzgl. der Kommunikation oder Ansteuerung, verbergen. Im GSA können hierzu die Gerätefunktionen und -dienste sowie der Gerätezustand oder der Zustand der wahrgenommenen Umgebung einheitlich (im RAS) repräsentiert und kommuniziert werden. Erhält ein GSA ein Aktionskommando aus seinem Netzwerk, setzt er dieses durch entsprechende Steuerungsbefehle[23] an das untergeordnete technische System um. Gleichzeitig überwacht[24] der GSA den Gerätezustand bzw. den Zustand seiner wahrgenommenen Umgebung und meldet Änderungen im Wahrnehmungsnetzwerk.

Abstraktionsebene 0

In dieser Ebene befinden sich die einzelnen technischen Systeme, die die Steuerungsbefehle vom GSA durch ihre Aktoren umsetzen und dadurch auf die Umgebung wirken. Weiterhin erfassen und interpretieren die Sensoren dieser technischen Systeme den eigenen und/oder den Zustand der Umgebung.

[23] Steuerungsbefehle werden über die gerätespezifische Kommunikationsschnittstelle übermittelt, dargestellt durch den roten Pfeil in Abbildung 24.

[24] Der Gerätezustand wird durch die gerätespezifische Kommunikationschnittstelle überwacht, dargestellt als blauer Pfeil in Abbildung 24.

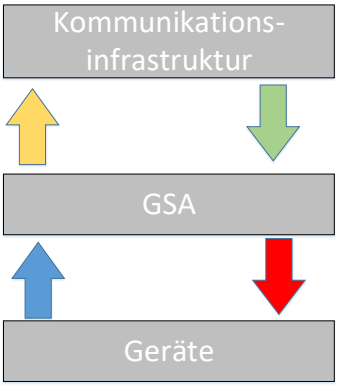

Abbildung 24: Geräteabstraktion durch gerätesteuernde Agenten.

Nachfolgend soll erläutert werden, wie die beiden Agententypen (GSA und IDAA) in Software umgesetzt wurden. Hierzu werden die einzelnen Softwaremodule und deren Kopplungen untereinander in Form einer Softwarearchitektur dargestellt. Des Weiteren wird auf die entwickelten Datenstrukturen und Algorithmen eingegangen.

6.3 Softwarearchitektur des IDAA

Der IDAA dient als zentrale Steuerungseinheit im RAS und ermöglicht die zielgerichtete Zusammenarbeit von Mensch und Assistenzsystem in sich verändernden komplexen Aufgabenstellungen. Um dies zu ermöglichen, sind in IDAAs Softwarearchitektur (siehe Abbildung 26) die in Abbildung 25 definierten Abstraktionsebenen durch mehrere Softwaremodule realisiert.

Der IDAA (in Abbildung 26 grün dargestellt) besitzt mehrere Schnittstellen. Eine grafische Benutzeroberfläche dient als Mensch-Computer-Schnittstelle für den Nutzer, um die Zielstellung der Demontage vorzugeben, die automatisiert ermittelte Demontagereihenfolge der Bauteile zu ändern sowie die Form der Unterstützung in den einzelnen Demontageaufgaben festzulegen. Eine weitere Schnittstelle, zu zwei im Rahmen dieser Arbeit entwickelten Softwarewerkzeugen, dient dem Im- und Export von Elementen der Wissensbasis, d. h. von Produkt- und Prozessmodellen. Für die Erstellung der Prozessmodelle kommt ein Prozessmodell-Editor zum Einsatz und für die Erstellung der Produktmodelle ein CAD-System, in das ein Software-Plug-in implementiert wurde. Eine dritte Schnitt-

stelle stellt die Verbindung zwischen IDAA und dem Ausführungs- sowie Wahrnehmungsnetzwerk her. Die Bedienung der Benutzeroberfläche des IDAA wird in Anhang C beschrieben.

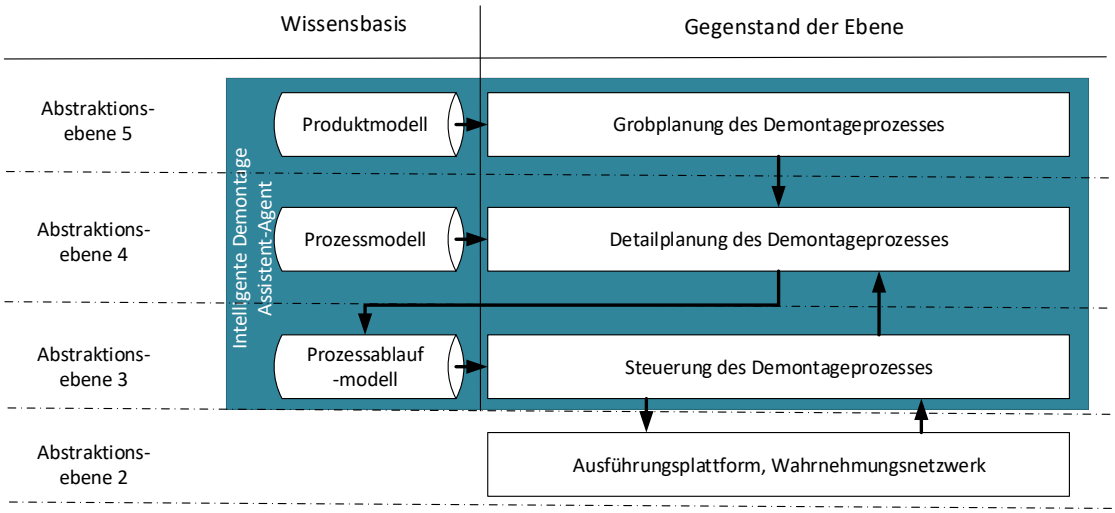

Abbildung 25: Abstraktionsebenen des IDAA.

Die einzelnen Softwaremodule IDAAs werden nun mit Bezug zu den Abstraktionsebenen im Detail beschrieben.

6.3.1 Grobplanung des Demontageprozesses

Der Gegenstand dieser Ebene ist die automatisierte Ermittlung des groben Demontageprozesses in Form einer Abfolge von Demontageschritten. Da der Demontageprozess abhängig vom individuellen Produktaufbau und von der Zielstellung des Nutzers ist, muss für jedes zu demontierende Produkt der Demontageprozess neu ermittelt werden. Damit diese zeitaufwendige Tätigkeit nicht durch den Nutzer erfolgt, besitzt das RAS bzw. der IDAA die Funktionalität der automatisierten Demontageplanung[25] (Abbildung 26). Für die Demontageplanung benötigt IDAA allerdings Informationen über den Produktaufbau und eine Beschreibung der Zielsetzung. Die Art, wie der Aufbau eines Produktes beschrieben wird, ist Gegenstand des nächsten Kapitels.

[25] Das Ansichtsfenster der Demontageplanung des IDAA ist in Abbildung 74 in Anhang C dargestellt.

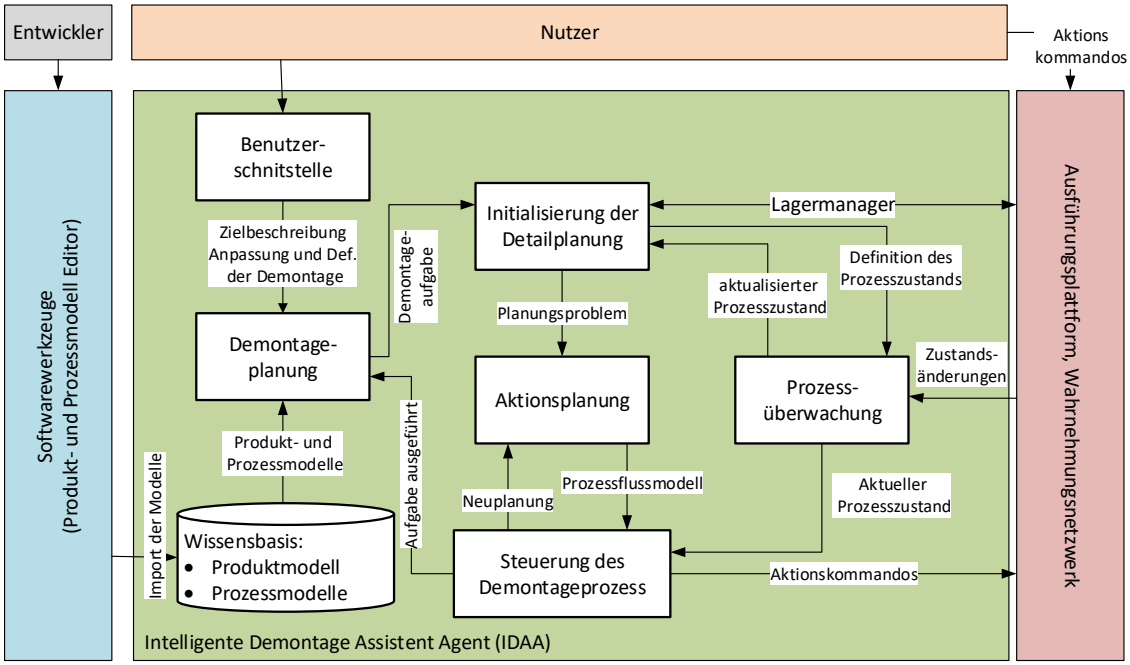

Abbildung 26: Softwarearchitektur, Module und Schnittstellen des IDAA.

6.3.1.1 Wissensbasis der Grobplanung

Für die Beschreibung des Produktaufbaus in einer maschineninterpretierbaren Form wird ein objektorientiertes (Produkt-)Modell P_D verwendet. Dieses wird mithilfe eines CAD-Systems erstellt, in einer Datei im XML-Format[26] exportiert und anschließend in die Wissensbasis des IDAA importiert (vgl. Abbildung 26). Im Produktmodell werden die Bauteile B und die Verbindungen V beschrieben, die zwischen den Bauteilen eines Produktes p bestehen. Durch die Einordnung von Bauteilen in Klassen K_B können den zu einer Klasse zugehörigen Bauteilen die für den Demontageprozess relevanten Informationen zugeordnet werden. Eine eigene Klasse von Bauteilen können z. B. Schrauben, Nieten oder Sicherungsringe bilden. Einfache Bauteile, wie Platten oder Deckel, können dagegen in einer Basis-Bauteilklasse zusammengefasst werden. Durch die Bildung von verschiedenen Klassen von Verbindungen K_V kann eine Zuordnung zu einem verbindungsspezifischen Demontageprozess stattfinden. Zwischen Bauteilen können vielfältige Verbindungen bestehen, wie einfache Kontaktformen, auf- oder eingelegte sowie gefügte Bauteile, oder spezifische Verbindungstechniken, wie Schrauben-, Niet-, Schweiß- oder

[26] Die Abkürzung XML steht für ‚*Extensible Markup Language*' [172]. Im Anhang A wird der Aufbau des Produktmodells im XML-Format dargestellt.

Klebeverbindungen. Beide Klassifizierungsschemata bilden zusammen die Produkt-Taxonomie $P_T = (K_B, K_V)$. Durch die Bildung von Instanzen i_B bzw. i_V dieser Bauteil- sowie Verbindungsklassen können die physikalischen Bauteile B in I_B und Verbindungstechniken V in I_V im Produktmodell $P_D = (I_B, I_V)$ sowie deren Relationen zueinander beschrieben werden.

Zum Beispiel könnte das Produktmodell $P_D = (\{i_{B1}\ i_{B2}\ i_{B3}\},\{\ i_{V1}\ i_{V2}\})$ eine Baugruppe (siehe Abbildung 27) beschreiben, die aus zwei gestapelten Platten i_{B1} und i_{B2} und einer sie verbindenden Schraube i_{B3} besteht. Die Verbindung zwischen den beiden Platten kann durch eine Aufgelegt-Verbindung i_{V1} und die Verbindung beider Platten mit der Schraube durch eine Schraubenverbindung i_{V2} beschrieben werden.

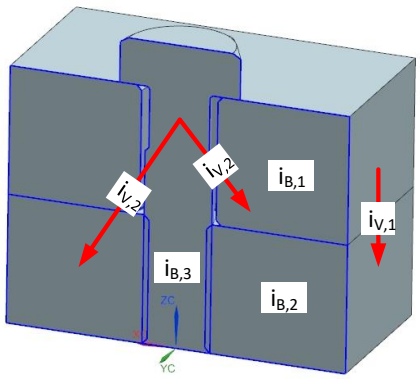

Abbildung 27: Beispiel Baugruppe.

Klassifizierung von Bauteilen

Allgemein dient eine Klassifizierung der systematischen Gruppierung von Objekten anhand ähnlicher geometrischer, funktionaler und physikalischer Merkmale. Im Produktmodell erfolgt die Klassifizierung der Bauteile B anhand von demontagerelevanten Merkmalen. Jede Bauteilklasse $k_B = n_{BK}(M)$ in der Produkt-Taxonomie besitzt einen eindeutigen Namen n_{BK} zur Identifikation und beschreibt eine einzigartige Zusammenstellung von demontagespezifischen Merkmalen $M = \{m_1, m_2, ..., m_n,\}$. Ein Merkmal m wird definiert als, $n_M(i_B) = m_W$ mit dem eindeutig identifizierbaren Namen n_M und Wert m_W des Merkmals einer Bauteilinstanz i_B. Eine Instanz $i_B = n_B(ID, k_B)$ mit dem Namen n_B und der zugeordneten Bauteilklasse k_B, die eindeutig durch ihre ID identifizierbar ist, weist

den in ihrer Klasse beschriebenen Merkmalen Werten zu, $M\left(i_B\right)=\left\{m_1=m_{W,1},...,m_n=m_{W,n},\right\}$ die dem repräsentierten Bauteil entsprechen.

Als Beispiel sind die Merkmale M der Basis Bauteilklasse $k_B^B = Basis(M)$ in der linken Spalte der Tabelle 4 mit ihren Merkmalsnamen dargestellt. Die Merkmalswerte in Spalte 2 von Tabelle 4 entsprechen der Bauteilinstanz $i_{B,2}$ aus Abbildung 27 mit dem Bauteilnamen ‚Platte40x40x15' und der ID ‚2' definiert als

$$i_{B,2} = \text{Platte40x40x15(2,Basis)}.$$

Durch die Merkmalswerte einer Bauteilklasseninstanz können somit die vom Bauteil bestimmten variablen Demontageprozessparameter dargestellt und über die folgende Funktion abgerufen werden:

$$Merkmalswert(n_m, i_B) = m_W \tag{3}$$

Speziellere Klassen von Bauteilen, beispielsweise Schrauben, erben diese Merkmale von der Basis-Bauteilklasse und definieren darüber hinaus weitere Merkmale. Als Beispiel sind die zusätzlichen Merkmale der Schrauben Bauteilklasse in

Tabelle 5 dargestellt. Die Merkmalswerte entsprechen der Instanz $i_{B,3} = \text{M6x40(3,Schrauben)}$ der Schrauben Bauteilklasse aus Abbildung 27. Da die Zerlegung eines Produktes oft durch die Demontage von Modulen oder Unterbaugruppen erfolgt, ist die Beschreibung dieser notwendig. Eine Unterbaugruppe kann als eine Instanz einer Bauteilklasse angesehen werden, die demontiert in ein eigenständiges Produktmodell übergeht $i_B \rightarrow P_D$. In einem Produkt enthaltene Flüssigkeiten und Gase könnten ebenfalls als Bauteilklassen und -instanzen abgebildet werden.

Tabelle 4: Merkmale der Basis-Bauteilklasse.

Merkmals-name	**Merkmalswert**	**Format [Datentyp] (SI-Einheit)**	**Merkmalsbeschreibung**
Bild	bild_wuerfel.jpg	[jpg]	Abbildung des Bauteils im montierten Zustand
Greif-Roboter-werkzeug	Greifer_XY	[string]	Name oder ID des Robotergreifers
Greiflage	1 0 0 0 0 1 0 0 0 0 1 7,5	3x3R,1x3T [1x12double] (mm)	Translationsvektor (T) und Rotationmatrix (R) der Greiflage bzgl. des Baugruppenkoordinatensystems (BGKS)
Material	C45	[string]	Materialbezeichnung
Gewicht	0,2	[double](Kg)	Gewicht des Bauteils
Höhe	15	[double](mm)	Höhe des Bauteils entlang seiner z-Achse
Breite	40	[double](mm)	Breite des Bauteils entlang seiner y-Achse
Länge	40	[double](mm)	Länge des Bauteils entlang seiner x-Achse

Tabelle 5: Zusätzliche Merkmale der Schrauben-Bauteilklasse.

Merkmals-name	**Merkmalswert**	**Format [Datentyp] (SI-Einheit)**	**Merkmalsbeschreibung**
Entschraub-Roboterwerkzeug	Schrauber_XY	[string]	Name oder ID des Roboterwerkzeugs zum Lösen der Verbindungstechnik
Entschraublage	1 0 0 0 0 1 0 0 0 0 1 32,5	3x3R,1x3T [1x12double] (mm)	Translationsvektor (T) und Rotationmatrix (R) der Werkzeuglage bzgl. des BGKS
Manuelles Werkzeug	Schrauben-schluessel_XY	[string]	Name oder ID des manuellen Werkzeugs
Schraubenantrieb	Aussensechs-kant_SW13	[string]	Name des Schraubenantriebs
Rechtsgewinde	Wahr	[bool]	Unterscheidung Rechts- und Linksgewinde
Gewindesteigung	1,25	[double] (mm/Umdrehung)	Gewindesteigung
Gewindelänge im Eingriff	15	[double] (mm)	Eindrehtiefe der Schraube in einer Gewindebohrung

Klassifizierung von Verbindungen

Die Beschreibung von unterschiedlichen Verbindungstechniken erfordert nur die Definition eindeutiger Namen für jede Verbindungsklasse k_V, damit sie in der Menge $K_V = \{k_{V,1}, k_{V,2}, ..., k_{V,n}\}$ identifiziert werden können. Die Art der Verbindungstechnik, die zwischen Bauteilen besteht, bestimmt deren Demontageprozess in erheblichem Maße, aber nicht vollständig. Zum einen ist der Demontageprozess von der Situation abhängig, z. B. vom Produktzustand, und zum anderen von den Fähigkeiten und Absichten der beteiligten handelnden Akteure. Aus diesem Grund kann eine Verbindungsklasse k_v nur einem implizit definierten Demontageprozess, der durch ein Prozessmodell[27] p_Z beschrieben wird, zugeordnet werden:

$$p : k_V \mapsto p_Z \tag{4}$$

Dazu muss der Name der Verbindungsklasse mit dem Namen des Prozessmodells übereinstimmen, d. h. Name(k_V)= Name(p_Z). Jede Verbindungsinstanz i_V kann durch eine eindeutige ID identifiziert werden und verweist mindestens auf eine Bauteilklasseninstanz i_{BV}, die die Verbindungsinstanz erzeugt, und auf mindestens eine Bauteilklasseninstanz i_{BE}, die durch die Verbindung beschränkt wird. Eine Verbindungsinstanz wird wie folgt definiert:

$i_V = (ID, k_V, B_V, B_E)$ mit $B_V = \{ID(i_{BV})\}$ und $B_E = \{ID(i_{BE})\}$.

Auch eine Gruppe von Bauteilinstanzen B_V, die eine Verbindung mit einer oder mehreren Bauteilinstanzen B_E herstellen, kann mit einer Verbindungsinstanz dargestellt werden, solange jedes Element in B_V mit jedem Element von B_E in Verbindung steht. Mit Bezug zur Abbildung 27 wären die Verschraubt-Verbindung und die Aufgelegt-Verbindung wie folgt definiert:

i_{v2} = (2, Verschraubt,{ID(i_{B3})}, {ID(i_{B1}),ID(i_{B2})})

i_{v1} = (1, Aufgelegt,{ID(i_{B1})},{ ID(i_{B2})})

Eine besondere in der Produkt-Taxonomie zu definierende Verbindungsklasse beschreibt den Sachverhalt, dass ein Bauteil den Zugang zu einem anderen Bauteil verhindert, z. B.,

[27] Das Prozessmodell wird in Kapitel 6.3.2.1 erläutert.

weil es letzteres verdeckt. Die entsprechende Verdeckt-Verbindungsklasse beschreibt diesen Fall, verweist aber auf kein Prozessmodell. Die Verdeckt-Verbindungsklasse modelliert lediglich eine für die automatisierte Demontageplanung notwendige Relation zwischen Bauteilen.

Produktgraph

Die Instanzen der Bauteil- und Verbindungsklassen und deren Relationen zueinander können in einen gerichteten zyklenfreien Produktgraphen $P_G = (N, E)$ überführt werden. Die Knoten N des Produktgraphen repräsentieren die Bauteilinstanzen I_B des Produktmodells P_D. Jeder Knoten $n(l_N)$ des Graphen repräsentiert eine Bauteilinstanz i_B und erhält die ID der Instanz als Knotenlabel l_N durch die folgende Abbildung:

$$f : i_B \rightarrow n_{i_B} \text{ mit } n_{i_B} = \left(ID(i_B) \right) \tag{5}$$

Die Kanten E des Produktgraphen werden anhand der Verbindungsinstanzen I_V des Produktmodells erzeugt. Eine gerichtete Kante $e = (n_H, n_T, l_E)$ zeigt vom Elternknoten n_H zum Kindknoten n_T und besitzt als Label l_E die ID der zugehörigen Verbindungsinstanz i_V. Eine Verbindungsinstanz i_V wird entsprechend der Mengen B_V und B_E, durch $|B_V||B_E|$ Kanten repräsentiert. Die sich aus einer Verbindungsinstanz ergebenden Kanten E_{iV} werden über das kartesische Kreuzprodukt der zugehörigen Knotenmengen $N_V = B_V$ und $N_E = B_E$ gebildet, es gilt die Abbildung

$$g : i_V \rightarrow E_{i_V} \tag{6}$$

$$\text{mit } E_{i_V} = N_V \times N_E \times \left\{ ID(i_V) \right\} = \left\{ e_1\left(n_1^V, n_1^E, l_E\right), ..., e_k\left(n_j^V, n_i^E, l_E\right) \right\},$$

$$\text{wobei } n^V \in N_V, n^E \in N_E, i = \left|N_E\right|, j = \left|N_V\right| \text{ und } k = i \cdot j.$$

Alle Kanten E des Produktgraphen P_G ergeben sich über die Vereinigungsmenge der einzelnen Kantenmengen, gebildet aus den Verbindungstechnikinstanzen I_V des Produktmodells P_D nach

$$E = \bigcup_{\forall x \in I_V} g(x)$$

Erläuterung am Beispiel eines Elektromotors

Zum besseren Verständnis sollen drei Verbindungsarten und zwei Bauteilarten am Beispiel eines Elektromotors (siehe Abbildung 28) erläutert werden.

Wie in der Abbildung dargestellt, verbinden die vier Schrauben (mit der ID von 0 bis 3) den Verschlussdeckel (ID 4) sowie die Dichtung (ID 5) mit dem Getriebegehäuse (ID 6). Die Schrauben werden als Instanzen i_{B0}–i_{B3} der Schrauben-Bauteilklasse k_B^S und der Verschlussdeckel, die Dichtung und das Getriebegehäuse als Instanzen i_{B4}, i_{B5} und i_{B6} der Basis-Bauteilklasse k_B^B modelliert und durch die Knoten n_0–n_6 im Produktgraph dargestellt. Die Verbindung zwischen ihnen wird durch eine Instanz i_{V0} der Verschraubt-Verbindungsklasse k_V^S mit der ID 0 wie folgt modelliert:

$i_{V0} = (ID, k_V^S, B_V, B_E)$ mit

$B_V = \{ID(i_{B0}), ID(i_{B1}), ID(i_{B2}), ID(i_{B3})\}$ und $B_E = \{ID(i_{B4}), ID(i_{B5}), ID(i_{B6})\}$

bzw. $i_{V0} = (0, Verschraubt, \{0,1,2,3\}, \{4,5,6\})$.

Eine Instanz i_{V0} der Verschraubt-Verbindungsklasse wird im Produktgraph durch $|B_V||B_E| = 12$ Kanten, $E_{iv0} = g(i_{V0}) = \{e_1(n_0,n_4,0), e_2(n_0,n_5,0), e_3(n_0,n_6,0), \ldots, e_{12}(n_3,n_6,0)\}$ dargestellt. Von jeder Schraube zeigt eine Kante zum Verschlussdeckel, zur Dichtung sowie zum Getriebegehäuse, wie in Abbildung 28 demonstriert. Als Nächstes wird die Verbindung zwischen Verschlussdeckel und Dichtung betrachtet. Sie wird als Instanz i_{V1} der Aufgelegt-Verbindungsklasse k_V^A mit der ID 1 wie folgt erzeugt $i_{V1} = (1, k_V^A, \{ID(i_{B4})\}, \{ID(i_{B5})\})$ bzw. $i_{V1} = (1, Aufgelegt, \{4\}, \{5\})$ und mit nur einer Kante vom Verschlussdeckel zur Dichtung dargestellt $E_{iv1} = g(i_{V1}) = \{e_1(n_4,n_5,1)\}$. Eine Instanz i_{V2} der Verdeckt-Verbindungsklasse k_V^V mit ID 2 wird verwendet, um zu beschreiben, dass ein Bauteil den Zugang zu einem anderen Bauteil verhindert. Dies ist der Fall beim Verschlussdeckel, der die Zylinderschraube mit ID 8 verdeckt. Die Schraube wird als Instanz i_{B8} der Schrauben-Bauteilklasse k_B^S und die Verdeckt-Verbindungsinstanz $i_{V2} = (2, k_V^V, \{ID(i_{B4})\}, \{ID(i_{B8})\})$ bzw. $i_{V2} = (2, Verdeckt, \{4\}, \{8\})$ wird durch eine Kante $E_{iv2} = g(i_{V2}) = \{e_1(n_4,n_8,2)\}$ vom Knoten n_4 des Verschlussdeckelinstanz zum Knoten n_7 der Zylinderschraubeninstanz dargestellt.

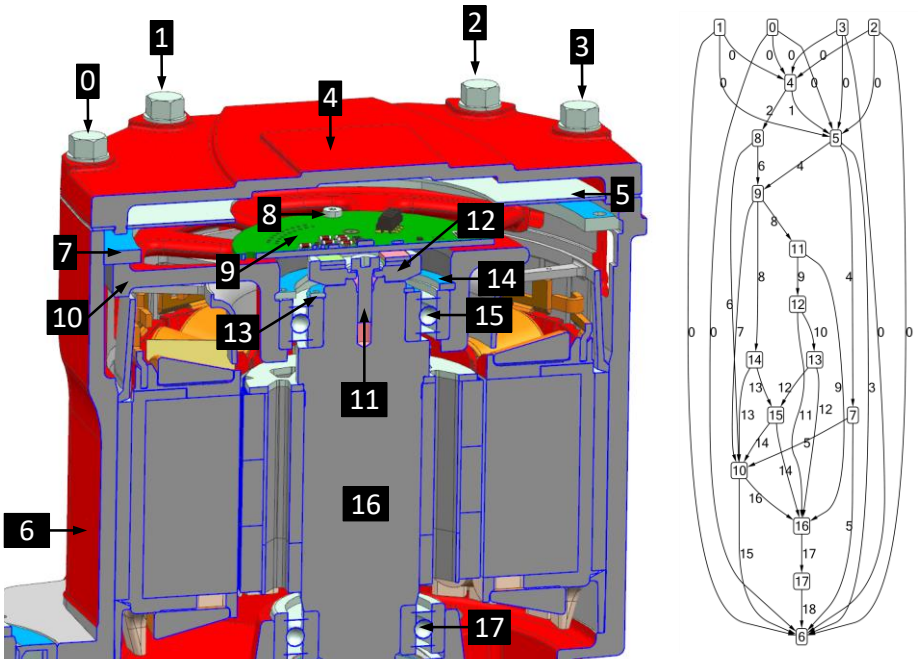

Abbildung 28: Schnittdarstellung und Produktgraph eines Elektromotors.

Erzeugung des Produktmodells

Der Ablauf zur Erstellung des Produktmodells mithilfe des CAD-Plug-ins ist in Abbildung 29 dargestellt. Das CAD-Plug-in wurde innerhalb einer betreuten Projektarbeit entwickelt und in die PLM-Software Siemens NX über deren offene Schnittstelle NX Open integriert. Das Plug-in unterstützt den Nutzer bei der manuellen Erzeugung des Produktmodells, d. h. der Erstellung von Bauteil- sowie Verbindungsinstanzen. Der Nutzer wählt hierzu die Art der Verbindung aus einer Liste aus und selektiert die Bauteile, die diese Verbindung herstellen. Nachfolgend wählt er die Bauteile, die durch diese Verbindung beschränkt werden. Durch Informationen, die manuell in die CAD-Modelle der Bauteile hinterlegt wurden, erkennt das Plug-in die zugehörige Bauteilklasse und bezieht automatisch die entsprechenden Merkmalswerte, z. B. die Lage des Bauteils bzgl. des Baugruppenkoordinatensystems, aus den CAD-Modellen. Sind alle Verbindungen definiert, wird das Produktmodell als XML-Datei exportiert. Da ein Produkt i. d. R. in vielen Varianten sowie aus einer hohen Anzahl von Bauteilen und Verbindungen besteht, stellt sich die Frage, wie das Produktmodell automatisiert erstellt werden kann. Hierzu könnten theoretisch entsprechend angepasste Produktkonfiguratoren und Variantenmanagementsysteme verwendet werden. Für den Bezug der Merkmalswerte einzelner Bauteile können die bereits bestehenden Informationsquellen eines Unternehmens dienen, z. B. Bauteil-

Bibliotheken in CAD-, PLM- oder ERP-Systemen, in denen Bauteile mit Sachmerkmalen (z. B. Material, Gewicht usw.) verwaltet werden.

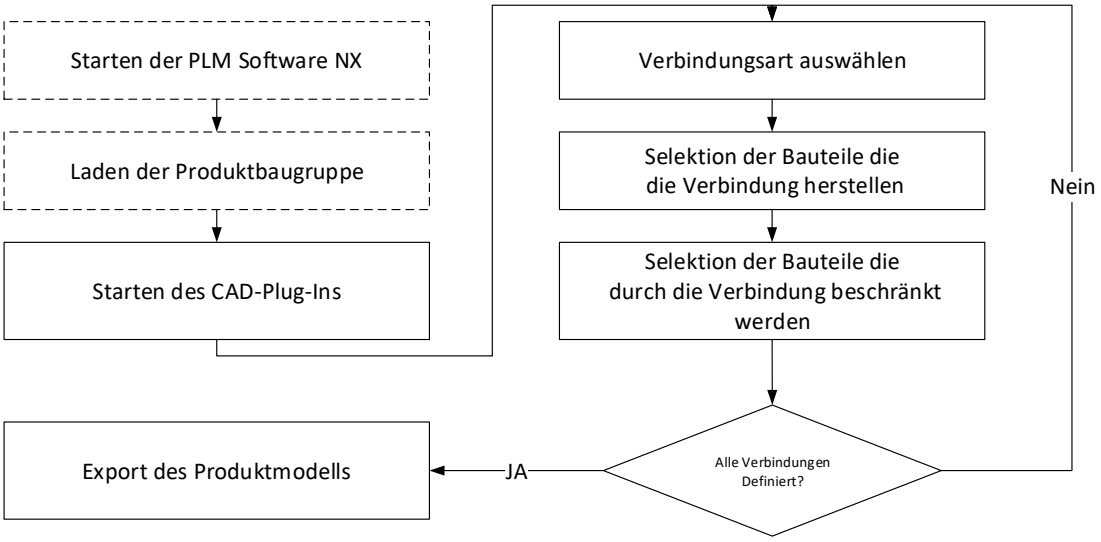

Abbildung 29: Ablauf der Erzeugung des Produktmodells.

Teilweise können diese Sachmerkmale direkt als Demontageinformationen in das Produktmodell übernommen werden. Andere Informationen können nur aus dem konkreten CAD-Modell einer Produktvariante gewonnen werden, zum Beispiel die Massenverteilung eines Bauteils oder dessen Position und Orientierung bzgl. des Baugruppenkoordinatensystems (BGKS). Weitere wichtige Demontageinformationen können zurzeit nur durch den Menschen zuverlässig definiert werden, wie die Auswahl eines zum Bauteil passenden Robotergreifwerkzeugs oder die Definition der Greiflage. Auch spezielle Roboterwerkzeuge zum Lösen von Verbindungstechniken und ihre Positionierung müssen in den Demontageinformationen hinterlegt werden. Hier zeigt sich, dass vorerst keine vollständig automatisierte Produktmodellerzeugung möglich sein wird und dass zunächst ein Simulations- und Entwicklungsaufwand verbleiben wird, dessen Wirtschaftlichkeit in erheblichem Maße durch die Qualität der Softwarewerkzeuge bestimmt wird. Eine nicht zu unterschätzende Problemstellung stellt die Klassifizierung und Namensgebung selbst dar. Verschiedene Menschen benennen und klassifizieren Bauteile oder Verbindungsarten unterschiedlich. Neben Normen als Quelle für anerkannte Klassifizierungen bieten industrielle Zweckverbände wie die ecl@ss e.V. Klassifizierungssysteme an. Das ecl@ss Klassifizierungssystem, das für den Austausch von Produktstammdaten zwischen Unter-

nehmen entworfen wurde, stellt ein umfangreiches Klassifizierungssystem zur Verfügung. Die einzelnen Klassen müssten jedoch um demontagespezifische Informationen erweitert werden. Welche Demontageinformationen dies sind, kann nur durch Applikationsversuchen ermittelt werden. Dabei müssen die Abhängigkeit zur verwendeten Hardware, Software sowie die Prozessumsetzung berücksichtigt werden.

6.3.1.2 Automatisierte Demontageplanung

Das Modul der Demontageplanung (vgl. Abbildung 26) ermittelt die Reihenfolge der zu demontierenden Bauteile sowie deren zugeordnete Demontageprozesse bzw. Prozessmodelle und bildet einen Demontageplan bestehend aus Demontageschritten, die wiederum aus Demontageaufgaben bestehen. Der Ablauf der Demontageplanung ist in Abbildung 30 dargestellt und im Anhang C im Rahmen der Bedienung der Benutzeroberfläche des IDAA erklärt. Sofern noch nicht erfolgt, müssen das Produkt- sowie Prozessmodell durch den Nutzer in den IDAA über die grafische Benutzeroberfläche (vgl. Abbildung 26) in die Wissensbasis geladen werden. Nach dem Import des Produktmodells erzeug der IDAA automatisiert den Produktgraphen.

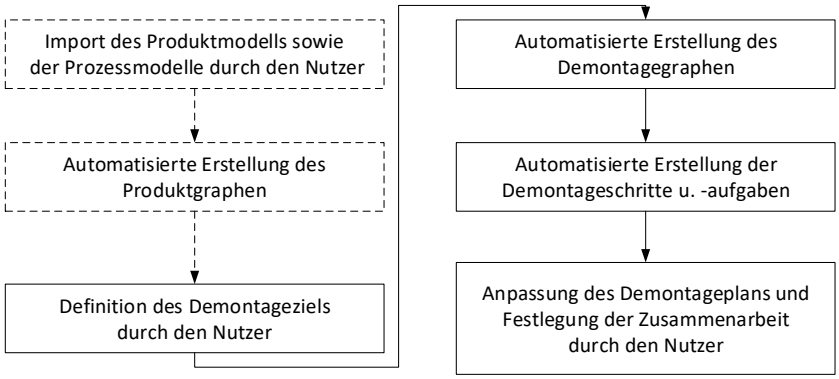

Abbildung 30: Ablauf der Demontageplanung.

Für die Demontagezielsetzung wählt der Nutzer über die grafische Benutzeroberfläche ein Zielbauteil aus der Bauteilliste des Produktmodells aus. Auf Basis dieser Demontagezielstellung erzeugt der IDAA zuerst einen Demontagegraphen und nachfolgend einen Demontageplan. Dieser kann nachfolgend durch den Nutzer über die Benutzeroberfläche des IDAA angepasst werden, z. B., um die Reihenfolge von Demontageaufgaben innerhalb eines Demontageschritts zu variieren. Weiterhin kann der Nutzer für die Demontage jedes Bauteils festlegen, welche Form der Zusammenarbeit stattfinden soll, etwa, ob der

Nutzer oder das RAS diese Aufgabe übernehmen soll oder ob beide bei der Aufgabenstellung zusammenarbeiten. Im Folgenden sollen die einzelnen Schritte und ihre Algorithmen näher beschrieben werden.

Erzeugung des Produktgraphen

Die Vorgehensweise zur Erzeugung des Produktgraphen aus dem Produktmodell ist in Algorithmus 2 dargestellt. In der Schleife in Zeile 3 wird für jede im Produktmodell definierte Bauteilinstanz ein Knoten nach der Formel (5) erzeugt und dem Produktgraphen hinzugefügt (Zeile 4). In der Schleife in Zeile 5 werden die Kanten der Verbindungsinstanzen entsprechend nach der Formel (6) erzeugt und den Kanten des Produktgraphen hinzugefügt (Zeile 12). In Zeile 13 wird der erzeugte Produktgraph zurückgegeben.

Algorithmus: Produktgraph
Eingabe: $P_D = (I_B, I_V)$
Ausgabe: $P_G = (N, E)$

(1) $N \leftarrow \varnothing, E \leftarrow \varnothing$
(2) $I_B \leftarrow \text{Bauteilinstanzen}(P_D), I_V \leftarrow \text{Verbindungsinstanzen}(P_D)$
(3) **foreach** $i_B \in I_B$
(4) $\quad N \leftarrow N \cup \{n(\text{ID}(i_B))\}$
(5) **foreach** $i_V \in I_V$
(6) $\quad B_V \leftarrow \text{VHBauteilinstanzen}(i_V)$
(7) $\quad B_E \leftarrow \text{VBBauteilinstanzen}(i_V)$
(8) $\quad$ **foreach** $i_{BV} \in B_V$
(9) $\quad\quad n_H \leftarrow n \in N : \text{KnotenLabel}(n) = \text{ID}(i_{BV})$
(10) $\quad\quad$ **foreach** $i_{BE} \in B_E$
(11) $\quad\quad\quad n_T \leftarrow n \in N : \text{KnotenLabel}(n) = \text{ID}(i_{BE})$
(12) $\quad\quad\quad E \leftarrow E \cup \{e(n_H, n_T, ID(i_V))\}$
(13) **return** $P_G = (N, E)$

Algorithmus 2: Erstellung des Produktgraphen.

Erzeugung des Demontagegraphen

Anhand des zu demontierenden Bauteils, bzw. dessen Knoten n_Z im Produktgraphen, erfolgt nach Algorithmus 3 die Erzeugung des Demontagegraphen. In Zeile 2 wird mithilfe des Zielknotens n_Z dessen Kindknoten N_T im Produktgraph ermittelt und den Knoten N_D des Demontagegraphen D_G hinzugefügt. In Zeile 3 werden die vom Zielknoten n_Z ausgehenden Kanten ermittelt und den Kanten E_D des Demontagegraphen D_G hinzugefügt. Danach, in Zeile 4, wird der Zielknoten n_Z einer FIFO-Warteschlange (oList) hinzugefügt. In der nachfolgenden Schleife (Zeile 5) wird der FIFO-Warteschlange jeweils ein Knoten n entnommen (Zeile 6), solange sich Elemente in der Warteschlange befinden.

Algorithmus: Demontagegraph
Eingabe: $P_G = (N, E), n_Z$
Ausgabe: $D_G = (N_D, E_D)$

(1) $N \leftarrow \text{Knoten}(P_G), E \leftarrow \text{Knoten}(P_G)$
(2) $N_D \leftarrow \{n_Z\} \cup \{\forall n \in N \mid \exists e \in E \text{ mit } e(n_Z, n, l_E)\}$
(3) $E_D \leftarrow \{\forall e \in E \mid \exists n \in N \text{ mit } e(n_Z, n, l_E)\}$
(4) oList.enqueue(n_Z)
(5) **while** oList $\neq \emptyset$
(6) $n \leftarrow$ oList.dequeue()
(7) $N_H \leftarrow \{\forall m \in N \mid \exists e \in E \text{ mit } e(m, n, l_E)\}$
(8) $N_H^* \leftarrow \{\forall n_H \in N_H \mid \neg \exists m \in N_D : n_H \equiv m\}$
(9) $N_D \leftarrow N_D \cup N_H^*$
(10) $E_D \leftarrow E_D \cup \{\forall e \in E \mid \exists m \in N \text{ mit } e(m, n, l_E)\}$
(11) oList.enqueue(N_H^*)
(12) **return** $D_G = (N_D, E_D)$

Algorithmus 3:Erstellung des Demontagegraphen.

In der Zeile (7) werden nun die Elternknoten N_H des Knoten n ermittelt. In der folgenden Zeile 8 werden die Elternknoten auf die Menge N^*_H beschränkt, die nicht bereits in den Knoten N_D des Demontagegraphen vorkommen. Diese neuen Knoten N^*_H werden anschließend den Knoten des Demontagegraphen (Zeile 9) sowie der FIFO-Warteschlange (Zeile 11) hinzugefügt. In Zeile 10 werden die eingehenden Kanten vom Knoten n ermittelt und den Kanten E_D des Demontagegraphen hinzugefügt. Der Algorithmus endet,

wenn keine Knoten mehr in der Warteschlange bevorratet sind, und liefert den Demontagegraphen zurück.

Der erzeugte Demontagegraph enthält nur die Bauteile, die demontiert werden müssen, sowie die Bauteile, die durch die Verbindungen des Zielbauteils fixiert werden. Durch die Kanten im Demontagegraphen werden die Abhängigkeiten dargestellt, die bzgl. der Demontagereihenfolge durch die Verbindungen zu berücksichtigen sind. Auf Basis dieses Demontagegraphen können nun die einzelnen Demontageschritte sowie Demontageaufgaben definiert und in einem Demontageplan zusammengefasst werden.

Erzeugung des Demontageplans

Die in einem Demontageschritt demontierbaren Bauteile stellen im Demontagegraphen Knoten N_S ohne eingehende Kanten dar. Alle Knoten ohne eingehende Kanten bilden somit Bauteile ab, die in einem Demontageschritt demontiert werden können. Jedes der zu demontierenden Bauteile definiert eine Demontageaufgabe t im jeweiligen Demontageschritt d. Eine Demontageaufgabe ist definiert als Tripel $t = (i_B, i_V, p_Z)$ bestehend aus einer Bauteil-i_B und Verbindungsinstanz i_V sowie dem der Verbindungsklasse zugeordneten Prozessmodell p_Z. Alle Demontageaufgaben werden in der Menge $T = \{t_1, t_2, ..., t_n,\}$ eines Demontageschritts $d = (T)$ zusammengefast. Die Demontageaufgaben in einem Demontageschritt können unabhängig voneinander gelöst werden, es liegt somit keine Ordnung in T vor. Nach der Erstellung eines Demontageschrittes werden die ermittelten Bauteile und ihre Verbindungstechniken, bzw. Knoten und deren Kanten im Demontagegraphen, entfernt. Der Vorgang der Demontageschritterstellung wiederholt sich, bis die Demontageaufgabe des Zielbauteils erstellt wurde. Der Demontageplan besteht somit aus einer Sequenz von Demontageschritten $D_S = \langle d_1, d_2, ..., d_m \rangle$.

In Algorithmus 4 ist die Erzeugung des Demontageplans beschrieben. Er basiert auf dem Prinzip der topologischen Sortierung. Der Algorithmus benötigt als Eingabe das Produktmodell P_D, die Prozessmodelle P_Z, den Demontagegraphen D_G sowie das Zielbauteil bzw. dessen Knoten n_Z. Die Schleife in Zeile 3 wird so lange durchlaufen, bis das Zielbauteil als Demontageaufgabe erstellt wurde. In jedem Schleifendurchgang werden die Knoten N_S im Demontagegraphen ermittelt, die keine eingehenden Kanten besitzen (Zeile 4). Darüber hinaus wird eine neue (leere) Menge T zur Speicherung der Demontageaufgaben im jeweiligen Demontageschritt erstellt (Zeile 5). Für jeden Knoten in der Menge N_S

werden in der Schleife (Zeile 6) die entsprechende Bauteilinstanz (Zeile 7), Verbindungsinstanz (Zeile 8) sowie das zugeordnete Prozessmodell p_Z ermittelt (Zeile 9) und es wird eine neue Demontageaufgabe erstellt. Diese wird anschließend in Zeile 10 der Menge T hinzugefügt. Durch die Prüfung in Zeile 11 wird erkannt, ob die Demontageaufgabe für die Entfernung des Zielbauteils erstellt wurde und somit die Schleifenbedingung von Zeile 3 negiert werden muss (Zeile 12). Wurden alle Demontageaufgaben in der Schleife (Zeile 6) erzeugt, wird ein Demontageschritt entwickelt und dem Demontageplan angefügt (Zeile 13). Vom Algorithmus zurückgegeben wird der Demontageplan in Zeile 14.

Algorithmus: Erzeugung von Demontageschritten und -aufgaben
Eingabe: $D_G = (N_D, E_D), P_D = (I_B, I_V), P_Z = \{p_{Z,1}, p_{Z,2}, ..., p_{Z,k}\}, n_Z$
Ausgabe: $D_S = \langle d_1, d_2, ..., d_n \rangle$

(1) $D_S = \langle\ \rangle, N_D \leftarrow Knoten(D_G), E_D \leftarrow Kanten(D_G)$
(2) SchleifenBedingung = *true*
(3) **while** SchleifenBedingung
(4) $N_S \leftarrow \{\forall n \in N_D \,|\, \neg\exists e \in E_D \text{ mit } e(m, n, l_E), m \in N_D\}$
(5) $T \leftarrow \varnothing$
(6) **foreach** $n_S \in N_S$
(7) $i_B \leftarrow \text{Bauteilinstanz}(n_s, P_D)$
(8) $i_V \leftarrow \text{Verbindungsinstanz}(n_s, P_D)$
(9) $p_Z \leftarrow \text{Prozessmodell}(Klasse(i_V), P_Z)$
(10) $T \leftarrow T \cup \{t(i_B, i_V, p_Z)\}$
(11) **if** $n_S \equiv n_Z$
(12) SchleifenBedingung = *false*
(13) $D_S \leftarrow D_S + \langle d(T) \rangle$
(14) **return** $D_S = \langle d_1, d_2, ..., d_n \rangle$

Algorithmus 4: Erstellung von Demontageschritten und -aufgaben.

Mit Bezug zum Beispiel aus Abbildung 27 könnte der erzeugte Demontageplan für die Demontage des Bauteils $i_{B,1}$ zwei Demontageschritte enthalten $D_S = \langle d_1, d_2 \rangle$, wobei jeder Demontageschritt nur eine Demontageaufgabe enthalten würde. Im ersten Demontageschritt und der darin definierten Demontageaufgabe $d_1 = \{t_1(i_{B,3}, i_{V,2}, p_Z^S)\}$ würde das Lösen der Schraubenverbindung $i_{V,2}$ und das Entfernen der Schraube $i_{B,3}$ durch den in p^S_Z definierten Demontageprozess von Schrauben erfolgen. Im zweiten Demontageschritt

und der darin definierten Demontageaufgabe $d_2 = \left\{ t_2 \left(i_{B,1}, i_{V,1}, p_Z^A \right) \right\}$ würde das Zielbauteil $i_{B,1}$ von seiner Auflage (Bauteil $i_{B,2}$) getrennt und in einer Box abgelegt werden.

Anpassung des Demontageplans und Festlegung der Zusammenarbeit

Nach der automatischen Erstellung kann der Nutzer den Demontageplan über die Benutzeroberfläche des IDAA manipulieren, um zum Beispiel die Reihenfolge von Demontageaufgaben in einem Demontageschritt zu ändern. Außerdem kann er über die Benutzeroberfläche definieren, wie jede Demontageaufgabe ausgeführt wird. Hierzu kann der Nutzer zwischen den vordefinierten Formen der Arbeitsteilung bzw. der Assistenz, die im jeweiligen Prozessmodell hinterlegt ist, eine zur Situation passende auswählen. Damit soll dem Nutzer die Möglichkeit gegeben werden, auf den Produktzustand angemessen zu reagieren oder den Demontageprozess entsprechend seinen Wünschen zu gestalten. Optimierungsfähig ist hier die Art der Mensch-Maschine-Schnittstelle. Das Interagieren mit der Benutzeroberfläche IDAAs behindert den Arbeitsfluss des Menschen, da dieser sich dem Bildschirm zuwenden muss und mittels Computermaus oder Touchscreen die Anpassungen vornehmen muss. Eine Kommunikation mittels natürlicher Sprache könnte den Arbeitsfluss positiv beeinflussen. Um diesen Definitionsaufwand weiter zu verringern, könnte etwa eine Intentionserkennung bzw. -vorhersage dienen. Bei der Intentionserkennung wird anhand des Verhaltens des Nutzers, z. B. seiner Gestik, dessen Intention für die Aufgabenverrichtung automatisch ermittelt. Greift der Nutzer zum Beispiel zum Schraubendreher, kann das System davon ausgehen, dass er die Schraube löst und keine Unterstützung vom Assistenzsystem erwartet. Verlässt der Nutzer hingegen den Demontagearbeitsplatz, kann das System daraus schließen, dass es selbst tätig werden soll. Die Intentionsvorhersage könnte anhand von Prozessmerkmalen (wie dem Bauteilgewicht) und vorausgegangenen Interaktionen mit dem Nutzer lernen, wann er welche Form von Unterstützung erwartet. Hierzu muss das System, auf Basis von aufgezeichneten Interaktionen des Nutzers, Zusammenhänge zwischen Prozessmerkmalen und der vom Nutzer gewählten Form der Arbeitsteilung erkennen. Diese Zusammenhänge können durch maschinelle Lernverfahren, z. B. Entscheidungsbäume, modelliert werden. Die Implementierung dieser Ansätze könnte den Prozess der Festlegung der Zusammenarbeit weiter verbessern und zur dritten Klasse von Assistenzsystemen nach [74] führen (vgl. Seite 27).

Nach der Anpassung des Demontageplans und der Festlegung der Zusammenarbeit kann der Nutzer die Ausführung des Plans starten und gemeinsam mit dem RAS das Produkt

demontieren, denn alle weiteren Schritte erfolgen ab hier automatisiert. Die im Demontageplan beschriebenen Demontageaufgaben werden hierzu nacheinander abgearbeitet. Wie dabei jede Demontageaufgabe konkretisiert und ausgeführt wird, wird nachfolgend, beginnend mit der Detailplanung des Demontageprozesses, erklärt.

6.3.2 Detailplanung des Demontageprozesses

In der Detailplanung findet die Konkretisierung des Demontageprozesses einer Demontageaufgabe $t = (i_B, i_V, p_Z)$ durch Planung der Handlungen der beteiligten Akteure statt. Hierzu erfolgen zwei Schritte, die Initialisierung der Detailplanung und die Aktionsplanung (vgl. Abbildung 26). Diese beiden Schritte sind im Anhang C im Rahmen der Bedienung der Benutzeroberfläche des IDAA weiterhin beschrieben. In der Initialisierung erfolgt die automatisierte Definition des Planungsproblems $\mathcal{P} = (\Sigma, s_0, g)$ auf Basis des in der Demontageaufgabe hinterlegten Prozessmodells sowie des Softwaremoduls zur Prozessüberwachung. Im Softwaremodul der Aktionsplanung erfolgt anschließend die Planung der Handlungen durch Lösung des Planungsproblems und die Erstellung des Prozessablaufmodells, das die Grundlage für die Steuerung des Demontageprozesses bildet.

Wesentlich für die automatisierte Definition des Planungsproblems sind die Prozessmodelle. Deshalb wird in diesem Kapitel zuerst der Aufbau der Prozessmodelle beschrieben. Nachfolgend wird das Softwaremodul für die Initialisierung der Detailplanung erörtert und dessen Funktion zur Bildung des Planungsproblems aufgezeigt. Am Ende dieses Kapitels wird das Softwaremodul zur Aktionsplanung (vgl. Abbildung 26) beschrieben.

6.3.2.1 Wissensbasis der Detailplanung

Die Wissensbasis der Detailplanung besteht aus einer Menge von Prozessmodellen[28] $P_Z = \{p_{Z,1}, ..., p_{Z,k}\}$. Jedes einer Verbindungsklasse zugeordnete Prozessmodell p_Z stellt ein Planungsproblem $\mathcal{P} = (\Sigma, s_0, g)$ dar, dessen Ziel die Entfernung des jeweiligen Bauteils aus der Baugruppe darstellt. Gegenüber der in Kapitel 5.2.2 beschriebenen Repräsenta-

[28] Die Prozessmodelle im XML-Importformat sind im Anhang B dargestellt.

tion eines Planungsproblems war für die vorliegende Problemstellung die Umsetzung einer anderen Repräsentationsform von Vorteil. Während in klassischen Planungssystemen die Werkstücke, der Greifer und der Roboter durch einfache Symbole repräsentiert wurden, erhalten sie in den Prozessmodellen eine interne Struktur, die ihre individuellen Eigenschaften, Zustände und ggf. Funktionen repräsentiert. Dies begünstigt eine allgemeingültigere Repräsentation eines technischen Systems, losgelöst von einer Planungsdomäne, und fördert die Bildung von unabhängigen Systemmodellen, die in verschiedenen Prozessmodellen wiederverwendet und entsprechend der jeweiligen Aufgabenstellung angepasst werden können. Besonders für komplexe Systeme wie ein Industrieroboter erspart dies wiederholten Modellierungsaufwand. Weiterhin wurde die Beschreibung von Zuständen und Aktionen erweitert, um den Anforderungen hinsichtlich einer realen Prozesssteuerung zu genügen. Der genaue Aufbau eines Prozessmodells und die Modifikationen hinsichtlich der Zustands- und Aktionsbeschreibung sollen nachfolgend dargestellt werden.

Prozessmodell

Wie im Bereich der Multiagentensysteme werden im Prozessmodell die Elemente einer Umgebung in passive Objekte und handelnde Agenten eingeteilt und modelliert. Die Agenten beschreiben die einzelnen technischen Systeme sowie den Nutzer und können im Gegensatz zu Objekten Aktionen ausführen. Die Beschreibung von Objekten und Agenten findet im Prozessmodell durch generalisierte Objekt- (o) bzw. Agentenmodelle (α) statt. Alle an einem verbindungsspezifischen Demontageprozess beteiligten Objekte und Agenten werden durch diese Objekt- bzw. Agentenmodelle beschrieben und in den Mengen $\mathcal{O}$ bzw. $\mathcal{A}$ im Prozessmodell zusammengefasst, wobei die Menge der Agenten zurzeit eine feste Teamzusammensetzung für den jeweiligen Demontageprozess kennzeichnet. Neben der Beschreibung von Objekten und Agenten können in einem Prozessmodell unterschiedliche Möglichkeiten modelliert werden, wie das RAS den Nutzer in der Demontageaufgabe unterstützt. Die dadurch entstehenden unterschiedlichen Formen der Arbeitsteilung oder Unterstützung werden in der Menge $\mathcal{B}$ zusammengefasst.

Zusammenfassend wird ein Prozessmodell p_Z durch einen Ausdruck der folgenden Form beschrieben:

$$p_Z = \mathrm{n}_P(\mathrm{b}, \mathcal{O}, \mathcal{A}, \mathcal{B})$$

In diesem Tupel steht n_P für den Namen des Prozessmodells und dient für dessen eindeutige Identifizierung und Zuordnung zu einer Verbindungsklasse. Weiterhin steht b für die vom Nutzer selektierte Form der Unterstützung aus der Menge $\mathcal{B}$, $\mathcal{O}$ für die Menge der Objekte und $\mathcal{A}$ für die Menge der Agenten. Zur besseren Verständlichkeit wird ein konkretes Prozessmodell beschrieben. Das Prozessmodell p_z^A wird der Aufgelegt-Verbindungsklasse k_V^A zugeordnet $p : k_V^A \mapsto p_Z^A$ und dient der Demontage von einfach aufeinandergestapelten Bauteilen. Es wird wie folgt definiert:

$$p_Z^A = Aufgelegt(\mathrm{b}, \mathcal{O}, \mathcal{A}, \mathcal{B}).$$

Dieses Prozessmodell soll für die Demontage einer Beispielbaugruppe dienen, die aus drei aufeinandergestapelten Bauteilen $i_{B,1}$ $i_{B,2}$ $i_{B,3}$ besteht. Anhand dieses Beispiels soll nun erläutert werden, welche und wie die einzelnen Objekte und Agenten sowie die möglichen Formen der Arbeitsteilung bzw. Unterstützung für dieses Prozessmodell beschrieben werden.

Beschreibung von Objekten und Agenten

Ein Objektmodell o wird im Prozessmodell durch einen eindeutigen Namen n, eine Menge von Parametern Y und eine Zustandsmenge Z beschrieben. Für ein Objekt steht somit das Dupel:

$$o = n(Z, Y)$$

Für das oben genannte Beispiel eines Prozessmodells werden zwei Objekte beschrieben: Das erste Objekt o_1 kennzeichnet das jeweilige zu demontierende Bauteil und das zweite Objekt o_2 eine passende Transportbox für dessen Ablage. Die beiden Objekte werden wie folgt definiert:

$$o_1 = Bauteil(Z, Y),\ o_2 = Transportbox(Z, Y)$$

und beschreiben die Menge $\mathcal{O} = \{o_1, o_2\}$ in $p_Z^A = Aufgelegt(\mathrm{b}, \mathcal{O}, \mathcal{A}, \mathcal{B})$

Ein Objektmodell steht stellvertretend für verschiedene reale Objekte. Zum Beispiel könnte das Objekt o_1 bei der Demontage der Beispielbaugruppe für eines der drei Bauteile $i_{B,1}$ $i_{B,2}$ $i_{B,3}$ stehen. Auch das Objekt o_2 steht jeweils für eine andere Transportkiste.

Die Definition eines Agenten im Prozessmodell erfolgt analog, mit dem Unterschied, dass zusätzlich dessen Aktionen in der Menge A zusammengefasst werden. Für ein Agentenmodell steht somit das Triple

$\alpha = \mathrm{n}(Z, Y, A)$.

Für das oben genannte Beispiel eines Prozessmodells werden drei Agenten beschrieben: Der erste Agent α_1 beschreibt den Menschen, der zweite Agent α_2 den Roboter und der dritte Agent α_3 den steuerbaren Greifer. Die Agenten werden wie folgt definiert:

$\alpha_1 = \mathit{Mensch}(Z, Y, A), \alpha_2 = \mathit{Roboter}(Z, Y, A), \alpha_3 = \mathit{Greifer}(Z, Y, A)$ und beschreiben die Menge $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3\}$ in $p_Z^A = \mathit{Aufgelegt}(\mathrm{b}, \mathcal{O}, \mathcal{A}, \mathcal{B})$.

Im Gegensatz zu Objekten, die verschiedene reale Objekte abbilden können, sind Agenten einem realen Akteur im Demontageprozess zugeordnet und beschreiben zurzeit eine feste Teamzusammensetzung.

Beschreibung der Zustandsmenge eines Objekts oder Agenten

Die Zustandsmenge eines Objekts bzw. Agenten n wird in folgender Form definiert: $Z_n = \{z_1, z_2, ..., z_k\}$. Diese Erweiterung berücksichtigt drei wesentliche Punkte:

– die Bildung eines angenommenen initialen Zustands $Z_n \rightarrow X_n(s_0)$ und
– den vorgesehenen Zielzustand $Z_n \rightarrow X_n(g)$ des Objekts oder Agenten n im Demontageprozess.
– Um den Zustand eines Objekts oder Agenten über das Wahrnehmungsnetzwerk zu ermitteln und zu aktualisieren, wird jeder Zustandsvariablen x zudem ein Kommunikationskanal k zugeordnet.

Jedes Element z in Z der Form

$$z = \left(x, w_0, w_g, k, D_{ns} \right)$$

beschreibt eine variable Eigenschaft des zugehörigen Objekts oder Agenten durch eine Zustandsvariable der Form $x = n_s\left(n\right)$ mit einem eindeutigen Symbol n_S und dem Namen n des zugehörigen Objekts oder Agenten. Darin bilden w_0, w_g den initialen Wert und den Zielwert der Zustandsvariablen ab, indem sie ein Element der Domäne D_{ns} definieren. Weiterhin beschreibt k einen Kommunikationskanal, über den der momentane Wert einer Zustandsvariable durch das Softwaremodul der Prozessüberwachung über das Wahrnehmungsnetzwerk ermittelt werden kann (vgl. Abbildung 26).

Die Beschreibung eines Agentenzustands wird nun am Beispiel des Greifers erklärt. Hierzu erfolgt die Bildung zweier Elemente von Z und die Definition zweier Zustandsvariablen wie folgt:

$z_1 = \left(x_1, w_{0,1}, w_{g,1}, k_1, D_{\text{GeräteStatus}} \right)$ und $z_2 = \left(x_2, w_{0,2}, w_{g,2}, k_2, D_{FingerStatus} \right)$ mit

$x_1 = \text{GeräteStatus}\left(\text{Greifer}\right)$ und $x_2 = \text{FingerStatus}\left(\text{Greifer}\right)$

Zur Beschreibung des Greiferzustands sollen zuerst die Domänen der Zustandsvariablen beschrieben werden. Diese sind in Tabelle 6 und Tabelle 7 aufgeführt.

$$D_{\text{GeräteStatus}} = \{\text{Werkzeugbahnhof,Deaktiviert,Aktiviert, Initialisierung,Bereit,Fehler}\}$$

$$D_{\text{FingerStatus}} = \{\text{Unbekannt,Geöffnet,Geschlossen,inBewegung,Zielposition,Gegriffen}\}$$

Durch die Zuweisung der Domänen zu den Zustandsvariablen $x \mapsto D_{ns}$ sowie die Zusammenfassung der Zustandsvariable in der Menge $X_n = \{x_1, x_2, ..., x_k\}$ können die möglichen Zustände eines Objektes oder Agenten repräsentiert werden.

Tabelle 6: GeräteStatus Domäne des Greiferagenten

Wert	**Beschreibung**
Werkzeugbahnhof	Dieser Wert beschreibt den Fall, dass der Greifer sich an seinem Platz im Werkzeugbahnhof befindet.
Deaktiviert	Der Greifer ist am Roboterflansch gerüstet, jedoch noch nicht aktiviert.
Aktiviert	Der Greifer ist am Roboterflansch gerüstet und der GSA hat diesen durch Steuerungsbefehle aktiviert.
Initialisierung	Der Greifer ist am Roboterflansch gerüstet und der GSA hat diesen durch Steuerungsbefehle zur Initialisierung instruiert.
Bereit	Der Greifer ist am Roboterflansch gerüstet und nun zur Verwendung bereit.
Fehler	Der Greifer ist in einem Fehlerzustand.

Tabelle 7: FingerStatus Domäne des Greiferagenten

Wert	**Beschreibung**
Unbekannt	Die Position der Finger ist unbekannt, z. B., wenn der Greifer nicht gerüstet oder aktiviert ist.
Geöffnet	Dieser Zustandswert steht für vollständig geöffnete Greiferfinger.
inBewegung	Dieser Zustandswert sagt aus, dass die Finger des Greifers in Bewegung sind.
Geschlossen	Dieser Zustandswert steht für vollständig geschlossene Greiferfinger.
Zielposition	Dieser Zustandswert bedeutet, dass die Greiferfinger eine Zielposition durch einen Steuerungsbefehl eingenommen haben.
Gegriffen	Dieser Zustandswert beschreibt, dass der Greifer über die Motorströme erkannt hat, dass ein Objekt gegriffen wurde.

Die Bildung des angenommenen initialen Zustands $X_n(s_0)$ eines Objekts oder Agenten erfolgt durch die Zuweisung der initialen Werte $x_n(s_0) \leftarrow w_0$ zu jeder Zustandsvariable in $X_n(s_0) = \{x_1 = w_{0,1}, ..., x_k = w_{0,k}\}$

vereinfacht dargestellt

$$s_{0,n} = \{n_{S,1}(n) = w_{0,1}, ..., n_{S,k}(n) = w_{0,k}\} \quad (7)$$

Soll zum Beispiel angenommen werden, dass der Greifer zu Beginn des Demontageprozesses nicht am Greifer gerüstet ist, könnte der initiale Zustand des Greifers wie folgt beschrieben werden:

$$s_{0,Greifer} = \{(\text{GeräteStatus}(\text{Greifer})=\text{Werkzeugbahnhof}),(\text{FingerStatus}(\text{Greifer})=\text{Unkekannt})\}$$

Analog erfolgt die Beschreibung des Zielzustands $X_n(g)$ eines Objektes oder Agenten im Demontageprozess durch die Zuweisung der Zielwerte $x_n(g) \leftarrow w_g$ zu jeder Zustandsvariable in

$$X_n(g) = \{x_1 = w_{g,1}, ..., x_k = w_{g,k}\}$$

vereinfacht dargestellt

$$g_n = \{n_{S,1}(n) = w_{g,1}, ..., n_{S,k}(n) = w_{g,k}\} \quad (8)$$

Wird zum Beispiel gewünscht, dass am Ende des Demontageprozesses der Greifer deaktiviert ist und die Finger geöffnet sind, dann könnte der Zielzustand als Ausdruck wie folgt definiert sein:

$$g_{Greifer} = \{\text{GeräteStatus}(\text{Greifer})=\text{Deaktiviert}, \text{FingerStatus}(\text{Greifer})=\text{Geöffnet}\}$$

Wenn keine Anforderungen an den Zielzustand eines Objekts oder Agenten vorgesehen sind, kann der Ausdruck die leere Menge $g_n = \{\ \}$ definieren.

Der aus dem Zustandsmodell gebildete initiale Prozesszustand stimmt i. d. R. nicht mit dem wirklichen Zustand eines Objektes oder Agenten im Demontageprozess überein. So könnte der Greifer etwa bereits gerüstet und mit geöffneten Fingern zur Verwendung bereit sein. Um diesen momentanen Zustand eines Objekts oder Agenten im Softwaremodul der Prozessüberwachung zu ermitteln, kann jeder Zustandsvariablen ein Kommunikationskanal zugeordnet werden $x_n \mapsto k$, zum Beispiel

$x_1 \mapsto$ GreiferTopic/Status/GeräteStatus und $x_2 \mapsto$ GreiferTopic/Status/FingerStatus

Beschreibung der Parametermenge eines Objekts oder Agenten

Die Beschreibung von Parametern dient der Parametrierung von Aktionen bzw. von Aktionskommandos, die zur Aktionsausführung über das Ausführungsnetzwerk an den entsprechenden GSA gesendet werden. Die Parameter eines Objekts oder Agenten n werden in der Menge $Y_n = \{y_1, y_2, ..., y_k\}$ zusammengefasst. Ein Parameter eines Objekts oder Agenten n mit dem eindeutigen Parameternamen n_Y wird definiert durch $y_n = (n_Y(n), w, y_K, y_M)$.

Ein Parameter kann als statischer oder dynamischer Parameter definiert werden. Statischen Parametern wird ein fester Parameterwert w zugeordnet. Sie stellen damit einen Ausdruck der Form $n_Y(n) = w$ dar. Ein statischer Parameter könnte zum Beispiel die kartesische Geschwindigkeit (in mm/Sek.) des Roboters in der Form $kartGeschwindigkeit(Roboter) = 100$ beschreiben und damit einen Parameter für Bewegungsaktionen des Roboters darstellen.

Dynamische Parameter bilden variable Werte ab, die an ein Merkmal y_M einer Klasse y_K, z. B. einer Bauteilklasse $y_K \in K_B$, gebunden werden können. Dies erlaubt die Initialisierung des Parameterwerts anhand von Klasseninstanzen i durch die Zuweisungsfunktion (3) in der Form

$$n_Y(n) \leftarrow Merkmalswert(y_M, i | i \in y_K) \tag{9}$$

Dies wird zum Beispiel verwendet, um die Greiflage einer in der Demontageaufgabe $t = (i_B, i_V, p_Z)$ hinterlegten Bauteilinstanz i_B zu beziehen. Der entsprechende dynamische Parameter wäre beschrieben durch

$$Greiflage(Bauteil) \leftarrow Merkmalswert(Greiflage, i_B | i_B \in Basis).$$

Die Greiflage des Bauteils wird beispielsweise als Parameter für die Bewegungsaktion des Roboters zum Anfahren des Bauteils verwendet. Ein anderes Beispiel, um das Gewicht des zu handhabenden Bauteils zu ermitteln, wäre

$$Gewicht(Bauteil) \leftarrow Merkmalswert(Gewicht, i_B | i_B \in Basis).$$

Dieser Parameter wird etwa benötigt, damit der Roboter das Bauteilgewicht während der Handführung kompensieren kann.

Die statischen Parameter der im Prozessmodell beschriebenen Objekte und Agenten stellen somit fest definierte Demontageprozessparameter dar. Dynamische Parameter kennzeichnen variable Demontageprozessparameter, die von konkreten Objekten, z. B. vom zu demontierenden Bauteil, oder durch den Agenten bzw. das technische System bestimmt werden.

Beschreibung der Aktionsmenge von Agenten

Im Gegensatz zu Kapitel 5.2.2.2 erfolgt keine Beschreibung von Aktionsvorlagen im Prozessmodell, sondern direkt die Definition von Aktionen in $A_n = \{a_1, a_2, ..., a_k\}$. Weiterhin wurde die Aktionsbeschreibung von Kapitel 5.2.2.2 um ein Kommando a_K, einen Kommunikationskanal k und einen Transitionseffekt (trans) zu folgender Form erweitert:

act($y_1, y_2, ..., y_k$)
Kommando: a_K
KomKanal: k
pre : $x_1, x_2, ..., x_m$
trans: $e_1, e_2, ..., e_n$
eff : $e_1, e_2, ..., e_o$
cost : a_c

Der eindeutige Name act der Aktion dient der eindeutigen Identifikation der Aktion. Die Parameter y der Aktionen können von jedem im Prozessmodell definierten Objekt oder Agenten stammen und dienen mit dem Aktionskommando a_K zur Erstellung des Aktionskommandos[29]. Um dieses über das Ausführungsnetzwerk an den entsprechenden Agenten zu senden, wird der Kommunikationskanal k in der Aktion hinterlegt.

Der Transitionseffekt ist optional und beschreibt, wie sich der Prozesszustand bei der Aktionsausführung verändert, und dient zur Erkennung der Aktionsausführung im Softwaremodul ‚Steuerung des Demontageprozesses' (vgl. Abbildung 26).

[29] Die Bildung von Aktionskommandos wird im Kapitel 6.3.4 beschrieben.

Der Transitionseffekt wird im Planungsalgorithmus nicht berücksichtigt, da er vom Effekt einer Aktion wieder geändert wird.

Als Beispiel für die Beschreibung von Aktionen sollen nun zwei Aktionen des Greifers demonstriert werden. Um den geschlossenen Greifer vollständig zu öffnen, dient die parameterlose Aktion ÖffneGreifer(), die wie folgt definiert ist:

ÖffneGreifer()
Kommando: Open
KomKanal: GreiferTopic/Aktionskommando
pre : FingerStatus(Greifer)=Geschlossen,GeräteStatus(Greifer)=Bereit
trans: FingerStatus(Greifer) ← inBewegung
eff : FingerStatus(Greifer) ← Geöffnet
cost : 1

Eine weitere Aktion, die das Greifen eines Bauteils mit maximaler Kraft und Geschwindigkeit der Greiferfinger beschreibt, ist wie folgt definiert:

GreifeBauteilFest(*FingerPosGeschlossen*(*Greifer*),*MaxGeschw*(*Greifer*),*MaxKraft*(*Greifer*))
Kommando: Move
KomKanal: GreiferTopic/Aktionskommando
pre : GeräteStatus(Greifer)=Bereit,FingerStatus(Greifer)=Geöffnet,
Lage(Roboter)=Greiflage,Werkzeug(Roboter)=Greifer, GeräteZustand(Roboter)=Wartend,
Lage(Bauteil)=Greiflage,
trans: FingerStatus(Greifer) ← inBewegung
eff : FingerStatus(Greifer) ← Gegriffen,Lage(Bauteil) ← Greifer
cost : 1

Wie an den Vorbedingungen dieser Aktion zu erkennen ist,

– muss der Roboter mit dem Greifer gerüstet an der Greiflage des Bauteils stehen und auf weitere Befehle warten (sich also nicht bewegen);
– muss der Greifer geöffnet und bereit sein;
– muss sich das Bauteil an seiner Greiflage (Montagelage) befinden.

Diese Vorbedingungen müssen im Prozesszustand s vorliegen, damit die Aktion anwendbar ist. Der Transitionseffekt beschreibt, wie sich der Prozesszustand s bei der Aktionsausführung entsprechend nach $\gamma(s, a_T) = s'$ ändert. Zum Beispiel wird sich der Wert der Zustandsvariable in s‘ bei der Aktionsausführung zu FingerStatus(Greifer)=*inBewegung*

verändern. Der Transitionseffekt einer Aktion wird im Planungsalgorithmus nicht berücksichtigt, da er vom Effekt einer Aktion überschrieben wird. Der Effekt der Aktion beschreibt, wie sich der Prozesszustand s‘ nach der Aktionsausführung verändern wird $\gamma(s', a_E) = s''$. Beispielsweise werden sich die Werte der Zustandsvariablen FingerStatus(Greifer)=Gegriffen und Lage(Bauteil)=Greifer in s‘‘ ergeben.

Beschreibung von Formen der Arbeitsteilung bzw. der Unterstützung

Mensch und Assistenzsystem können auf unterschiedliche Arten beim Lösen einer Demontageaufgabe zusammenarbeiten. Um die Zusammenarbeit von Mensch und Assistenzsystem genau zu beschreiben, müssten die einzelnen elementaren Handlungen sowie deren Abfolge exakt definiert werden. Eine solche exakte Definition setzt aber eine bestimmte Ausgangssituation voraus. Ist diese nicht gegeben, kann die Handlungsfolge nicht ausgeführt werden bzw. würde sie nicht zum vorgesehenen Ergebnis führen. Durch die Beschreibung von festen Abläufen wäre das Assistenzsystem somit nicht in der Lage, die Handlungen an verschiedene Ausgangssituationen anzupassen. Durch den Einsatz eines klassischen Planungssystems kann das Assistenzsystem die Handlungen an die Ausgangssituation anpassen, allerdings hätte der Nutzer keinen Einfluss auf die entstehende Arbeitsteilung und könnte das Verhalten des Assistenzsystems nicht vorhersehen. Denn der Planungsalgorithmus würde die Handlungsfolge und damit die Form der Arbeitsteilung bestimmen. Ein Mittelweg, damit der Nutzer die Form der Arbeitsteilung vorgeben kann, das Assistenzsystem aber die Handlungsfolge an die Ausgangssituation anpassen kann, basiert auf dem entwickelten Ansatz, die Aktionskosten entsprechend der gewünschten Arbeitsteilung anzupassen. Im umgesetzten Planungsalgorithmus (Suche mit einheitlichen Kosten) werden Aktionsfolgen mit geringen Kosten zuerst verfolgt. Erhalten nun die Aktionen, die zu einer gewünschten Form der Unterstützung gehören, geringe Kosten und alle anderen Aktionen sehr hohe Kosten, dann wird der Planungsalgorithmus in den Lösungsraum der gewünschten Arbeitsteilung expandieren[30] und eine Aktionsfolge ermitteln, die der gewünschten Arbeitsteilung entspricht, jedoch die notwendigen Anpassungen, entsprechend der Ausgangssituation, besitzt.

[30] Die Aktionskostenanpassung gleicht dem Effekt einer Heuristik.

In einem Prozessmodell können hierzu die möglichen Arbeitsteilungen in der Menge $\mathcal{B}$ definiert werden. Ein Element b dieser Menge beschreibt eine Form der Unterstützung bzw. Arbeitsteilung zwischen Mensch und Assistenzsystem und wird durch einen eindeutigen Namen n_B und durch eine Menge von Aktionen A_B definiert. Eine Arbeitsteilung im Prozessmodell wird somit durch $b = n_B(A_B)$ beschrieben. Die Menge A_B enthält die minimale Anzahl von Aktionen, die unter optimalen Voraussetzungen zur entsprechenden Arbeitsteilung führt.

In einem Beispiel werden drei verschiedene Formen der Zusammenarbeit definiert in $\mathcal{B} = \{b_1, b_2, b_3\}$ von $p_Z^A = Aufgelegt(\mathrm{b}, \mathcal{O}, \mathcal{A}, \mathcal{B})$.

In der ersten Form b_1 leistet das RAS keine Unterstützung; der Mensch führt die Demontageaufgabe manuell aus. In diesem Fall bewegt der Nutzer seine Hand zum Bauteil (Aktion a_1), greift das Bauteil (Aktion a_2), führt das Bauteil zur Box (Aktion a_3) und legt es darin ab (Aktion a_4). Da die Bewegungen des Menschen zurzeit noch nicht dahingehend überwacht werden können, um automatisiert die Ausführung einer Aufgabe zu erkennen, wird zur Koordination mit dem RAS eine Roboteraktion ‚AktiviereWarteModus' (Aktion a_5) hinzugefügt. Diese Aktion versetzt den Roboter in einen Wartemodus, den der Mensch durch eine leichte Kraftausübung gegen die Robotermechanik über einen Sprachbefehl oder durch Drücken der Aktionstaste am Roboterflansch aufheben kann. Das RAS kann nachfolgend die nächste Demontageaufgabe planen und ausführen. Diese manuell vom Nutzer ausgeführte Demontagetätigkeit wird wie folgt definiert:

$$b_1 = manuell(\{a_1, a_2, a_3, a_4, a_5\})$$

In der zweiten Form b_2 erfolgt die Ausführung der Demontageaufgabe automatisiert durch das RAS, unter Annahme der folgenden (optimalen) Anfangssituation:

- Der Roboter steht in seiner Grundstellung.
- Der Roboter hat den Greifer gerüstet.
- Der Roboter wartet auf Befehle.
- Der Greifer ist zur Verwendung bereit.
- Die Greiferfinger sind geöffnet.
- Das Bauteil befindet sich in seiner Montagelage.

Zunächst fährt der Roboter über das Bauteil in einer PTP-Bewegung (Aktion a_6). Dann fährt er in einer Linearbewegung die Greiflage (Position und Orientierung) des Bauteils an (Aktion a_7) und der Greifer greift das Bauteil (Aktion a_8). Danach fährt der Roboter in einer Linearbewegung wieder über das Bauteil (Aktion a_9) und anschließend mit einer PTP-Bewegung über die Transportbox (Aktion a_{10}). An dieser fährt der Roboter durch eine Linearbewegung direkt über die Transportbox (Aktion a_{11}), der Greifer öffnet sich und lässt das Bauteil in die Transportbox fallen (Aktion a_{12}). Danach fährt der Roboter in einer Linearbewegung wieder über die Transportbox (Aktion a_{13}) und hat damit die Demontageaufgabe ausgeführt. Diese automatisierte Form der Unterstützung bzw. die Übernahme der Demontagetätigkeit, wird wie folgt definiert:

$$b_2 = automatisiert\left(\left\{a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}\right\}\right).$$

Die dritte Form b_3 beschreibt die Kollaboration von Menschen und dem RAS bei der Ausführung der Demontageaufgabe. Unter Annahme der oben genannten Situation wird der Roboter sich über dem Bauteil mit einer PTP-Bewegung positionieren (Aktion a_6), dann aber in den Handführungsmodus übergehen (Aktion a_{14}). Der Nutzer führt den Roboter per Hand zur Greiflage des Bauteils und deaktiviert die Handführung durch Drücken der Aktionstaste am Roboterflansch (Aktion a_{15}). Der Greifer greift das Bauteil (Aktion a_8) und der Roboter fährt mit einer relativen Linearbewegung über die Greiflage (Aktion a_{16}). Anschließend fährt der Roboter mit einer PTP-Bewegung über die Transportbox (Aktion a_{10}), nähert sich dieser durch eine kartesische Linearbewegung (Aktion a_{11}), der Greifer öffnet sich und lässt das Bauteil in die Transportbox fallen (Aktion a_{12}). Danach fährt der Roboter in einer Linearbewegung wieder über die Transportbox (Aktion a_{13}). Diese Form der Zusammenarbeit wird wie folgt definiert:

$$b_3 = kollaborativ\left(\left\{a_6, a_{14}, a_{15}, a_8, a_{16}, a_{10}, a_{11}, a_{12}, a_{13}\right\}\right).$$

Liegt die oben beschriebene Anfangssituation nicht vor, werden weitere Aktionen durch das Planungssystem der Detailplanung bestimmt und eine zur Situation passende Handlungsfolge ermittelt, zum Beispiel, indem der Greifer geöffnet oder gerüstet, aktiviert und initialisiert wird.

Aus diesen vordefinierten Formen der Arbeitsteilung kann der Nutzer in der Grobplanung des Demontageprozesses die gewünschte Form für jede Demontageaufgabe festlegen. Die selektierte Form der Arbeitsteilung wird im Prozessmodell b zugewiesen, beispielsweise für die kollaborative Arbeitsteilung

$$p_Z^A = Aufgelegt(\mathrm{b}_3, \mathcal{O}, \mathcal{A}, \mathcal{B})$$

Erstellung des Prozessmodells

Für die Erstellung oder Änderung der Prozessmodelle wurde ein Prozessmodelleditor entwickelt. Dieser unterstützt den Entwickler durch eine grafische Benutzeroberfläche bei der Erstellung neuer Prozess-, Objekt- und Agentenmodellen sowie bei der Definition der Zustände, Aktionen und Formen der Unterstützung. Im Editor weiterhin implementiert ist ein Simulationswerkzeug, mit dem die Aktionsplanung anhand variabler initialer Prozesszustände überprüft und visualisiert werden kann. Außerdem ist eine Schnittstelle für den Import sowie Export der Prozessmodelle in Form von XML-Dateien implementiert.

6.3.2.2 Initialisierung der Detailplanung

In diesem Softwaremodul[31] des IDAA erfolgt zuerst die Ermittlung einer passenden Transportbox und manueller Handwerkzeuge, wie eines Schraubenschlüssels, anhand der in der Demontageaufgabe t= (i_B,i_V,p_Z) hinterlegten Bauteilinstanz und ihrer Merkmale. Danach erfolgt die Erstellung des angenommenen initialen s_0 und des Zielprozesszustands g. Dies erfolgt auf Basis der im Prozessmodell in $\mathcal{O}$ bzw. $\mathcal{A}$ beschriebenen Objekte und Agenten n und deren Zustandsmodellen durch Bildung der initialen bzw. Zielzustände der einzelnen Objekte sowie Agenten nach Formel (7) und (8) sowie deren Vereinigung nach

$$X_P(s_0) = \bigcup_{\forall n \in \mathcal{O} \vee \mathcal{A}} X_n(s_0) \text{ vereinfacht dargestellt} \qquad s_0 = \bigcup_{\forall n \in \mathcal{O} \vee \mathcal{A}} s_{0,n} \tag{10}$$

$$X_P(g) = \bigcup_{\forall n \in \mathcal{O} \vee \mathcal{A}} X_n(g) \text{ vereinfacht dargestellt} \qquad g = \bigcup_{\forall n \in \mathcal{O} \vee \mathcal{A}} g_n \tag{11}$$

[31] Das Ansichtsfenster der ‚Initialisierung der Detailplanung' des IDAA ist in Abbildung 75 in Anhang C dargestellt.

Zum Beispiel wären der angenommene initiale Prozesszustand s_0 und der Zielzustand g in $p_Z^A = Aufgelegt(\mathrm{b}, \mathcal{O}, \mathcal{A}, \mathcal{B})$ wie folgt definiert:

$s_0 = \{$GeräteStatus(Greifer)=Werkzeugbahnhof, FingerStatus(Greifer)=Unbekannt,
GeräteStatus(Roboter)=Unbekannt, Werkzeug(Roboter)=Unbekannt,
Lage(Roboter)=Unbekannt, Status(Mensch)=Wartend,
LageRHand(Mensch)=AmKörper, LageLHand(Mensch)=AmKörper,
Lage(Bautei)=Greiflage, Lage(Transportbox)=AblegeLage$\}$

$g = \{$Lage(Bautei)=AblegeLage$\}$

Der aus dem Prozessmodell gewonnene angenommene initiale Prozesszustand s_0 wird nachfolgend an das Softwaremodul der Prozessüberwachung übergeben, aktualisiert und zurückgegeben, z. B. wie folgt:

$s_0 = \{$GeräteStatus(Greifer)=Bereit, FingerStatus(Greifer)=Geschlossen,
GeräteStatus(Roboter)=Wartend, Werkzeug(Roboter)=Greifer,
Lage(Roboter)=Grundstellung, Status(Mensch)=Wartend,
LageRHand(Mensch)=AmKörper, LageLHand(Mensch)=AmKörper,
Lage(Bautei)=Greiflage, Lage(Transportbox)=AblegeLage$\}$

Bei der Aktualisierung erhalten diejenigen Zustandsvariablen den aktuellen Zustandswert, die einen Kommunikationskanal k definiert haben. Die Zustandsvariablen, denen kein Kommunikationskanal zugeordnet ist, bilden Annahmen über den initialen Demontageprozesszustand.

Im nächsten Schritt werden alle Aktionsmengen A_n der im Prozessmodell p_Z definierten Agenten n in der Menge A zusammengefasst und die statischen sowie dynamischen Parameterwerte zugewiesen.

$$A = \bigcup_{\forall n \in \mathcal{A}} A_n$$

Eine Auswahl dieser Aktionsmenge würde folgende Form[32] besitzen:

$A = \{$ÖffneGreifer(),SchließeGreifer(),GreifeBauteilFest(0,255,255),
PTPRoboterBewegung(BGKS,0.0,100.0,200.0,0.0,0.0,0.0,0.6,überGreiflage),
LinRoboterBewegung(BGKS,0.0,100.0,150.0,0.0,0.0,0.0,100,Greiflage)$\}$

Nachfolgend werden den Aktionen in der Menge A_P entsprechend der vom Nutzer selektierten Form der Arbeitsteilung in $p_z(\mathrm{b},\mathcal{O},\mathcal{A},\mathcal{B})$ Aktionskosten zugewiesen.

Die Aktionen, die in der Aktionsmenge A_B der Arbeitsteilung b vorkommen, werden in A minimale Aktionskosten zugewiesen

$$\mathrm{A}=\left\{\text{Aktionskosten}(a) \leftarrow 1 \middle| \forall a \in \text{Aktionmenge}(b)\right\}$$

Im letzten Schritt wird das vollständig definierte Planungsproblem $\mathcal{P} = (\Sigma, s_0, g)$ mit $\Sigma = (S, A, \gamma)$ an das Softwaremodul der Aktionsplanung[33] übergeben.

6.3.2.3 Aktionsplanung

Für die Planung der Handlungen (Aktionen) der GSA bzw. des Menschen (Agenten) in der Detailplanung wurde der in Kapitel 5.2.2.6 beschriebene deterministische Planungsalgorithmus im IDAA umgesetzt. Bei der Planung wird davon ausgegangen, dass Aktionen ihren vorgesehenen Effekt hervorrufen. Tritt bei der Planausführung jedoch ein alternativer Effekt auf, z. B. durch eine nicht erfolgreich ausgeführte Aktion, dann erfolgt eine Neuplanung, um die Aktionsfolge an die neue Situation anzupassen. Durch diesen Ansatz wird ein rechenaufwendiger, nicht deterministischer Planungsprozess vermieden. Der Planungsalgorithmus generiert ausgehend vom initialen Prozesszustand s_0 einen Suchbaum[34], der die unterschiedlichen Aktionsfolgen durch Äste beschreibt. Der Ast bzw. die Aktionsfolge, die zuerst den Zielprozesszustand generiert, stellt die Lösung des Planungsproblems als Plan $\Pi = \langle a_1, a_2, ..., a_n \rangle$ dar. Zusätzlich wird vom Planungsalgorithmus die zu erwartende Abfolge von Prozesszuständen S_A in der Form

[32] Im Anhang D werden die Aktionen der einzelnen technischen Systeme des RAS und deren Parameter in beschrieben.

[33] Das Ansichtsfenster der Aktionsplanung des IDAA ist in Abbildung 77 in Anhang C dargestellt.

[34] In Anhang C in Abbildung 78 ist ein solcher Suchbaum abgebildet.

$S_A = \langle s_0, s_{T,1}, s_1, s_{T,2}, s_2, ..., s_{T,n}, s_n \rangle$ worin $s_{T,i} = \gamma(s_{i-1}, a_{T,i})$ und $s_i = \gamma(s_{i-1}, a_{E,i})$ gebildet,

die auch die Prozesszustände s_T enthält, die durch den Transitionseffekt einer Aktion gebildet werden. Die Aktions- und Prozesszustandssequenz bilden das Prozessablaufmodell $P_A = (\Pi, S_A)$, das an das Softwaremodul der Prozesssteuerung übergeben wird.

6.3.3 Softwaremodul der Prozessüberwachung

Das Modul der Prozessüberwachung[35] vom IDAA dient zur Wahrnehmung des aktuellen beobachtbaren Prozesszustands. Das Modul bietet zwei Funktionalitäten. Die erste Funktion dient der Aktualisierung des initialen Prozesszustands s_0 für die Planung der Handlungen in der Detailplanung. Die zweite Funktion dient zur Koordination der Handlungen und der Überwachung des Demontageprozesses bei der Aktionsausführung.

Für die erste Funktion erhält dieses Modul den angenommenen initialen Prozesszustand s_0, der auf Basis des Prozessmodells nach Formel (10) in der Initialisierung der Detailplanung gebildet wurde. Anhand der Kommunikationskanäle K, die den Zustandsvariablen im Prozesszustand X_P zugeordnet wurden, kann im Softwaremodul der Pro-zessüberwachung ein Hintergrundprozess gestartet werden, der die in K definierten Kanäle des Wahrnehmungsnetzwerks abonniert.[36] Der Hintergrundprozess erhält nachfolgend über das Wahrnehmungsnetzwerk Veränderungen der Zustandsvariablenwerte in einer ereignisbasierten Form. Um den initialen Prozesszustand zu ermitteln, werden zuerst alle Zustandsvariablenwerte im Prozesszustand s_0 wie folgt aktualisiert:

$$s_0 = \{ x \leftarrow Wahrnehmungsnetzwerk(k) \mid \forall x \text{ mit } x \mapsto k \}$$

[35] Das Ansichtsfenster der Prozessüberwachung des IDAA ist in Anhang C in Abbildung 76 dargestellt.

[36] Der Begriff wird im Anhang E im Kontext des verwendeten MQTT Kommunikationsprotokoll erklärt.

Ein konkretes Beispiel

s_0 = {GeräteStatus(Greifer) ← Wahrnehmungsnetzwerk(GreiferTopic/Status/GeräteStatus),
FingerStatus(Greifer) ← Wahrnehmungsnetzwerk(GreiferTopic/Status/FingerStatus),
GeräteStatus(Roboter) ← Wahrnehmungsnetzwerk(RoboterTopic/Status/GeräteStatus),
...}

Für die zweite Funktion des Moduls erfolgt eine durch Ereignisse (engl.: *events*) ausgelöste Bildung eines neuen Prozesszustands s. Ein Ereignis $e\left(k,(x,w)\right)$ wird durch den Erhalt einer Zustandsänderungsmitteilung (x,w) eines gerätesteuernden Agenten über einen abonnierten Kommunikationskanal k ausgelöst. Der neue Prozesszustand wird gebildet durch

$\gamma\left(s_i, e\left(k,\left(x,w\right)\right)\right) = s_{i+1}$ mit

$s_{i+1} = \left\{(x,w)\middle|\text{e beschreibt Effekt x} \leftarrow \text{w}\right\} \cup \left\{(x,w) \in s\middle|\text{e keinen Effekt für } x \text{ beschreibt}\right\}$

Der durch ein Ereignis gebildete neue Prozesszustand s_{i+1} wird anschließend an das Softwaremodul der ‚Steuerung des Demontageprozesses' ebenfalls durch Ereignisse übergeben.

6.3.4 Steuerung des Demontageprozesses

Von der Detailplanung erhält dieses Modul das für die Demotageaufgabe ermittelte Prozessablaufmodell $P_A = \left(\Pi, S_A\right)$ sowie den aktuellen beobachtbaren Prozesszustand s vom Modul der Prozessüberwachung. Für die Ausführung einer Demontageaufgabe wird hierzu ebenfalls ein Hintergrundprozess gestartet. Der Ablauf der Aktionsausführung ist in Abbildung 31 dargestellt und soll nun erläutert werden.

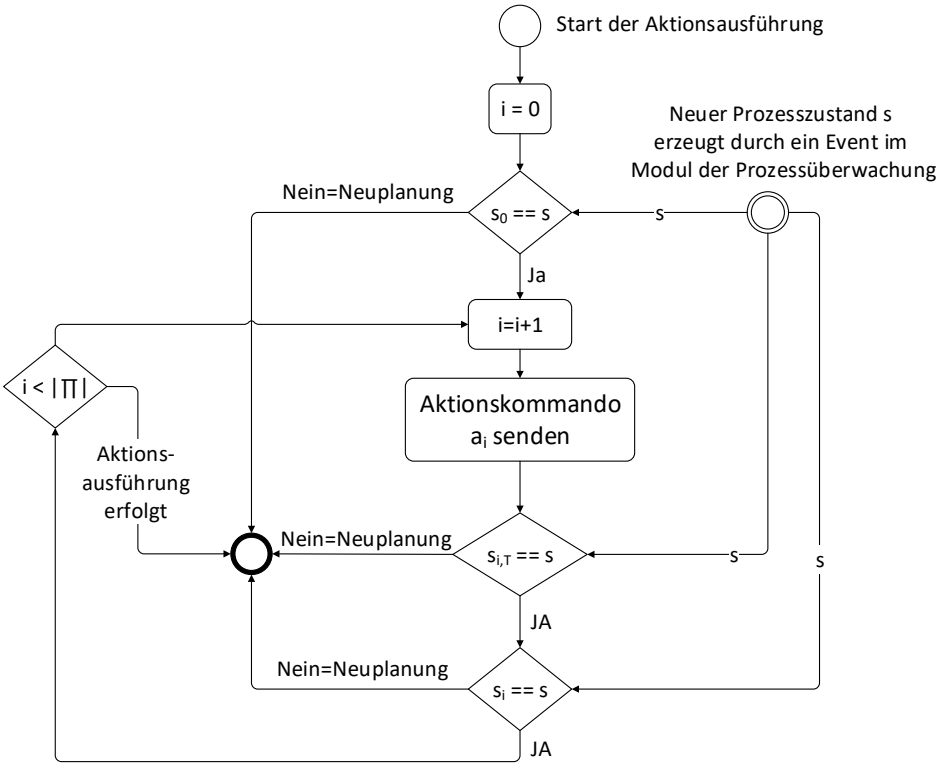

Abbildung 31: Ablauf der Aktionsausführung.

Der erste Schritt in der Aktionsausführung ist die Überprüfung, ob der beobachtbare aktuelle Prozesszustand s noch mit dem zur Planung verwendeten initialen Prozesszustand s_0 in S_A übereinstimmt. Ist dies nicht der Fall, muss eine Neuplanung erfolgen. Im anderen Fall wird zuerst das Aktionskommando c_i der Aktion a_i in Π gebildet. Dies erfolgt durch das Kommando a_K gefolgt von den kommaseparierten Parameterwerten y der Aktion a_i.

$$c_i = Kommando(a_i), Parameterwerte(a_i)$$

Für die Aktionsausführung über den entsprechenden Kommunikationskanal k_i des Ausführungsnetzwerks wird das Aktionskommando wie folgt veröffentlicht:[37]

$$Ausführungsnetzwerk(Komkanal(a_i)) \leftarrow c_i$$

Für die Aktion GreifeBauteilFest erfolgt dies zum Beispiel durch

$$Ausführungsnetzwerk(\text{GreiferTopic/Aktionskommando}) \leftarrow Move, 0, 255, 255 .$$

[37] Der Begriff wird in Anhang E im Kontext des verwendeten MQTT Kommunikationsprotokoll erklärt.

Nach Veröffentlichen des Aktionskommandos wartet der Hintergrundprozess auf den nächsten, durch das Modul der Prozessüberwachung gebildeten, Prozesszustand s und vergleicht diesen mit dem erwarteten Aktionsausführungsprozesszustand $s_{i,T}$ in S_A. Wird hier eine Abweichung erkannt, erfolgt die Neuplanung. Im anderen Fall wird auf den, durch den Effekt einer Aktion gebildeten Prozesszustand s_i in S_A gewartet. Wird dieser erreicht, kann das nächste Aktionskommando versendet werden. Der Vorgang wiederholt sich so lange, bis alle Aktionen mit dem erwarteten Effekt ausgeführt sind und der Hintergrundprozess beendet ist.

Auf diese Art lassen sich Handlungen verschiedener GSA oder des Nutzers während der Ausführung koordinieren, der Demontageprozess kann überwacht werden und auf Planabweichungen reagieren. Wenn eine Demontage ausgeführt wurde, wird die nächste Demontageaufgabe der Detailplanung übergeben. Dieser Prozess wiederholt sich, bis alle Demontageaufgaben ausgeführt wurden.

6.4 Softwarearchitektur der gerätesteuernden Agenten

Die Softwarearchitektur der einzelnen GSA sind abhängig vom gesteuerten technischen System und können sich deshalb hinsichtlich ihrer Umsetzung stark unterscheiden. Während zum Beispiel in dem Industrieroboter Kuka iiwa mittels der Hochsprache Java alle Funktionen eines GSA, bis auf die Kommunikation[38], direkt implementiert werden könnten, ist dies bei anderen technischen Systemen, wie dem verwendeten Robotiq Greifer, nicht möglich. Für diese Systeme muss ein GSA durch externe Software umgesetzt werden, der an die gerätespezifischen Eigenschaften anzupassen ist. Die grundlegenden Funktionen eines GSA können jedoch anhand der in Abbildung 32 dargestellten Softwarearchitektur beschrieben werden. In der Softwarearchitektur eines GSA erfolgt die Implementierung einer gerätespezifischen Kommunikationsschnittstelle und einer Kommunikationsschnittstelle zur Kommunikationsinfrastruktur des RAS. Innerhalb der GSA-Software werden die möglichen Gerätezustände bzw. die möglichen Zustände des beobachteten Objektes nach dem Prinzip von Zustandsvariablen $X_n = \{x_1, x_2, ..., x_n\}$ be-

[38] Die notwendigen Ports wurden durch den Hersteller gesperrt.

schrieben und den einzelnen Zustandsvariablen wird ein Kommunikationskanal zugeordnet. Die Zustandsvariablen, ihre Domänen sowie die Kommunikationskanäle müssen, mit denen im Prozessmodell übereinstimmen.

Um den aktuellen Gerätezustand s zu bilden, ist eine Abbildung p notwendig, die anhand der abrufbaren Geräteinformationen i den Zustandsvariablen die entsprechenden Werte zuweist.

$$p : i \rightarrow s_n \quad (12) \qquad \text{mit } s_n = \{x_1 = w_1, x_2 = w_2, ..., x_k = w_k\}$$

Wird der Gerätezustand kontinuierlich durch den GSA überwacht, können die Zustandsänderungen durch Vergleich des aktuellen mit dem letzten Gerätezustand erkannt und bei Änderungen auf den entsprechenden Kommunikationskanälen K des Wahrnehmungsnetzwerks veröffentlicht werden. Kann der Zustand eines Gerätes nicht überwacht werden, muss ein Zustandstransitionssystem im GSA implementiert werden, dass beschreibt wie sich der Zustand des Systems bei der Aktionsausführung verändert.

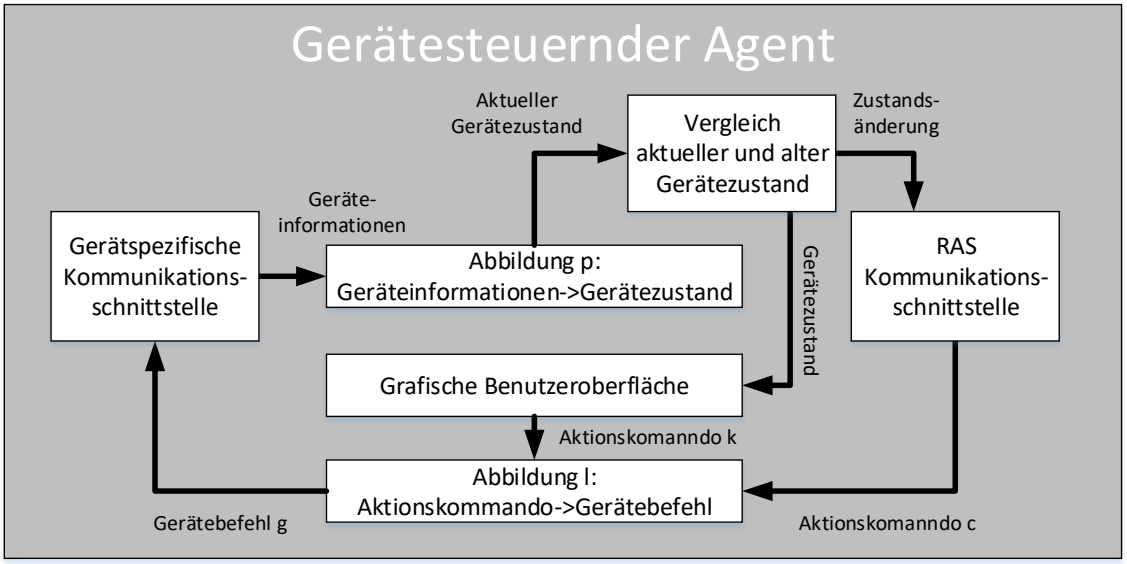

Abbildung 32: Softwarearchitektur eines gerätesteuernden Agenten.

Über die einheitliche Kommunikationsschnittstelle erhalten die gerätesteuernden Agenten Aktionskommandos c aus dem Aufgabennetzwerk des RAS. Über eine weitere Abbildung

$$l : c \rightarrow gb \quad (13)$$

wird das Aktionskommando in einen gerätespezifischen Befehl gb umgewandelt sowie über die gerätespezifische Kommunikation an das Gerät gesendet und ausgeführt. Je nach Gerät und dessen geplanten Zweck können weitere Funktionen bzw. Funktionalitäten in den GSA implementiert werden. Zum Beispiel ist es im Entwurf vorgesehen, dass GSA anderen GSA Aktionskommandos senden können, um andere Geräte im RAS zu steuern. Dies ermöglicht einen dezentralen Steuerungsansatz und somit die direkte Zusammenarbeit zwischen Geräten ohne eine übergeordnete zentralistische Steuerung. Der Nutzen liegt in der Entlastung des übergeordneten Steuerungssystems und der damit verbundenen schnelleren Reaktionsfähigkeit des RAS.

Die Abbildungen p und l sowie die Beschreibung der Zustandsvariablen sind geräte- bzw. objektspezifisch und werden in einer gerätespezifischen Wissensbasis zusammengefasst. Ergänzt wird die GSA-Software durch eine grafische Benutzeroberfläche, die es dem Menschen ermöglicht, den aktuellen Systemzustand abzulesen und direkt Kommandos an das angeschlossene Gerät zu schicken.

7 Experimentelle Validierung des robotergestützten Assistenzsystems

In diesem Kapitel wird die experimentelle Validierung des entwickelten RAS behandelt. Hierzu erfolgt zuerst die Darstellung des zur Demontage vorgesehenen Antriebssystems. Danach werden der Aufbau des Demontagearbeitsplatzes sowie die darin integrierten technischen Systeme beschrieben. Am Ende des Kapitels wird auf die konkreten Demontageprozesse zur Zerlegung des Motors eingegangen.

7.1 Das Antriebssystem

Im Rahmen der experimentellen Validierung des RAS sollte die Demontage eines mechatronischen Antriebssystems der Firma SEW Eurodrive erfolgen. Der zu demontierende Elektromotor gehört zu den innovativen mechatronischen Antriebssystemen der Produktreihe MOVIGEAR®, die zurzeit in zwei Leistungsklassen MGF4 und MGF2 verfügbar sind. Das Antriebssystem zeichnet sich besonders durch seine Modularität aus, die es dem Antriebssystem erlaubt, individuell an Anforderungen angepasst zu werden und dadurch hochgradig energieeffizient zu arbeiten. Durch die Vielzahl integrierbarer Elektronikkomponenten, wie z. B. Frequenzumrichter und Kommunikationsmodule, eignet sich das System besonders für dezentrale Anwendungen. Das Antriebssystem besteht aus drei Modulen: Elektromotor, Industriegetriebe und Kommunikations- und Steuerungselektronik, siehe Abbildung 33.

Für jedes Modul steht eine Baukastenmethodik zur Verfügung, die wiederum eine Vielzahl von Varianten erzeugt. Hinzu kommen Ausstattungsmodifikationen durch Zubehör und Optionen. Grob überschlagen werden in der MOVIGEAR-Produktreihe 167 Milliarden Varianten verwaltet. Weiter steigt die Anzahl von Varianten, wenn Lackierungen, Softwareversionen und kundenspezifische Anpassungen berücksichtigt werden. Das Produkt unterliegt weiterhin einer ständigen Veränderung durch Produktoptimierungsprozesse. Die Montage des Antriebsystems erfolgt rein manuell durch Fachkräfte in kleinen Losgrößen anhand der Baugruppestücklisten und einer Konfigurationsbeschreibung.

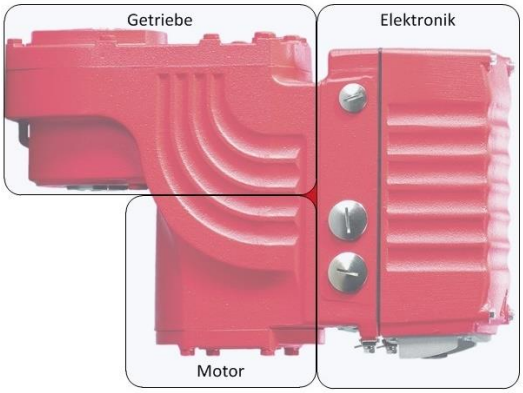

Abbildung 33: Das mechatronische Antriebssystem MOVIGEAR [173].

Im Getriebemodul befindet sich synthetisches Getriebeöl, das durch Ölablassschrauben abgelassen werden kann. Je nach Montagelage und Variante des Antriebssystems verändert sich die Position der Ölablassschraube bzw. des Entlüftungsventils. Die mit Getriebeöl benetzten Bauteile sind schwieriger zu handhaben, da die notwendige Reibung zwischen Greifer und Bauteil verloren geht. Ein sicherer Griff der ölbenetzten Bauteile wäre nur durch Formschluss zu erreichen, bei der Handhabung würde es allerdings zu einer Verunreinigung des Arbeitsplatzes, zu Rutschgefahr und zur Exposition des Nutzers mit dem gesundheitsgefährdeten synthetischen Getriebeöl kommen. Aus diesem Grund ist vorgesehen, das Getriebeöl vor der Getriebedemontage über die Ölablassschrauben abzusaugen und den Getrieberaum zu spülen. Im Rahmen dieser Arbeit wurde das Getriebeöl entfernt und fachgerecht entsorgt sowie der Getrieberaum und die Bauteile vom Öl befreit.

Für die Demontage des Getriebes müssen zuerst die Schrauben am Verschlussdeckel gelöst und entfernt werden. Je nach Varianten sind dies zwölf (MGF2) oder 14 (MGF4) Schrauben, die mit geringem Anzugsmoment (<15 Nm) montiert wurden.

Der Getriebesatz des Flachgetriebes besteht, je nach Variante, aus zwei oder drei Getriebestufen, die in Lagerschalen im Antriebsgehäuse sowie im Getriebedeckel sitzen, siehe Abbildung 34. Je nach Übersetzung variieren die Anzahl der Getriebestufen sowie die Zahnräder und Ritzewellen. Durch den Lagersitz der Getriebestufen im Getriebedeckel kann dieser nicht einfach abgenommen werden. Eine Möglichkeit ist es, den Getriebedeckel mit einem Abzieher zu entfernen. Eine andere Möglichkeit besteht darin, durch Schläge auf die Abtriebswelle, den Getriebedeckel etwas vom Getriebegehäuse zu lösen. Nachfolgend kann im entstehenden Spalt ein Werkzeug, z. B. Schraubenzieher, angesetzt

und der Getriebedeckel durch Hebeln entfernt werden. In beiden Möglichkeiten kann es zu Verklemmungen kommen, jedoch bleibt beim Abziehen des Getriebedeckels der Getriebesatz geordnet im Getriebe, während beim zweiten Verfahren die Abtriebswelle am Getriebedeckel verbleibt und die anderen Getriebestufen eine unbestimmte Lage im Getriebe einnehmen. Das Abziehen des Getriebedeckels ermöglicht es deshalb, den Getriebesatz, mithilfe einer Demontagevorrichtung, vollständig zu entnehmen.

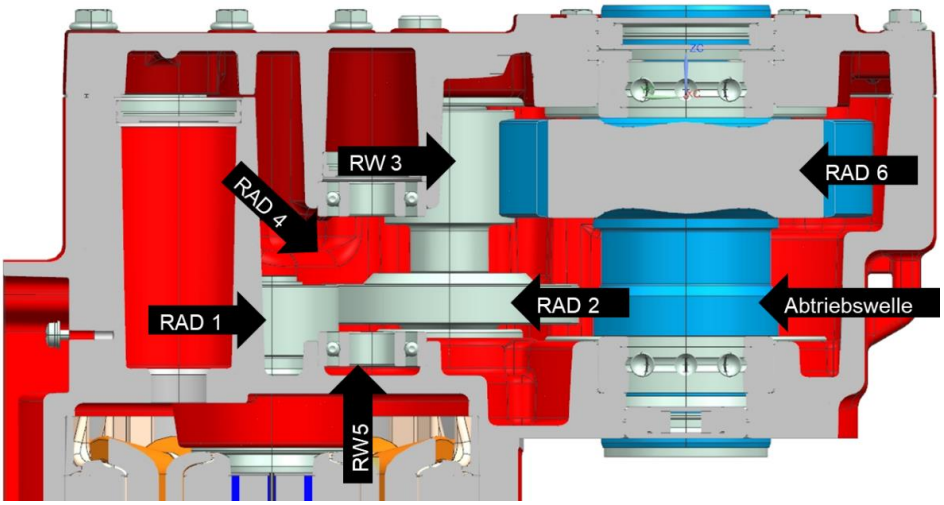

Abbildung 34: Baugruppenschnitt des Getriebemoduls.

Jede Getriebestufe (siehe Abbildung 35) besteht aus zwei Kugellagern, einer Welle bzw. Ritzelwelle und einem Zahnrad. Die Kugellager sitzen mit einer Übermaßpassung auf den abgestuften Wellenenden. Zwischen Welle und Zahnrad besteht ebenfalls eine Übermaßpassung sowie eine Welle-Nabe-Verbindung mit Passfeder. Um die Kräfte zum Auspressen der Kugellager und Zahnräder aufzubringen, wird eine hydraulische Presse mit passenden Auflagen und Pressstempel benötigt. In den Versuchen lösten sich die Kugellager und Zahnräder bei etwa 50 kN.

Das Antriebsritzel ist durch eine Presspassung, eine Klebeverbindung und eine Welle-Nabe-Verbindung (mit Passfeder) mit dem Rotor des Motors verbunden. Weiterhin sichert ein Sicherungsring die Lage des Antriebsritzels auf dem Rotor. Für die Demontage des Motors muss zuerst der Sicherungsring gelöst, dann das Antriebsritzel erwärmt und anschließend mit einem Spezialwerkzeug das Antriebsritzel abgezogen werden. An beiden Seiten der Abtriebswelle sowie am Antriebsritzel dichten Wellendichtringe den Getriebebereich ab. Im Getrieberaum können noch Druckausgleichselemente verbaut sein.

An der Getriebeseite können weitere Anbauteile angebracht werden, wie zum Beispiel eine Drehmomentstütze oder Temperatursensoren für das Getriebeöl.

Abbildung 35: Die Getriebestufen des Motors.

Das Motormodul ist in Abbildung 36 dargestellt. Der Motordeckel (2) ist mit acht Schrauben (1) befestigt, die mit einem Anzugsmoment von 12 Nm montiert wurden. Unter dem Motordeckel befinden sich eine Gummidichtung (3) und ein Inkrementalgeber(6), der mit zwei Schrauben (5) am Lagerschild (12) befestigt ist. Das Lagerschild ist mit einem Sicherungsring (4) im Gehäuse gesichert. Im Lagerschild befindet sich das Festlager (11) des Rotors(13), das durch zwei Sicherungsringe (9,10) gesichert ist. Am Ende des Rotors ist ein Magnetträger (8) für den Inkrementalgeber mit einer Schraube (7) befestigt. Das Statorpaket ist mit dem Gehäuse (15) verklebt. Für das MFG2 sowie MFG4 gibt es jeweils zwei Varianten von Drehstrommotoren. Der Rotor der Motoren ist permanenterregt und setzt sich aus einzelnen Neodym-Magneten zusammen, die mit Metallkleber mit dem Rotor verklebt sind. Das Loslager (14) des Rotors befindet sich im Getriebegehäuse.

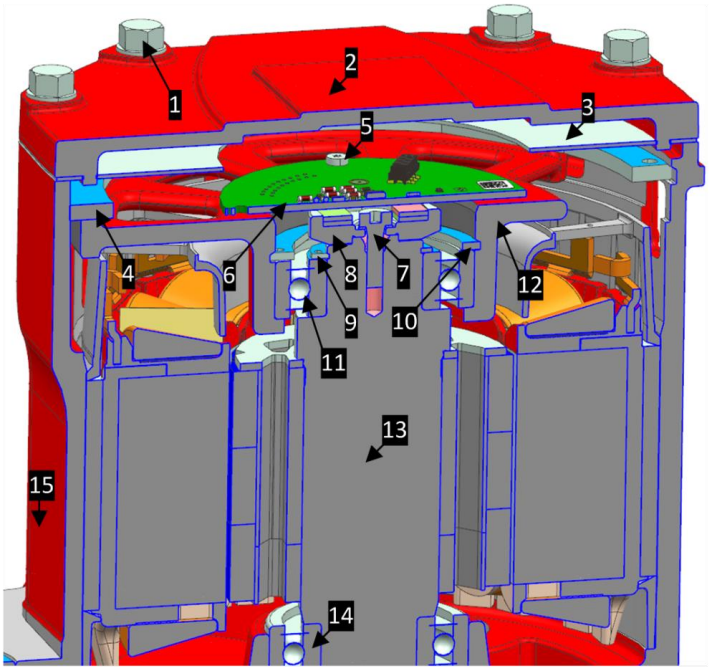

Abbildung 36: Baugruppenschnitt durch den Motor.

Die Hauptkomponenten des Elektronikmoduls sind der Elektronikdeckel, das Elektronikgehäuse, die Beschaltungsplatine sowie Stecker und Abdeckungen, die an den Kabeldurchgangsöffnungen montiert werden können, siehe Abbildung 37. Der Elektronikdeckel ist mit vier Schrauben am Elektronikgehäuse befestigt.

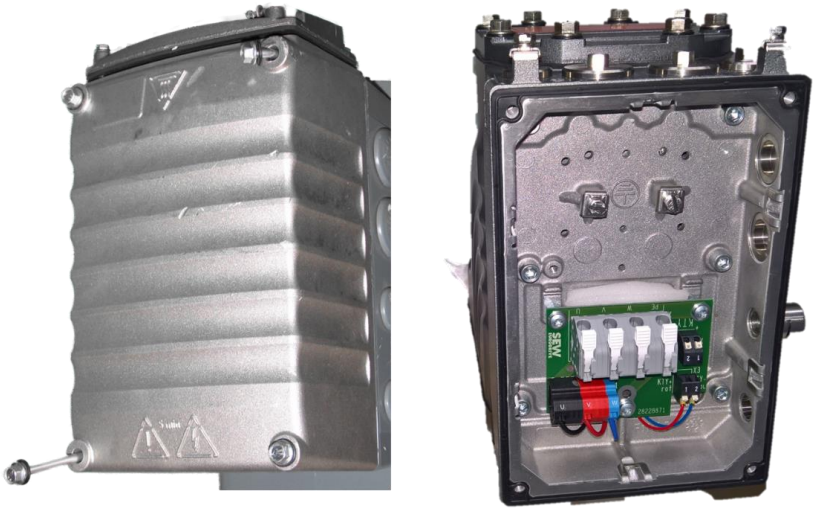

Abbildung 37: Der Elektronikkasten des Motors.

Die in Elektronikmodul untergebrachte Elektronik kann sich je nach Antriebsvariante stark unterscheiden. Bei dem vorliegenden Synchronmotor ist eine Beschaltungsplatine mit Anschlüssen für die drei Phasen des Motors sowie eines Schutzleiters verbaut. In Abbildung 37 ist auch ein Beispiel für beschädigte Bauteile dargestellt: Die Schrauben des Elektronikdeckels sind verformt und dadurch nicht automatisiert zu demontieren. Andere Varianten können im Elektronikasten dezentrale Steuerungs- und Kommunikationselektronik integrieren, die auch über frei programmierbare Ein- und Ausgänge für Sensoren oder Aktoren verfügen. Für die Verkabelung besitzt der Elektronikkasten hierzu mehrere Anschlüsse, an die unterschiedliche Stecker oder Verschlussschrauben angebracht werden können, wodurch eine hohe Varianz an Konfigurationen entsteht. Das Elektronikgehäuse wird durch sechs Schrauben mit dem Antriebsgehäuse verbunden und besitzt eine Öffnung, durch die Kabel in den Motorraum geführt werden. Zur Demontage des Elektronikgehäuses müssen daher zuerst die Kabelverbindungen gelöst werden.

7.2 Der Demontagearbeitsplatz

Der Demontagearbeitsplatz ist in Abbildung 38 dargestellt. Integriert sind zwei Industrieroboter, ein Kuka KR 125 (3) und einen Kuka Leichtbauroboter (LBR) iiwa 14 R820 mit Touch Pneumatik Flansch (15).

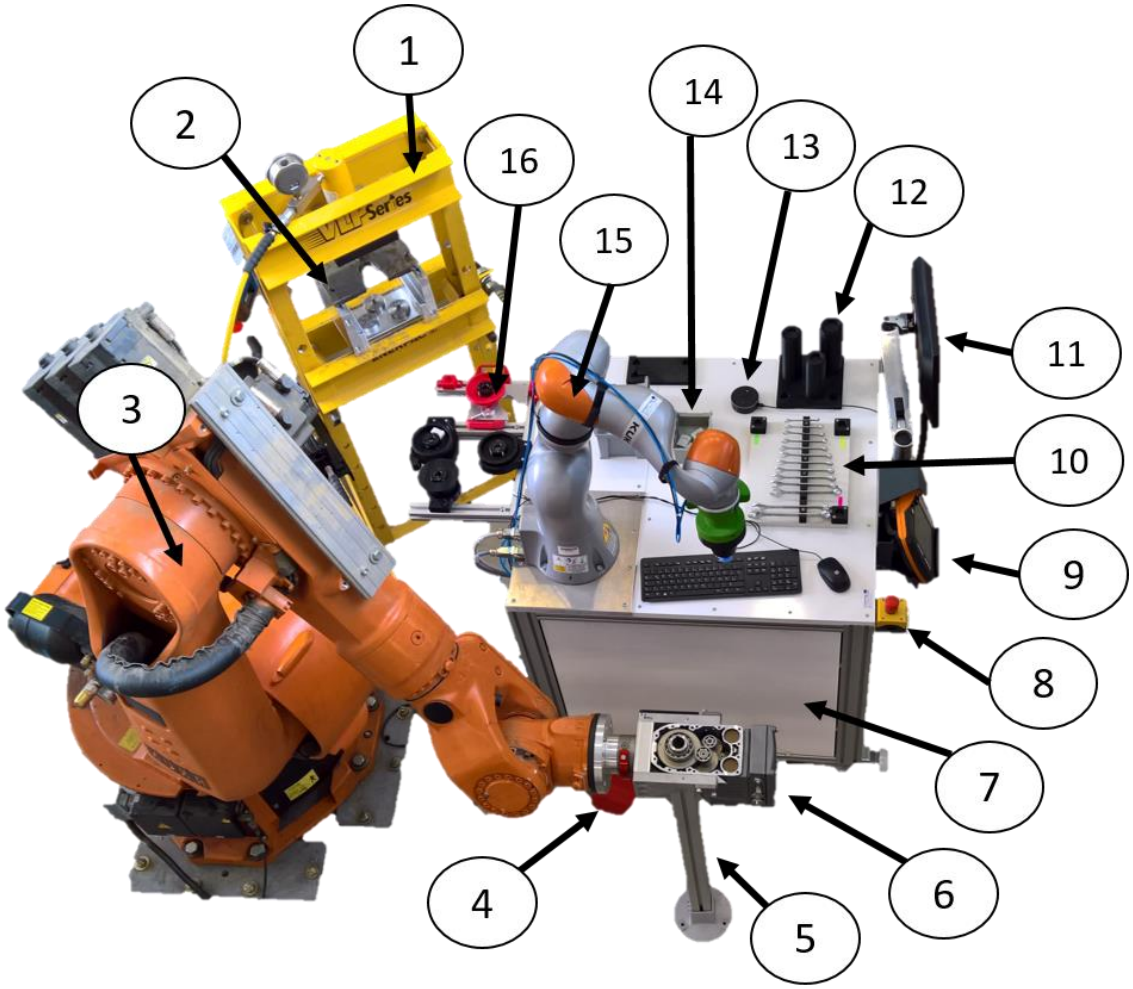

Abbildung 38: Der Demontagearbeitsplatz.

Der Kuka iiwa ist auf einer beweglichen Plattform (7) befestigt. In der Plattform befinden sich die Robotersteuerung des LBR iiwa, ein Steuerungscomputer, auf dem die Softwareagenten (die GSA sowie IDAA) sowie ein Mosquito MQTT Broker ausgeführt werden, sowie weitere Peripherie. Zum Beispiel ein WLAN-Router, Pneumatikventile, ein EtherCat-Buskoppler sowie zwei Einplatinencomputer (Raspberry Pi 3). An der Plattform sind weiterhin ein *Touchscreen* (11), das *SmartPad* (9) des LBR iiwa, ein Werkzeugbahnhof (16) und ein externer Notausschalter (8) montiert. Auf der Plattform sind außerdem

Transportkisten (14), ein Werkzeugträger (10), eine Ablage für die einzelnen Getriebestufen (12) sowie ein Amazon Echo Dot (13) positioniert. Bestandteil des Arbeitsplatzes ist zudem eine manuell bedienbare hydraulische Presse (1) für die Demontage der Getriebestufen. Zum Auspressen der Kugellager und Zahnräder der Getriebestufen wurden hierzu eine Auflage sowie Auspressdorne konstruiert und gefertigt (2). Am Kuka KR 125 ist das mechatronische Antriebssystem in einer Halterung (6) fixiert. Unter der Abtriebswelle des Antriebssystems ist weiterhin eine Auspressstütze (5) zur Demontage des Getriebes platziert. Ein roter Fußtaster (4) dient dem manuellen Wechsel von Werkzeugen. Auf die einzelnen technischen Systeme, sowie deren Integration in das RAS, soll nun näher eingegangen werden.

7.2.1 Der Kuka LBR iiwa

Der Kuka LBR iiwa 14 R820 ist ein MRK-fähiger Industrieroboter der neusten Generation. Er besitzt eine Zentralhand und verfügt mit sieben rotatorischen Achsen über einen redundanten Freiheitsgrad. Dieser erlaubt es dem Roboter, eine Position und Orientierung in seinem Arbeitsraum in theoretisch unendlich vielen Posen anzunehmen. Diese Flexibilität ist besonders in engen Räumen vorteilhaft, erfordert aber die Einführung eines zusätzlichen Redundanzwinkels zur eindeutigen Beschreibung der Roboterpose. Die abgerundete Geometrie der Robotermechanik, die interne Verkabelung sowie die Einschränkungen einzelner Achsenwinkel dienen der Sicherheit. Durch die Achswinkelbeschränkungen der vierten Achse auf ± 120° entsteht jedoch ein großer innerer, nicht erreichbarer Arbeitsraum. Auch die Achswinkelbeschränkung der siebten Achse auf ± 175° beschränkt die Bewegung ungünstig. Im LBR iiwa sind in den Antriebssystemen Drehmomentensensoren integriert. Dies ermöglicht die Messung der Antriebsdrehmomente, die Ermittlung von äußeren, auf die Roboterstruktur wirkenden Kräften und die Integration einer aktiven Impedanzregelung. Letztere macht eine nachgiebige Roboterbewegung möglich, die besonders bei Fügeaufgaben vorteilhaft ist. Auf der Basis der Impedanzregelung kann auch das Handführen des Roboters durch den Nutzer erfolgen. Der LBR iiwa 14 R820 besitzt eine maximale Nutzlast von 14 kg und einen maximalen Arbeitsraum von 820 mm. Durch die Ausstattungsvariante mit Touch-Pneumatik Flansch besitzt der Roboter einen für die Handführung notwendigen drei stufige Zustimmungstaster, eine frei programmierbare Applikationstaste, ein LED-Band zur Statusanzeige sowie zwei Pneumatik- und weitere digitale Anschlüsse am Roboterflansch. An Letzterem angebracht ist

ein pneumatisches Schnellwechselsystem mit pneumatischer und elektrischer Medienübertragung für den automatisierten Werkzeugwechsel. Die am Roboterflansch hervorstehenden Teile sowie das Schnellwechselsystem mit Adapter und die zusätzlichen Kabel- und Pneumatikleitungen wurden aus Gründen der Sicherheit durch eine Verkleidung verdeckt, siehe Abbildung 39.

Abbildung 39: Flansch des Kuka LBR iiwa.

Das Verriegeln und Entriegeln des Schnellwechselsystems erfolgt durch Pressluft über ein 5/2-Wegeventil. Das Ventil kann über den EtherCat-Buskoppler des LBR iiwa oder einen Fußschalter über ein Koppelrelais gesteuert werden. Im Werkzeugbahnhof des Kuka LBR iiwa sind die in Abbildung 40 und Abbildung 41 dargestellten Roboterefektoren untergebracht und können, bis auf das Schraubwerkzeug, automatisiert gerüstet werden.

Eine Besonderheit des Kuka LBR iiwa ist seine Steuerung, das *Kuka Sunrise Cabinet*. Diese Steuerung ermöglicht die Programmierung des Roboters in der Hochsprache Java und damit die Entwicklung komplexer objektorientierter Roboterprogramme. Als Entwicklungsumgebung für die Roboterprogramme dient eine modifizierte Eclipse IDE Indigo, die als *Kuka Sunrise Workbench* bezeichnet wird. Für die Erstellung der Roboterprogramme stehen in dieser Entwicklungsumgebung umfangreiche Bibliotheken zur Verfügung. Über die *Sunrise Workbench* erfolgt auch die Konfigurierung des Robotersystems, zum Beispiel die Definition von Werkzeugen, Werkstücken, Arbeitskoordinatensysteme sowie die Konfiguration von Sicherheitseinstellungen.

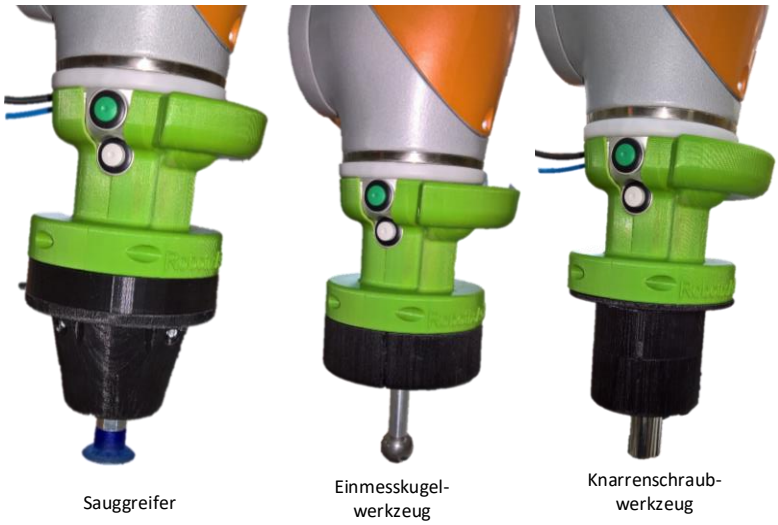

Abbildung 40: Drei Roboterwerkzeuge des Kuka LBR iiwa.

Abbildung 41: Drei weitere Roboterwerkzeuge des Kuka LBR iiwa.

Die Steuerung des LBR ist mit dem WLAN-Router über Ethernet verbunden, siehe Abbildung 42. Die gerätespezifische Kommunikation zwischen dem Kuka LBR iiwa GSA[39], der auf dem Steuerungscomputer ausgeführt wird, und dem Roboterprogramm, das auf

[39] Die Benutzeroberfläche des Kuka LBR iiwa GSA ist in Anhang D beschrieben.

der Robotersteuerung ausgeführt wird, erfolgt durch zwei unidirektionale TCP/IP[40]-Verbindungen. Die einheitliche Kommunikationsschnittstelle zwischen Kuka LBR iiwa GSA und dem RAS erfolgt auf Grundlage des MQTT-Kommunikationsprotokoll. Hierzu abonniert der GSA den Kommunikationskanal (*MQTT topic*), über den die Aktionskommandos (als MQTT Nachrichten) für den Kuka LBR iiwa gesendet werden. Erhält der GSA ein Aktionskommando über den abonnierten Kommunikationskanal aus dem Ausführungsnetzwerk, wird das Aktionskommando über die erste TCP/IP-Verbindung an das Roboterprogramm weitergeleitet und dort verarbeitet. Über die zweite TCP/IP-Verbindung werden dem GSA aus dem Roboterprogramm Änderungen des Gerätezustand oder Aktionskommandos an andere GSA übermittelt, die dann auf den entsprechenden Kommunikationskanälen (*MQTT topics*) des Wahrnehmungsnetzwerks bzw. des Ausführungsnetzwerks (als MQTT Nachricht) veröffentlicht werden. Das Roboterprogramm des LBR iiwa soll anhand Abbildung 43 nun folgend erläutert werden.

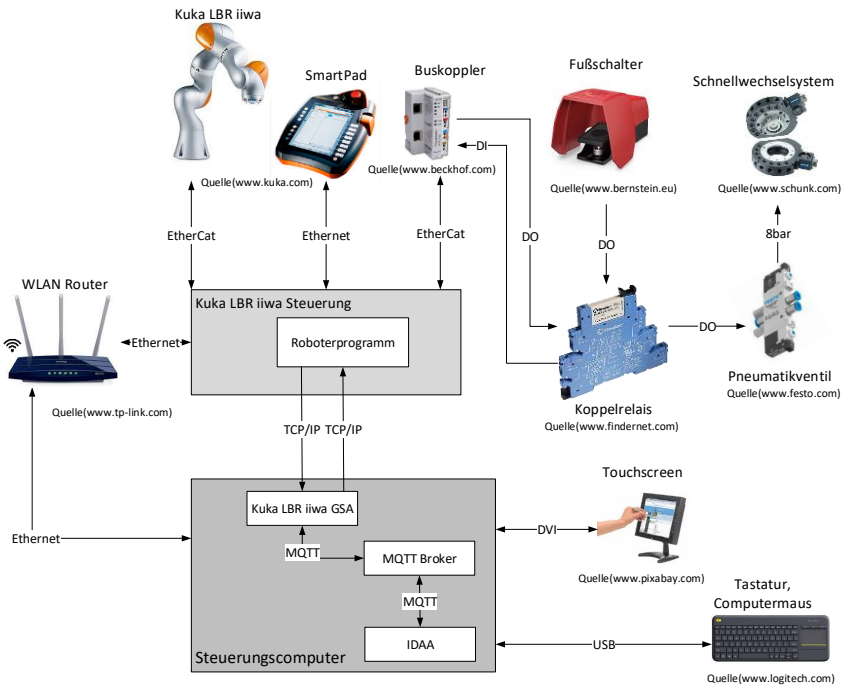

Abbildung 42: Integration des Kuka LBR iiwa in das RAS.

[40] TCP/IP steht für ‚*Transmission Control Protocol/Internet Protocol*' und dient zur Kommunikation von Rechnern in einem Netzwerk.

In der Initialisierung des Roboterprogramms werden die eingelernten und vermessenen Werkzeuge, Arbeitskoordinatensysteme und der initiale Roboterzustand erstellt. Weiterhin werden in der Initialisierung die beiden TCP/IP-Kommunikationsverbindungen mit dem GSA aufgebaut. Die erste TCP/IP-Verbindung wird in einem parallelen Hintergrundprozess gestartet, wobei ein sogenannter Zuhörer (engl. *listener*) erzeugt wird, der auf eingehende Aktionskommandos des GSA wartet. Trifft ein Aktionskommando ein, erfolgt der Aufruf der entsprechenden, im Roboterprogramm programmierten, Methode[41] und dessen Ausführung. In der Methode wird das Aktionskommando in die entsprechenden Steuerungsbefehle nach Abbildung (13) umgewandelt und ausgeführt. Weiterhin in der aufgerufenen Methode hinterlegt ist ein Teilaspekt der Formel (12), der bestimmt, wie sich die Gerätezustände bei der Ausführung der Methode verändern können. Änderungen des Gerätezustands bei der Aktionsausführung werden über die zweite TCP/IP-Verbindung an den GSA übermittelt. Nach Ausführung des Steuerbefehls wird erneut auf eingehende Steuerungsbefehle gewartet. Gestartet und beendet wird das Roboterprogramm durch den Nutzer über das *SmartPad*.

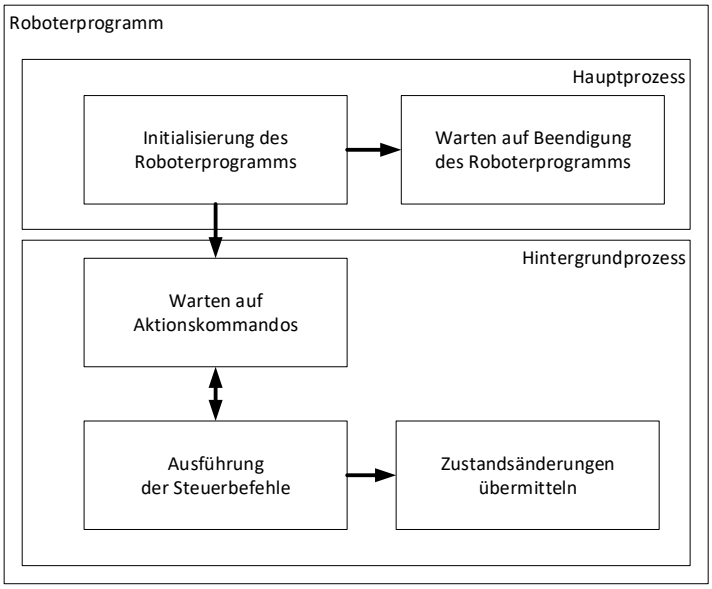

Abbildung 43: Aufbau des Kuka LBR iiwa Roboterprogramms.

[41] Im Anhang D ist in Abbildung 84 der Aufbau einer Methode im Roboterprogramm dargestellt.

7.2.2 Der Kuka KR 125

Der Kuka KR 125/2 Tj ist ein klassischer, nicht MRK-fähiger Industrieroboter mit der Robotersteuerung KRC1 (Baujahr 2001). Der Einsatz dieses Industrieroboters innerhalb des Versuchsaufbaus wurde deshalb hinsichtlich der Sicherheitsrisiken diskutiert. Da im Laborbetrieb die Programmausführung (auch des Kuka LBR iiwa) nur im Einrichtbetrieb (T1-Modus) mit reduzierter Geschwindigkeit durch speziell geschultes Personal erfolgt, wurde das Sicherheitsrisiko als vertretbar eingestuft. Die Programmierung des Roboters erfolgt in der *Kuka Robot Language* (KRL), einer proprietären, imperativen und prozeduralen Programmiersprache. Am Roboterflansch des Kuka KR 125 ist ein manuelles Schnellwechselsystem angebracht. An dieses wird das zu demontierende Antriebsystem angebracht, das hierzu fest mit einer Haltevorrichtung verbunden ist, siehe Abbildung 44. Die Handhabung des Antriebssystems durch den Kuka KR 125 beschränkt sich auf zwei Bewegungsbefehle mit kurzen Verfahrwegen und soll die Möglichkeit einer zukünftigen robotergestützten Handhabung der Motoren darstellen. Für die robotergestützte Handhabung müsste auf die Gestaltung des Antriebssystems einflussgenommen werden. Hierzu sollte im Gehäuse des Antriebssystems eine mechanische Schnittstelle für das wiederholgenaue automatisierte Rüsten vorgesehen werden.

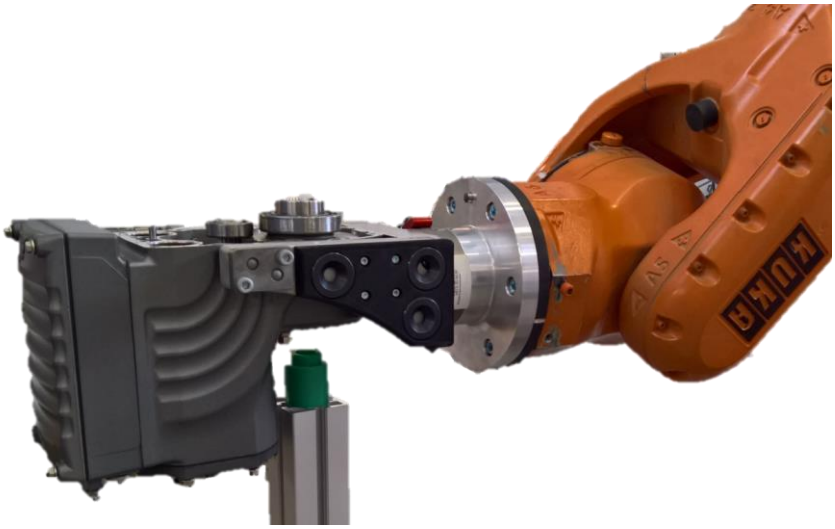

Abbildung 44: Befestigung des Antriebssystem am Kuka KR 125.

Um die Lage des Baugruppenkoordinatensystems des Antriebssystems zu ermitteln, dient eine an der Halterung angebrachten Einmessplatte mit drei Referenzpunkten. Die drei

Referenzpunkte werden mithilfe des Einmesskugelwerkzeugs von Abbildung 41 durch Handführung des LBR iiwa eingelernt, wodurch eine Transformation ${}^{R}T_{E}$ vom Weltkoordinatensystem (R) des Roboters zum Koordinatensystem der Einmessplatte (E) ermittelt wird. Die feste Transformation ${}^{E}T_{BGKS}$ vom Koordinatensystem der Einmessplatte zum Baugruppenkoordinatensystem (BGKS) des Antriebssystems wurde einmalig ermittelt. Hierzu wurden zwei Hilfskoordinatensystem ${}^{E}T_{S}$ sowie ${}^{BGKS}T_{S}$ durch drei Schraubenbohrungen am Getriebegehäuse eingemessen bzw. aus dem CAD-Modell ermittelt. Durch ${}^{E}T_{BGKS} = {}^{E}T_{S} \cdot \left({}^{BGKS}T_{S}\right)^{-1}$ konnte nachfolgend die feste Transformation berechnet werden. Durch diese Transformation kann der LBR iiwa z. B. die Greiflage der Bauteile direkt anfahren, da im Produktmodell die Position und Orientierung der Bauteile bzgl. des BGKS angegeben sind.

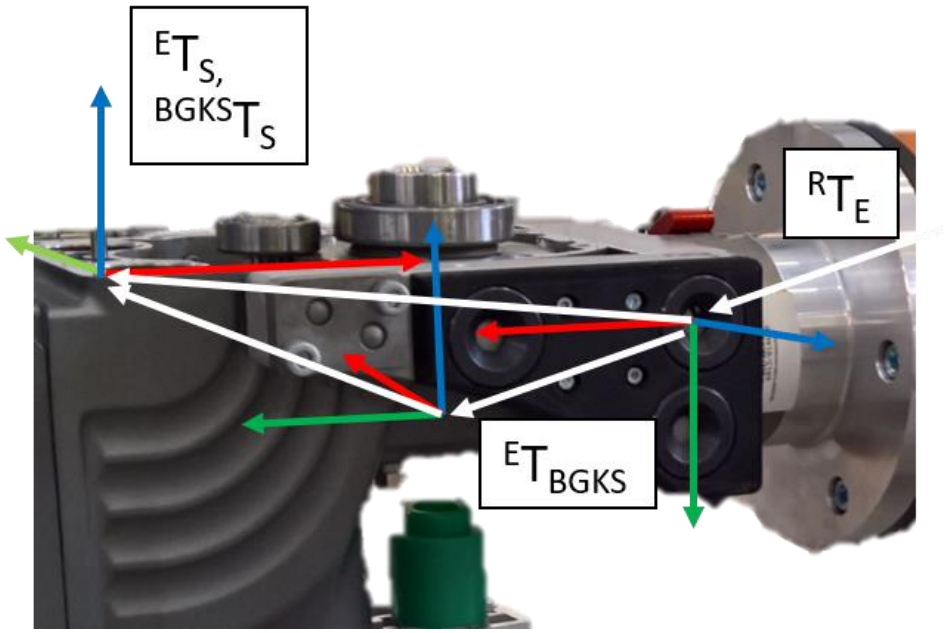

Abbildung 45: Koordinatensysteme am Antriebssystem.

Die Kommunikation mit der Steuerung des Kuka KR 125 ist eingeschränkt. Grundlegend gibt es die Möglichkeit über digitale Signale oder eine serielle RS 232 Schnittstelle mit dem Roboter zu kommunizieren. Für die entwickelten Funktionen des Kuka KR 125 war die Kommunikation durch digitale Signale ausreichend. Das Schalten der digitalen Eingänge an der Steuerung des KR 125 erfolgt durch ein WLAN- und MQTT-fähige Relaiskarte (ESP32LR88 der Firma Robot Electronics), siehe Abbildung 46.

Das klassische Roboterprogramm des KR 125 wird vom Nutzer über das Handbediengerät gestartet. Danach fährt der Roboter auf seine Grundstellung, die in Abbildung 44 dargestellt ist, und wartet mit der Programmausführung auf die entsprechenden digitalen Signale. Erhält der Kuka KR 125 GSA ein Aktionskommando (als MQTT-Nachricht) auf dem entsprechenden Kommunikationskanal (*MQTT topic*) aus dem Ausführungsnetzwerk, wandelt der GSA das Aktionskommando entsprechend der Abbildung (13) in einen Steuerungsbefehl und sendet diesen an die Relaiskarte. Diese schaltet den entsprechenden digitalen Eingang an der Robotersteuerung und das Roboterprogramm wird bis zur nächsten Programmzeile ausgeführt, in der auf ein weiteres digitales Signal gewartet wird.

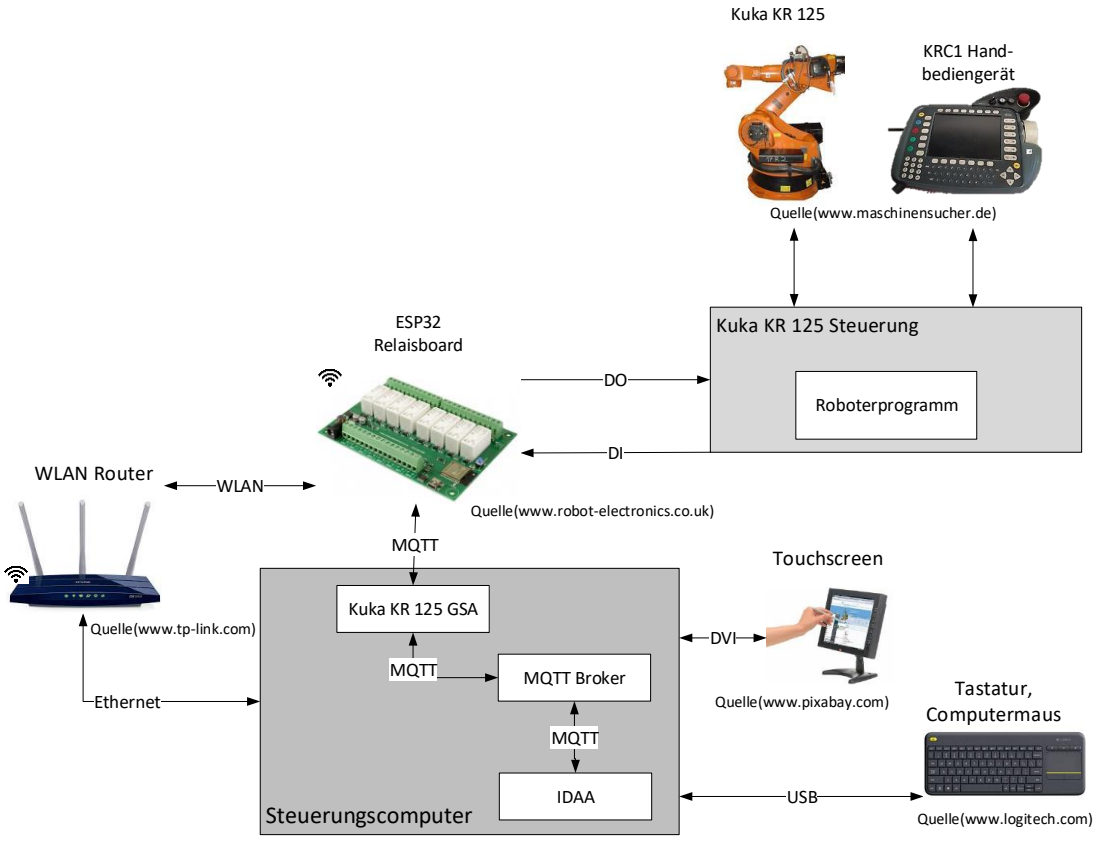

Abbildung 46: Integration des Kuka KR 125 in das RAS.

Bei der Programmausführung werden im Roboter weiterhin digitale Ausgänge geschaltet, die mit digitalen Eingängen der Relaiskarte verbunden sind. Beim Schalten eines digitalen Eingangs an der Relaiskarte wird eine MQTT-Nachricht an den GSA gesendet. Dieser wandelt diese Information entsprechend der Formel (12) in einen Gerätezustand um und

veröffentlicht die Veränderungen des Gerätezustands auf den entsprechenden Kommunikationskanälen (*MQTT topics*) (als MQTT-Nachricht) des Wahrnehmungsnetzwerks.

7.2.3 Der Robotiq Greifer

Der adaptive Zwei-Finger-Greifer 85 der Firma Robotiq besitzt eine Parallelkinematik mit einem maximalen Öffnungswinkel von 85 mm und einem Handhabungsgewicht von bis zu 5 kg. Durch die Parallelkinematik können Objekte parallel oder umfassend gegriffen werden. Die Finger des Greifers sind zudem positions-, geschwindigkeits- und kraftgeregelt. Angeschlossen ist der Greifer über einen USB-RS485-Adapter an dem Steuerungscomputer in der beweglichen Plattform, siehe Abbildung 47. Das geschirmte vieradrige Kabel zur Übertragung des differenziellen RS485-Signals sowie der 24-V-Versorgungsspannung wird entlang der Roboterstruktur an das elektrische Übertragungselement des Schnellwechselsystems geführt. Die Spannungsversorgung und Signalleitungen werden beim Rüsten des Greifers verbunden. Über das Modus-RTU-Protokoll kann der Greifer gesteuert bzw. überwacht werden, indem mit sogenannten Funktionscodes Werte in die 16-Bit-Register des Greifers geschrieben bzw. Werte aus den Registern ausgelesen werden. Für die Kommunikation mit dem Greifer wurde im Robotiq Greifer GSA die Softwarebibliothek *EasyModbus* [174] verwendet. Erhält der Robotiq Greifer GSA ein Aktionskommando (als MQTT-Nachricht) auf dem entsprechenden Kommunikationskanal (*MQTT topic*) des Ausführungsnetzwerks, wird das Aktionskommando entsprechend der Abbildung (13) in einen Steuerungsbefehl gewandelt und an den Greifer gesendet, der diesen dann ausführt. Die Überwachung des Greiferzustands erfolgt durch zyklisches Auslesen der Register in einem Hintergrundprozess. Bei jedem Lesevorgang werden die Geräteinformationen über die Formel (12) in einen Gerätezustand umgewandelt. Erkennt der GSA eine Veränderung des Gerätezustands, veröffentlicht er diesen im entsprechenden Kommunikationskanal (*MQTT topic*) (als MQTT-Nachricht) im Wahrnehmungsnetzwerk.

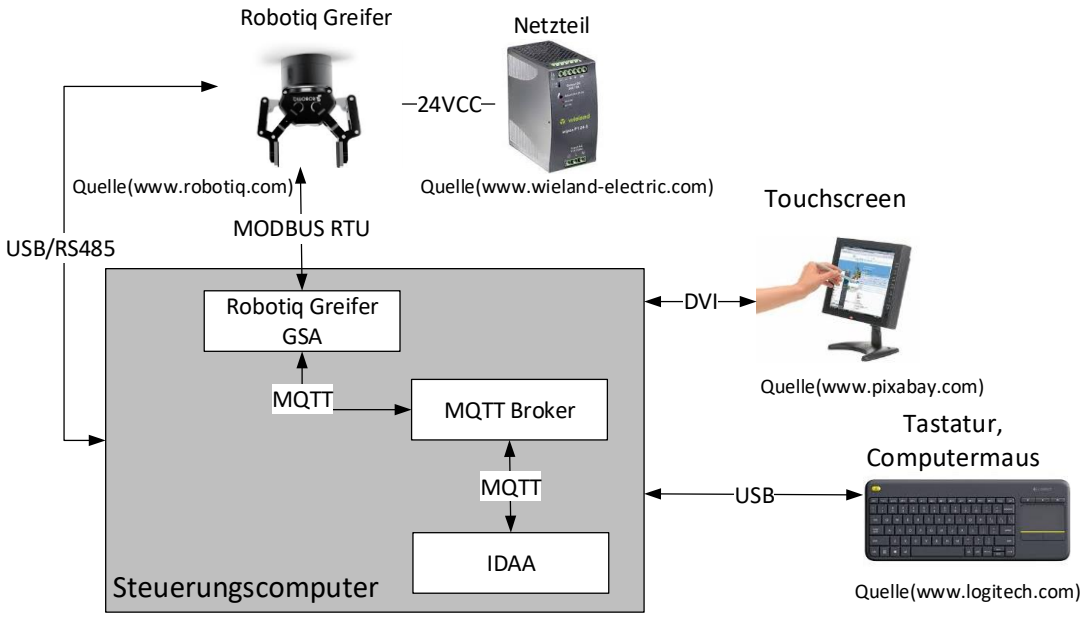

Abbildung 47: Integration des Robotiq Greifer in das RAS.

7.2.4 Der Sauggreifer

Ein weiterer in der Plattform integrierter Robotereffektor ist der Sauggreifer. Der Unterdruck zum Betrieb des Sauggreifers wird durch eine Vakuumdüse erzeugt, die über ein Pneumatikventil mit Pressluft versorgt wird. Die Pneumatikleitung läuft entlang der Roboterstruktur an das pneumatische Übertragungselement des Schnellwechselsystems. Beim Rüsten des Werkzeugs wird somit die Verbindung mit der Pneumatikleitung hergestellt. Zum Schalten des Pneumatikventils wird eine Relaiskarte verwendet, die auf einem Einplatinencomputer montiert ist. Auf diesem wird ein *Node-Red-Service* ausgeführt, der durch MQTT-Nachrichten Steuerungsbefehle vom Sauggreifer GSA zum Schalten der Relais erhält. Der Sauggreifer GSA abonniert den entsprechenden Kommunikationskanal des Ausführungsnetzwerks, erhält darüber Aktionskommandos, wandelt diese in Steuerungsbefehle um und sendet diese an den *Node-Red-Service* zur Ausführung. Entsprechend dem im Sauggreifer GSA hinterlegten Zustandstransitionssystem wird der Gerätezustand verändert und im entsprechenden Kommunikationskanal im Wahrnehmungsnetzwerk veröffentlicht.

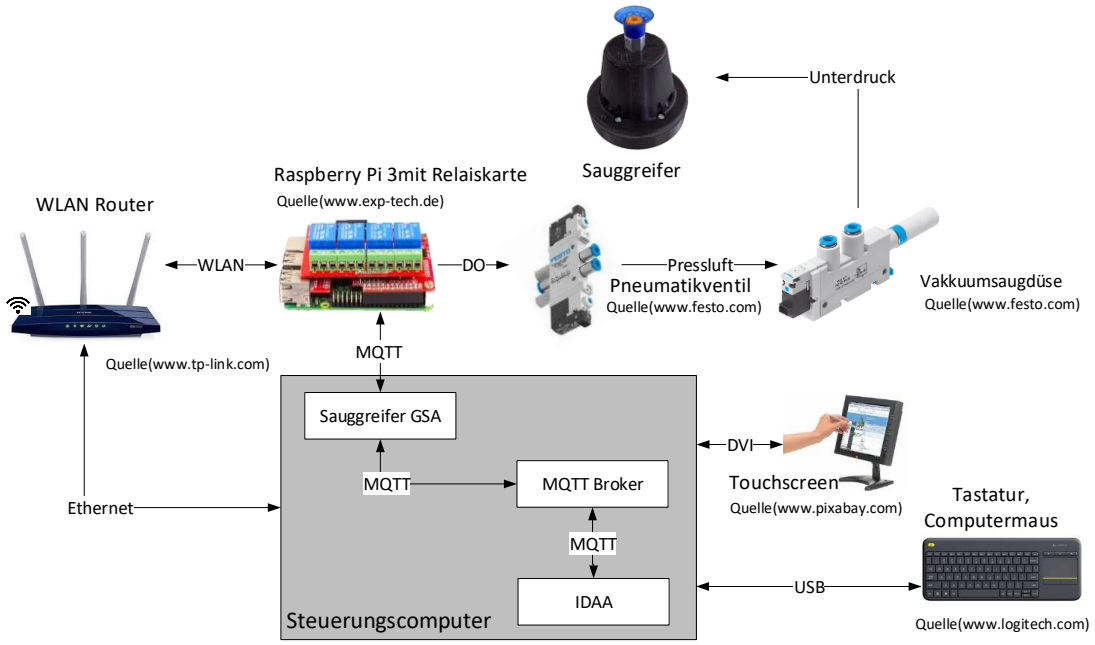

Abbildung 48: Integration des Sauggreifer in das RAS.

7.2.5 Das Schraubwerkzeug

Für die Integration des Schraubwerkzeugs wurde dieses mit einem WLAN- und MQTT-fähigen Mikrocontroller sowie mit einer mechanischen Schnittstelle für das Schnellwechselsystem des LBR iiwa ausgerüstet (vgl. Abbildung 49).

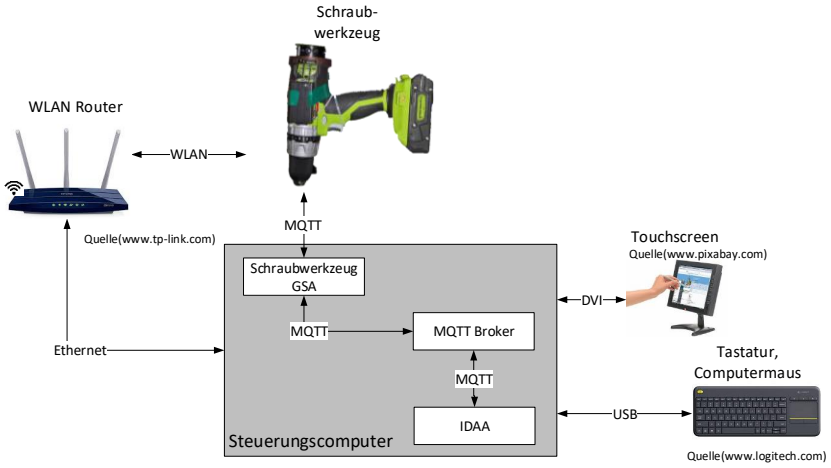

Abbildung 49: Integration des Schrauberwerkzeugs das RAS.

Durch den Mikrokontroller kann die Drehzahl des Akkuschraubers variabel durch den GSA gesteuert werden. Durch die mechanische Drehmomentbeschränkung des Akkuschraubers wird der LBR iiwa weiterhin vor hohen Drehmomenten geschützt. Die Integration des Schraubwerkzeugs durch einen GSA erfolgt nach dem bereits beschriebenen Vorgehen.

7.2.6 Der Sprachassistent Alexa

Die Integration des Sprachassistenten Amazon Alexa durch ein Amazon Alexa Dot erfolgte durch die Entwicklung eines sogenannten *Alexa Skills*. Der Sprachassistent ermöglicht dem Nutzer, mittels natürlicher Sprache mit dem RAS zu kommunizieren. Ausgeführt wird der *Alexa Skill* als *Cloud*-Applikation innerhalb der Amazon-Webservice-IT-Infrastruktur. In der momentanen Umsetzung kann der Nutzer durch den Sprachassistenten lediglich eine Auswahl von Aktionskommandos direkt an den Robotiq Greifer und den LBR iiwa senden. Die Integration weiterer Aktionskommandos, auch für die anderen Geräte des RAS sowie den IDAA, muss noch erfolgen. Für weitere Details zur Integration wird auf [96] verwiesen.

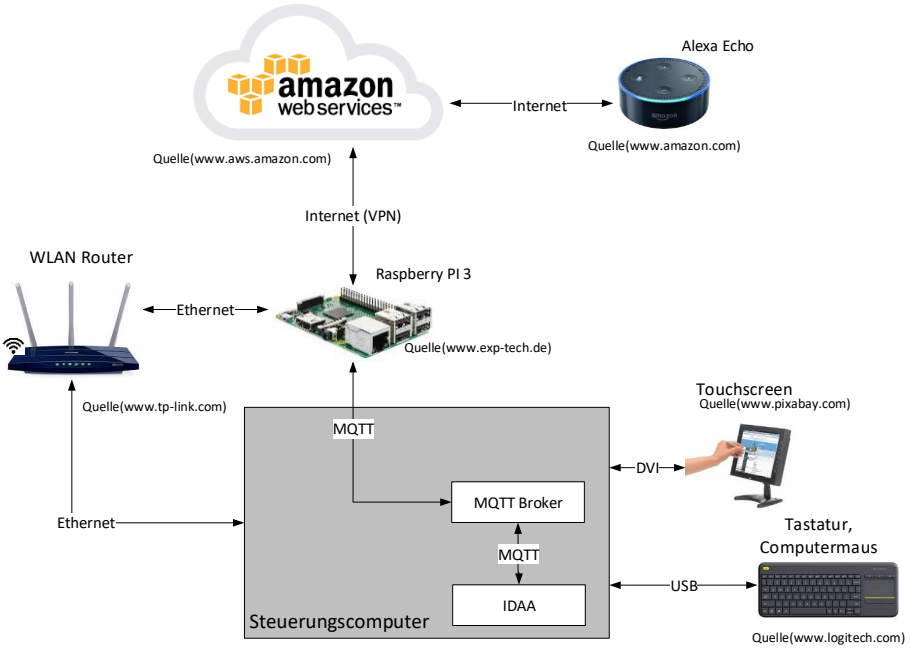

Abbildung 50: Integration des Sprachassistenten Alexa.

7.2.7 Die Microsoft Kinect

Für die Überwachung des Nutzers war die Microsoft Kinect vorgesehen. Ursprünglich wurde die Microsoft Kinect als Mensch-Computerschnittstelle für Spielkonsolen entwickelt, später folgte eine Version (*Microsoft Kinect for Windows*), die an Windows-PCs genutzt und in Softwareapplikationen mithilfe einer Programmierschnittstelle (API) integriert werden konnte. Durch die integrierten Mikrofone sowie eine Farbkamera und einen *Time-of-Flight*-Tiefensensor stellte die Microsoft Kinect ein kostengünstiges Sensorsystem für Anwendungen in der Robotik dar. Sie kann für Sprach-, Bild- und Gesichtserkennung sowie für die Umweltmodellierung und *3D-Motion-Capturing* verwendet werden. In dieser Arbeit sollte das Sensorsystem dazu verwendet werden, die Position der Hände zu überwachen, um Rückschlüsse auf den Prozessablauf zur Koordinierung der Handlungen zu gewinnen. Hierzu sollte der Ein- bzw. Austritt der Hände in und aus dynamisch erstellten virtuellen Räumen, zum Beispiel der Raum um ein Bauteil oder eine Transportbox, überwacht und damit der Zustand des Nutzers aktualisiert werden. Die Integration der Microsoft Kinect und des GSA ist wie in Abbildung 51 dargestellt vorgesehen, befindet sich jedoch noch in der Entwicklung.

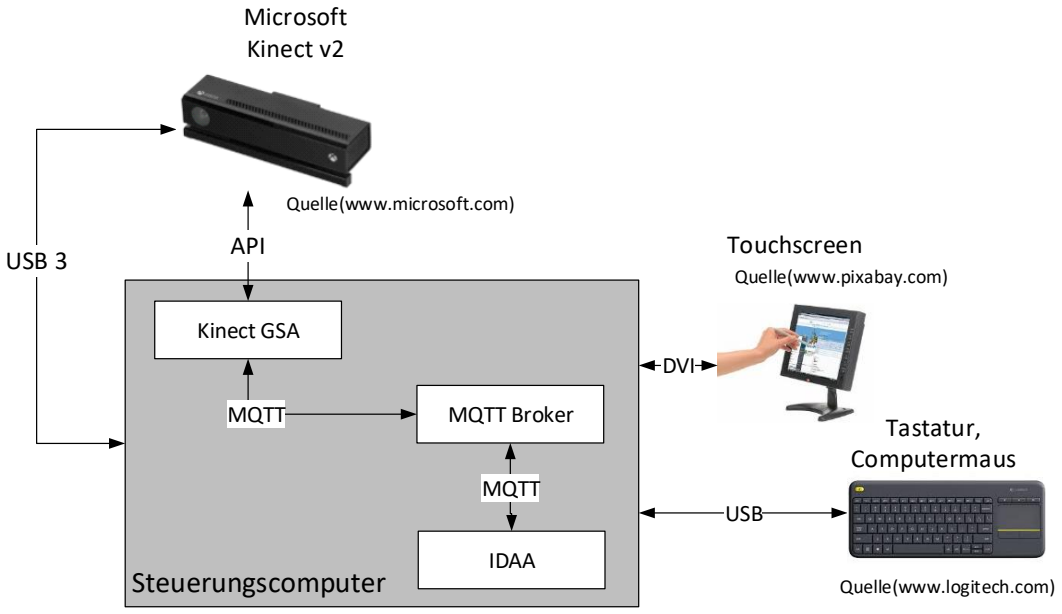

Abbildung 51: Integration der Microsoft Kinect V2 in das RAS.

7.3 Ablauf der assistierten Demontage

In diesem Kapitel soll jeweils eine Möglichkeit der Zusammenarbeit von Mensch und RAS in vier verschiedenen Demontageprozessen dargestellt werden.

7.3.1 Lösen von Schraubenverbindungen

Der LBR iiwa steht in seiner Grundstellung und erhält ein Aktionskommando zum manuellen Rüsten (T8, N29)[42] des Schraubwerkzeugs. Der Nutzer arretiert das Schraubwerkzeug manuell über den Fußtaster und bestätigt die Aktion durch Drücken der Applikationstaste am Roboterflansch. Nachfolgend fährt der LBR iiwa mittels PTP-Bewegung (T8, N4) über die Schraubenposition und dann mit einer linearen Bewegung (T8, N5) auf die Schraube. Auf der Schraubenposition erhält der LBR iiwa ein Aktionskommando (T8, N27), das ihn in Impedanzregelung versetzt. Zum Ausgleich der Schraubenbewegung und einer eventuellen leichten Exzentrizität zur Schraubachse werden die translatorischen Freiheitsgrade X, Y, Z als nachgiebig definiert. Um den Roboter weiterhin vor hohen Drehmomenten zu schützen, wird der rotatorische Freiheitsgrad um die Schraubenachse (die der Z-Achse des Schraubwerkzeugs entspricht) ebenfalls als nachgiebig definiert. Nachfolgend erhält das Schraubwerkzeug ein Aktionskommando (T11, N2) zum Lösen der Schraube. Nach Beendigung des Schraubvorgangs fährt der LBR iiwa über die Schraube (T8, N6), der Nutzer entnimmt die Schraube und legt sie in einer Transportbox ab. Das Entnehmen und Ablegen der Schraube bestätigt (T8, N21) der Nutzer nachfolgend durch Drücken der Applikationstaste am Roboterflansch, einen Sprachbefehl oder durch Kraftausübung gegen die Roboterstruktur. Der Ablauf ist in Abbildung 52 und Abbildung 53 dargestellt.

[42] Dieser Ausdruck (TX, NY) steht verkürzt für, vgl. Tabelle X, Aktionskommando Nummer Y im Anhang D.

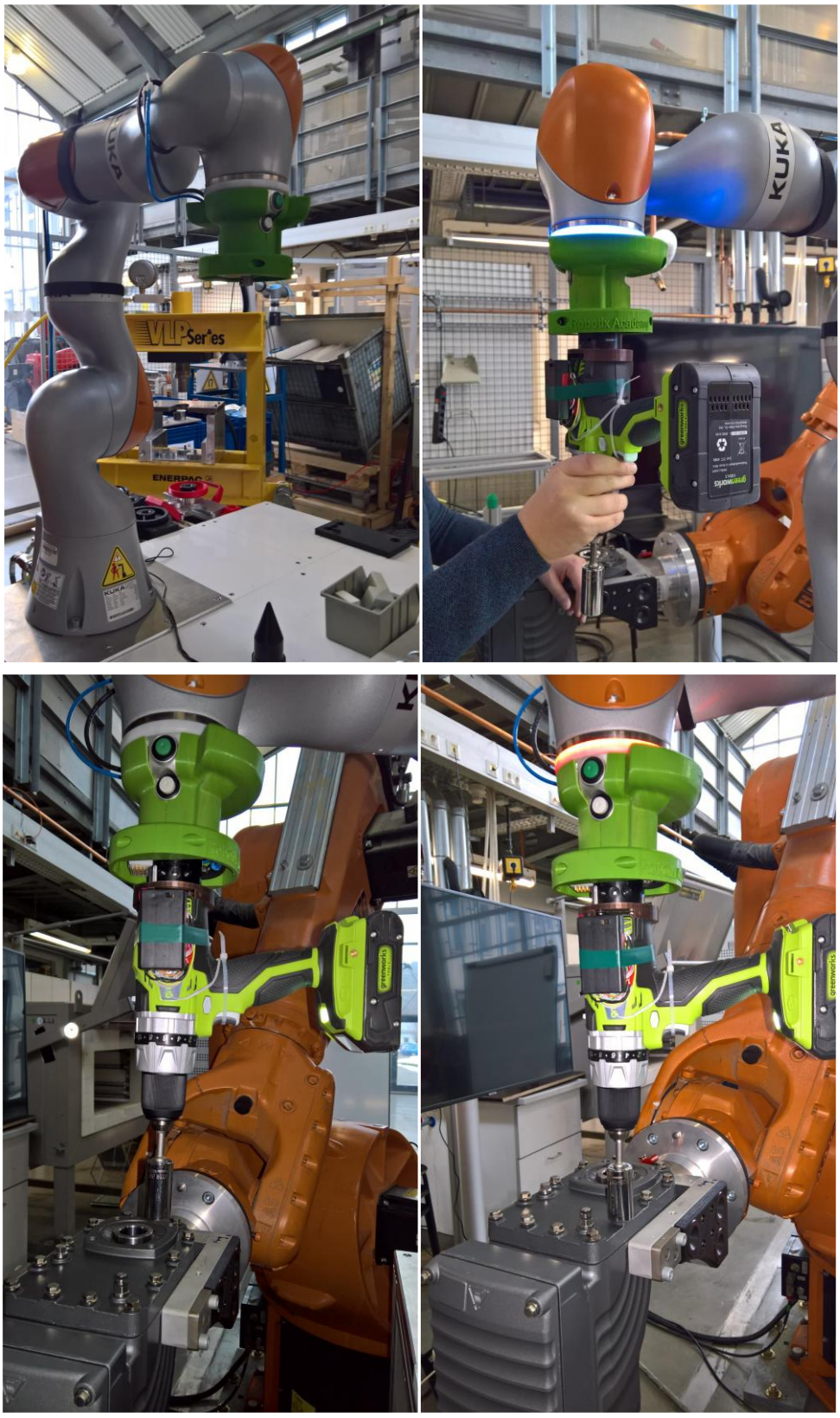

Abbildung 52: Lösen von Schraubenverbindungen, erster Teil.

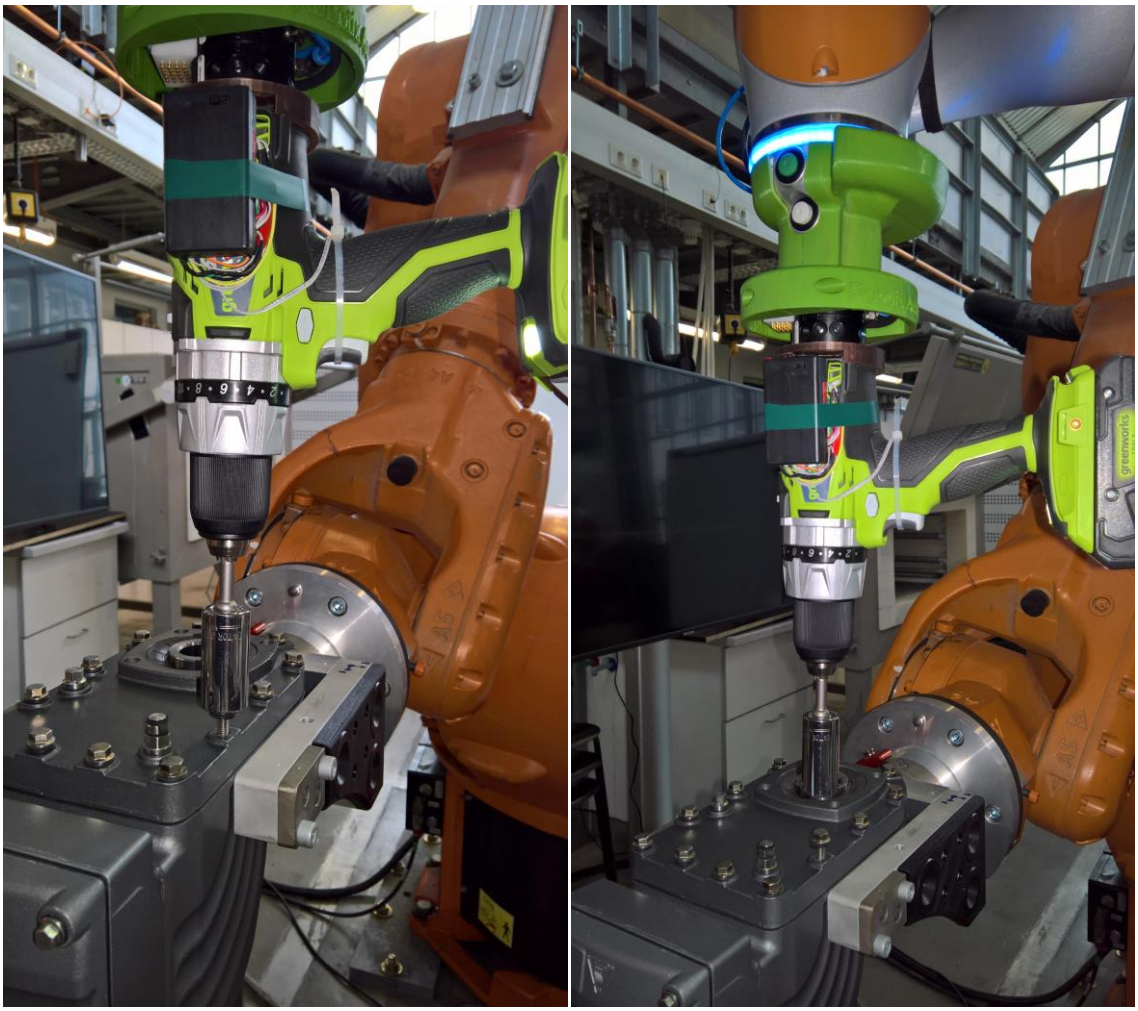

Abbildung 53: Lösen von Schraubenverbindungen, zweiter Teil.

Beim Lösen vieler Schraubenverbindungen wird hier ein Nachteil des Ansatzes deutlich. Beim Lösen jeder Schraube muss nämlich der Nutzer die Entfernung der Schraube bestätigen, damit das RAS die Demontage der nächsten Schraube startet. Günstiger wäre es, wenn das RAS zuerst alle Schrauben löst und am Schluss auf die Bestätigung des Nutzers wartet. Zur Lösung dieser Problemstellung gibt es drei Ansätze. Der erste Ansatz ist die Definition einer Assistenzfunktion mit und eine ohne Bestätigung. Somit könnten alle Schrauben durch eine Assistenzfunktion ohne warten gelöst und an der letzten Schraube die Entnahme aller Schrauben bestätigt werden. Der zweite Ansatz besteht darin, alle Schrauben einer Schraubenverbindung in einer Demontageaufgabe zusammenzufassen und sie damit gemeinsam zu demontieren. Hierzu müssten im Prozessmodell dynamisch mehrere Objektmodelle, die die einzelnen Schrauben repräsentieren, sowie Aktionen, zum Lösen der einzelnen Schrauben, erzeugt werden. In der Aktionsplanung würde dadurch jedoch ein rechenintensiver, tiefer Suchbaum entstehen. Der dritte Ansatz wäre

die Wahl einer anderen Planungsmethodik auf Basis hierarchischer Aufgabennetze (eng. *hierarchical task networks* (HTN)) [126, 127, 175]. Durch den Einsatz von HTN würde auch die Definition verschiedener Arbeitsteilungen bzw. Formen der Unterstützung strukturierter erfolgen. Im HTN-Planungsansatz wird eine Aufgabe in immer kleinere Teilaufgaben (engl. *subtasks*) zergliedert, bis letztendlich jede Teilaufgabe einer Aktion entspricht. Die Reihenfolge der Aktionen kann dabei anhand weiterer Randbedingungen angepasst werden, was hinsichtlich der oben genannten Problemstellung beim Lösen mehrerer Schraubenverbindungen von Vorteil wäre. Jedoch erfordert der Einsatz von HTN-Planungssystemen die Definition von Methoden, die eine Aufgabe in Teilaufgaben zergliedern, wodurch ein weiterer Modellierungsaufwand entsteht. Außerdem sind die Algorithmen zur Planerzeugung, die ein Bedingungserfüllungsproblem (engl. *constraint satisfaction problem* (CSP)) darstellen, wesentlich komplexer hinsichtlich der Implementierung. Die Verwendung eines frei verfügbaren HTN-Planungssystems, wie z. B. SHOP 2 [175], wäre möglich, würde jedoch neue Problemstellungen hinsichtlich der Implementierung im RAS aufwerfen.

7.3.2 Abziehen des Getriebedeckels

Zum Abziehen des Getriebedeckels montiert der Nutzer zuerst eine Abziehvorrichtung an das Antriebssystem und entrüstet den Schrauber (T8, N9) manuell. Der LBR iiwa rüstet das Knarrenschraubwerkzeug (T8, N10) und bewegt sich mit einer PTP-Bewegung (T8, N4) über die Abziehvorrichtung. Dort angekommen erhält der LBR iiwa ein Aktionskommando, das ihn in Impedanzregelung (T8, 27) versetzt. Der Nutzer führt den Roboter auf die Schraube der Abziehvorrichtung und bestätigt dies durch Drücken der Applikationstaste am Roboterflansch. Mit einer Verzögerung von zwei Sekunden beginnt der Roboter den Schraubvorgang (T8, N28). Dabei passt der Roboter seine Position ständig der Schraubenbewegung an. Der Nutzer verhindert beim Abziehen das Verklemmen des Getriebedeckels, indem er mit zwei Kunststoffkeilen durch Hebeln auf der gegenüberliegenden Seite des Getriebedeckels unterstützt. Nach dem Schraubvorgang fährt der Roboter linear über die Schraubposition (T8, N6) und dann mit einer PTP-Bewegung zur Grundstellung (T8, N2). Der Nutzer demontiert die Abziehvorrichtung und legt den Getriebedeckel ab. Der Ablauf ist in Abbildung 54, Abbildung 55 und Abbildung 56 dargestellt.

Abbildung 54: Lösen des Getriebedeckels, erster Teil.

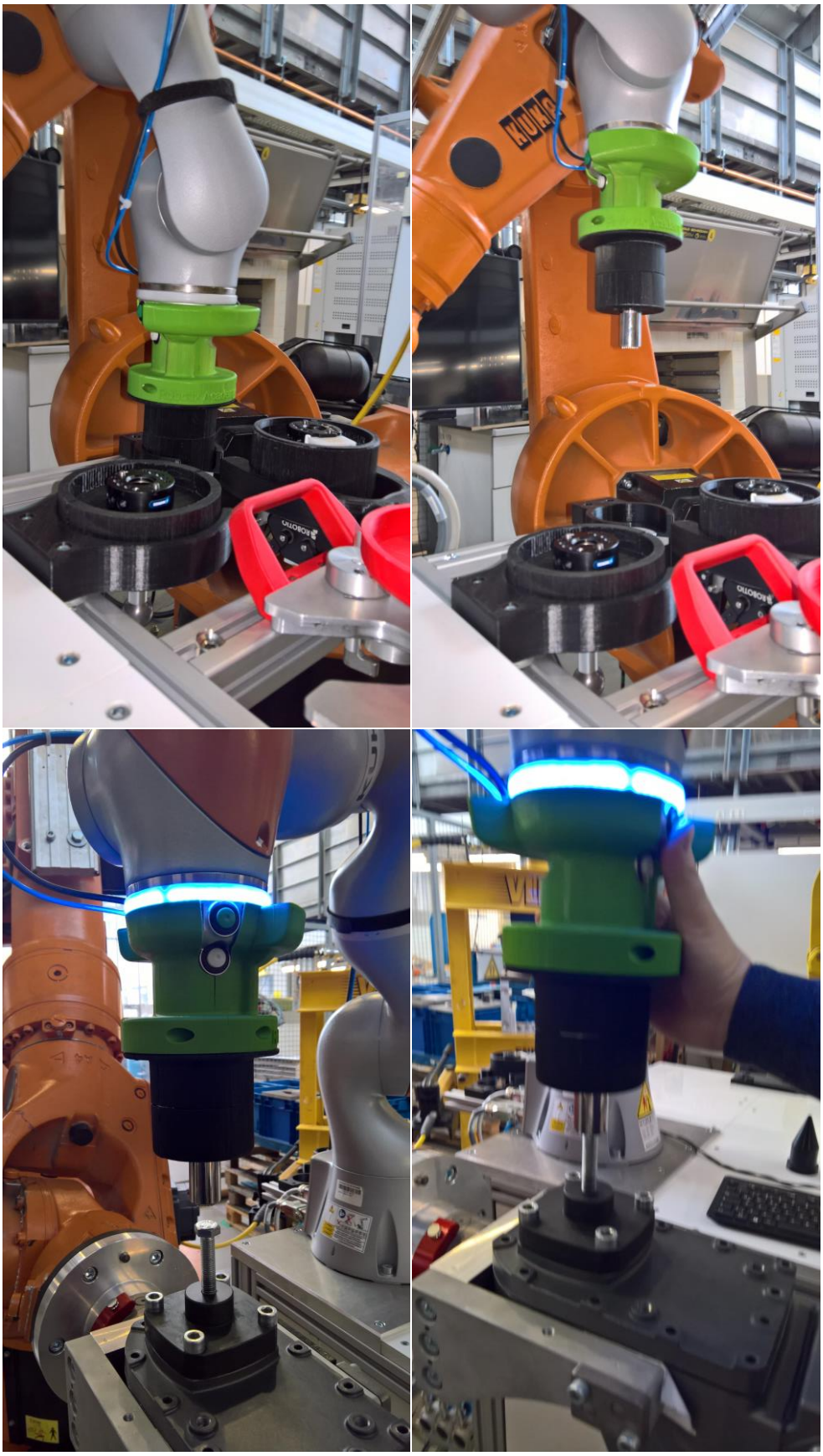

Abbildung 55: Abziehen des Getriebedeckels, zweiter Teil.

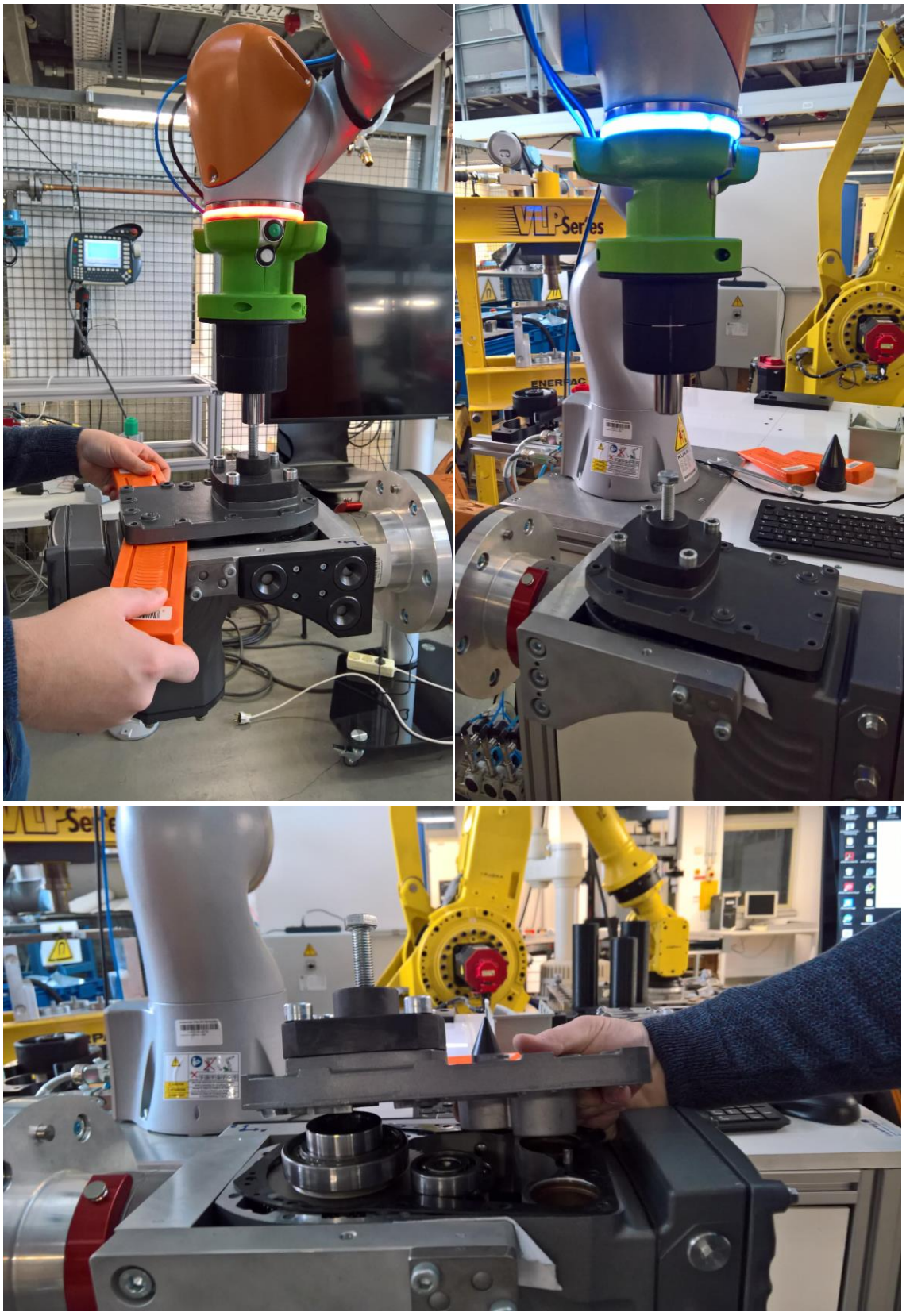

Abbildung 56: Abziehen des Getriebedeckels, dritter Teil.

7.3.3 Entnahme des Getriebesatzes

Zum Entnehmen des Getriebesatzes wechselt der LBR iiwa zuerst das Knarrenschraubwerkzeug gegen den Getriebesatzgreifer (T8, N12) und fährt anschließend über die Getriebeposition (T8, N4). Dort angekommen wechselt der LBR iiwa in Impedanzregelung (T8, N27). Der Nutzer positioniert anschließend den Getriebesatzgreifer, verriegelt diesen und bestätigt diese Handlung durch Drücken der Applikationstaste am Roboterflansch. Der LBR iiwa erhält nachfolgend über ein Aktionskommando das Gewicht der Getriebesatzbaugruppe (T8, N25) und wird in Impedanzregelung (T8, N27) versetzt. Hierzu werden alle Freiheitsgrade als nachgiebig und mit einer geringen Steifigkeit definiert. Nun beginnt der Auspressvorgang des Getriebesatzes. Hierzu fährt (T10, N1) der Kuka KR 125 mit der Abtriebswelle des Antriebssystems auf den Auspressdorn der Auspressstütze. Durch den Auspressvorgang lösen sich die einzelnen Getriebestufen aus ihrem Lagersitz im Gehäuse des Antriebssystems. Der Nutzer kann nun den Roboter, der das Gewicht des Getriebesatzes kompensiert, aus der Auspresslage führen und im freien Raum über dem Antriebssystem positionieren. Diesen Vorgang bestätigt der Nutzer durch Drücken der Applikationstaste. Mit einer Verzögerung von zwei Sekunden fährt der LBR iiwa in einer PTP-Bewegung (T8, N3) in die Nähe der Ablageposition der einzelnen Getriebestufen. Dort entriegelt der Nutzer den Getriebesatzgreifer und entnimmt die einzelnen Getriebestufen und positioniert sie auf den entsprechenden Ablagepositionen. Durch Drücken der Applikationstaste bestätigt (T8, N19) der Nutzer die Entnahme der Getriebestufen, das Bauteilgewicht wird zurückgesetzt (T8, N26) und der LBR iiwa bewegt sich mit kurzer Verzögerung mit einer PTP-Bewegung (T8, N2) wieder zur Grundstellung. Der Ablauf ist in Abbildung 57 und in Abbildung 58 dargestellt.

Abbildung 57: Entnahme des Getriebesatz, erster Teil.

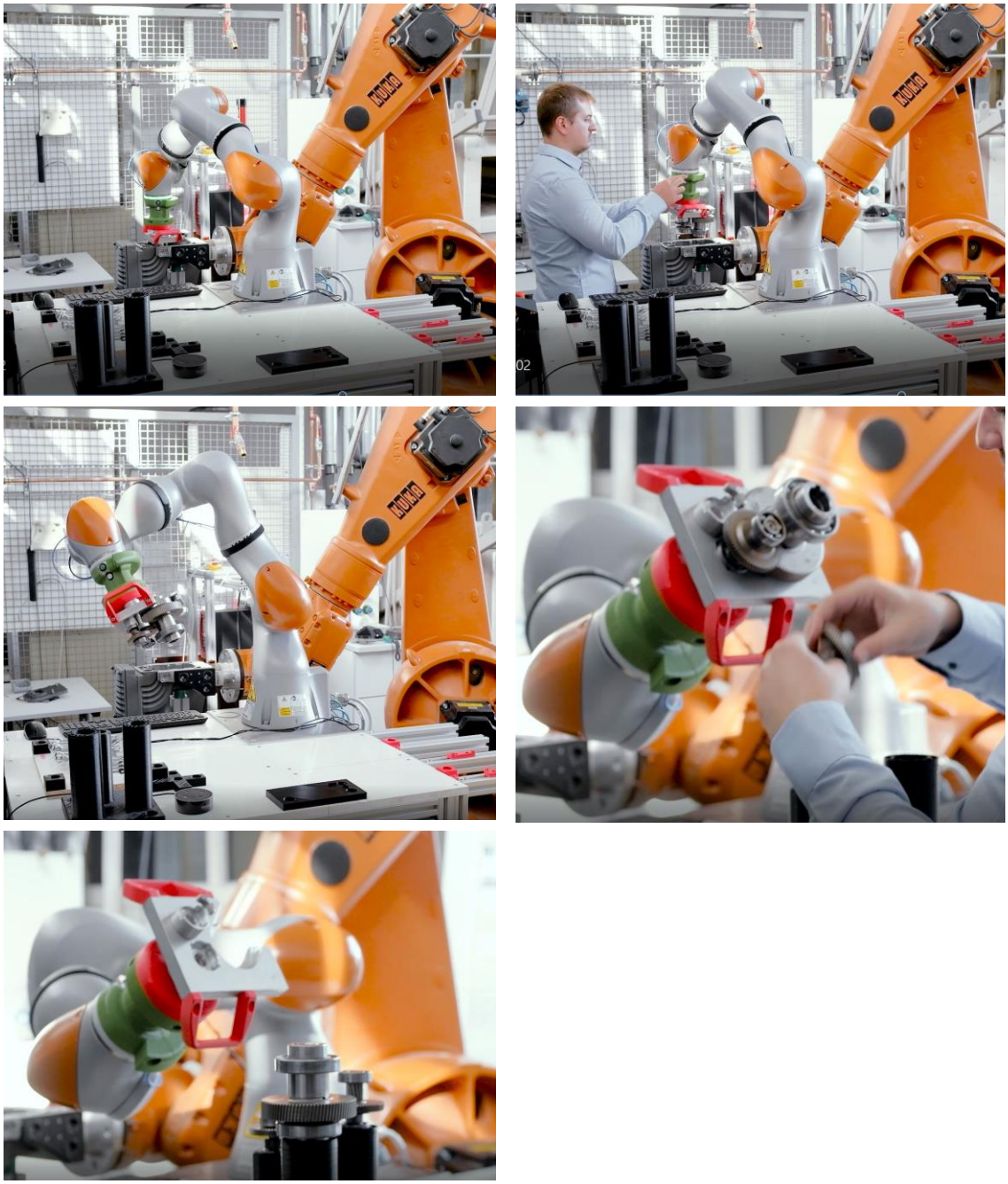

Abbildung 58: Entnahme Getriebesatz, zweiter Teil.

7.3.4 Auspressen von Kugellager und Zahnräder

Die Zerlegung einer Getriebestufe erfolgt in zwei Schritten nach dem folgenden Ablauf. Der LBR iiwa positioniert sich mit einer PTP-Bewegung vor der Getriebestufe und fährt dann die Greifposition linear an. Anschließend erhält der Greifer ein Befehl die Getriebestufe zu greifen. Danach fährt der Roboter in einer linearen Bewegung in den freien Raum über der Ablageposition. Es folgt eine zweistufige PTP-Umsetzbewegung vor die Auspressauflage in der Hydraulikpresse. Der LBR iiwa positioniert nun die Getriebestufe für den ersten Auspressvorgang zentrisch unter dem Auspresszylinder und verschiebt dabei die Auspressauflage in die richtige Position. An der Auspressposition angekommen wechselt der LBR iiwa in die Impedanzregelung. Der translatorische Freiheitsgrad in Richtung der Auspressbewegung wird als nachgiebig definiert sowie die beiden rotatorischen Freiheitsgrade um die beiden anderen translatorischen Freiheitsgrade. Dadurch kann der Roboter der Auspressbewegung folgen und eine leichte Verkippung der Lager bzw. des Zahnrads auf der Auflage ausgleichen. Der Nutzer montiert den entsprechenden Dorn auf dem Auspresszylinder und beginnt mit dem Herausdrücken durch Betätigung der Hydraulikhandpumpe. Nach Abschluss des Auspressvorgangs entnimmt der Nutzer das Kugellager und das Zahnrad und legt sie in den entsprechenden Transportboxen ab. Durch einen Sprachbefehl bestätigt der Nutzer diesen Vorgang. Nun folgt der zweite Auspressvorgang. Der Roboter fährt hierzu linear zuerst aus der Auflage hinaus und dreht den Greifer anschließend um 180°. Danach bewegt sich der LBR iiwa linear zur Auflage und legt die Abtriebswelle mit dem verbleibenden Kugellager zentrisch unter dem Auspresszylinder auf der Ablage ab. Durch eine dreistufige lineare Umsetzbewegung positioniert sich der LBR iiwa unter der Auflage und greift die Abtriebswelle. Die Impedanzregelung des Roboters wird wieder aktiviert und das verbleibende Kugellager durch den Nutzer ausgepresst. Nach Bestätigung durch einen Sprachbefehl bewegt sich der Roboter aus der Presse und übergibt die Abtriebswelle dem Nutzer. Der Demontageprozess wird in Abbildung 59 bis Abbildung 62 dargestellt.

Abbildung 59:Auspressen der Zahnräder und Kugellager, erster Teil.

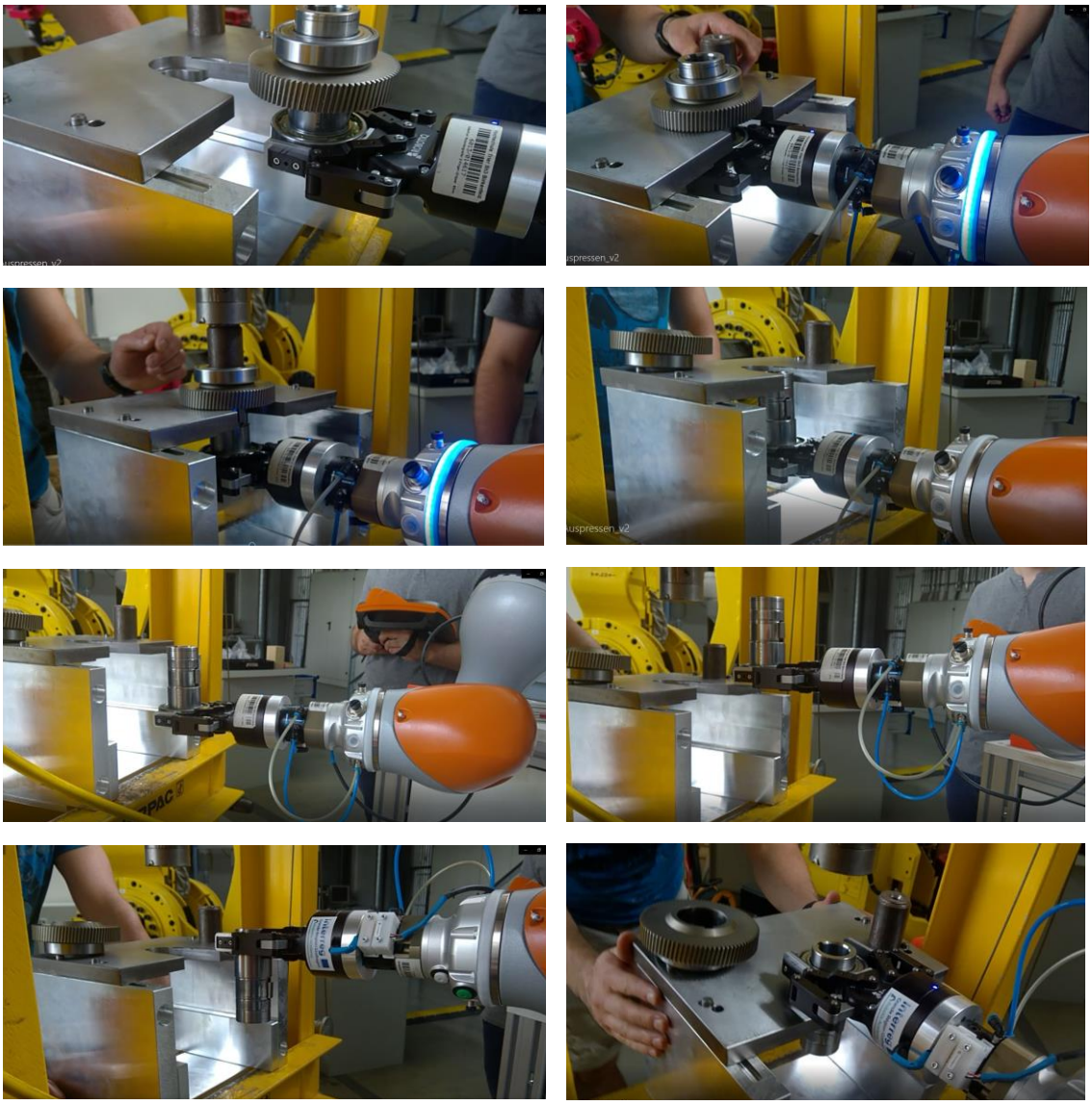

Abbildung 60: Auspressen der Zahnräder und Kugellager, zweiter Teil.

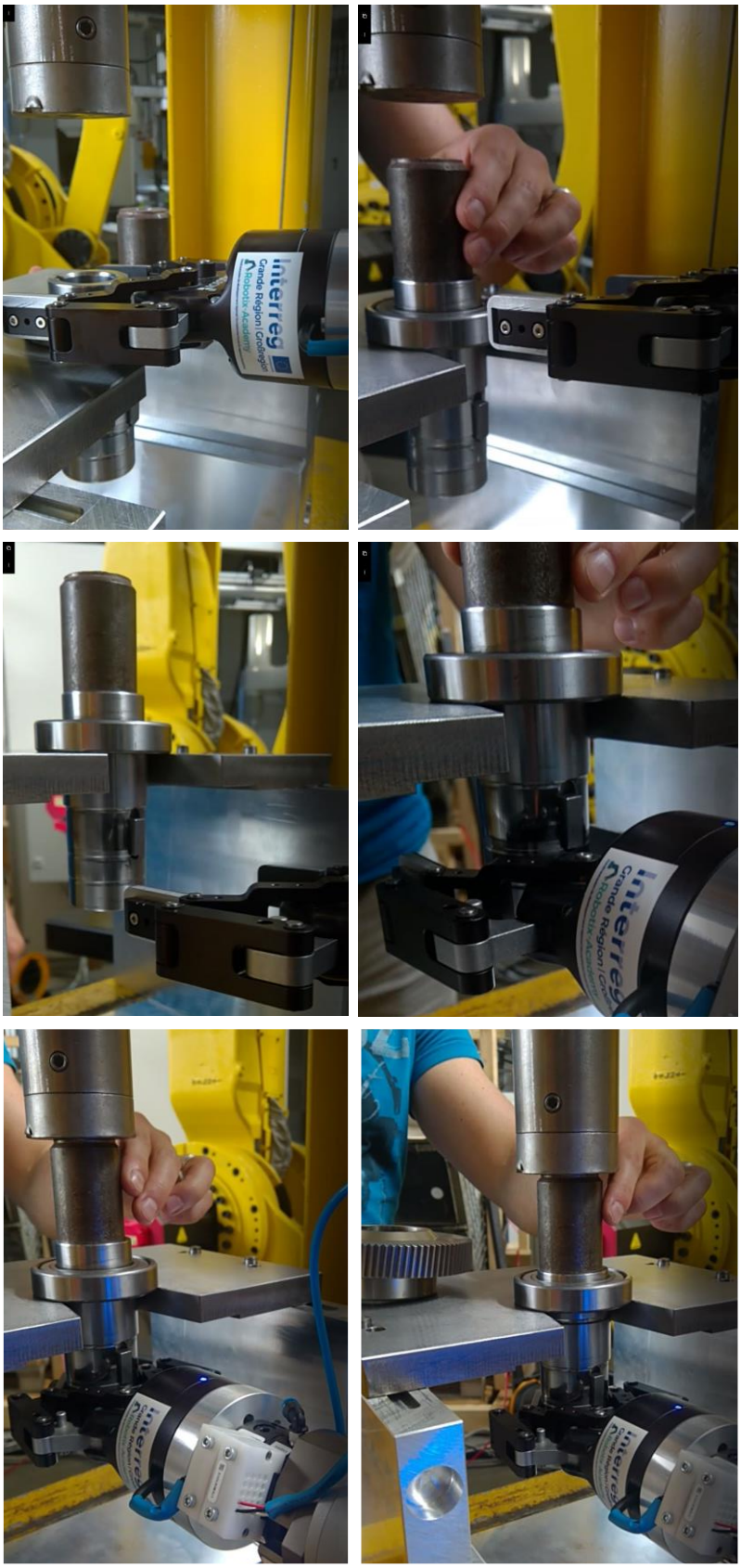

Abbildung 61: Auspressen der Zahnräder und Kugellager, dritter Teil.

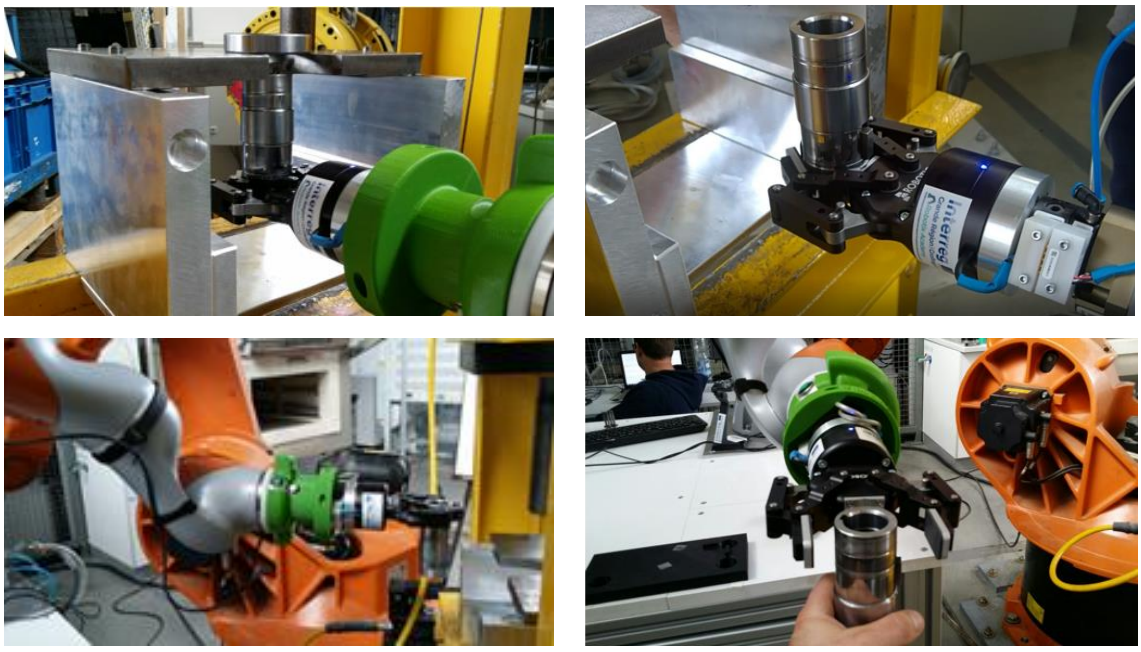

Abbildung 62. Auspressen der Zahnräder und Kugellager, vierter Teil.

Die Umsetzung dieses komplexen Demontageprozesses durch einzelne Aktionskommandos des IDAA konnte noch nicht erfolgen. Stattdessen wurde der gesamte Demontageprozess als Methode im Roboterprogramm hinterlegt, das durch ein einziges Aktionskommando ausgeführt wird. Die Steuerung des Greifers erfolgte hierbei dezentral aus dem Roboterprogramm. Hierzu wurden Aktionskommandos über die TCP/IP-Kommunikationsschnittstelle mit dem Kuka LBR iiwa GSA über das Ausführungsnetzwerk an den Greifer gesendet. Dies zeigt die Flexibilität des dezentralen Steuerungsansatzes und der RAS-Architektur. Die variable Greiflage der einzelnen Getriebestufen, die durch den variablen Durchmesser der gegriffenen Bauteile entsteht, und die Positionierung der Bauteile in verschiedenen Positionen in der Hydraulikpresse machte es zudem notwendig, einzelne Lagen einzulernen und im Roboterprogramm zu hinterlegen. Wobei die grundlegende Problematik darin besteht, dass zurzeit, nur einzelne Bauteile oder Unterbaugruppen (als Ganzes) im System modelliert wurden. In diesem Fall wäre es jedoch notwendig eine Unterbaugruppe und die einzelnen Bauteilen in ihr zu modellieren. Somit könnte im Prozessmodell auf die einzelnen Bauteile der Baugruppe und deren Merkmale zugegriffen werden, um zum Beispiel die Auspresslage eines Zahnrads zu erhalten. Momentan wird jedoch eine Unterbaugruppe als ein Bauteil im Prozessmodell dargestellt und somit ist der Zugriff auf einzelne Bauteilmerkmale nicht möglich. Dies stellt somit eine wichtige Erweiterung des Konzepts für die weitere Entwicklung des RAS dar.

8 Zusammenfassung und Ausblick

Gegenstand dieser Dissertation ist die Entwicklung eines intelligenten, robotergestützten Assistenzsystems, das den Menschen bei der zerstörungsfreien Demontage industrieller Produkte unterstützt. Die vom Assistenzsystem ausgehende Unterstützung sollte hierbei nicht punktuell in einzelnen Demontagetätigkeiten erfolgen, sondern über den gesamten Demontageprozess eines Produkts. Ein Merkmal des hierzu entwickelten Assistenzsystems ist seine offene, als Multiagentensystem umgesetzte Systemarchitektur, die die Integration, Koordinierung und Steuerung verschiedenster technischer Systeme auf Grundlage einer einheitlichen Kommunikationsinfrastruktur nach dem Ansatz des Internets der Dinge ermöglicht. Durch die Integration eines Demontageplanungssystems ist das Assistenzsystem in der Lage, die Demontage unterschiedlicher Produkte entsprechend einer variablen Zielstellung automatisiert zu planen und dem Nutzer die Wahl über die vom System ausgehende Unterstützung in den einzelnen Demontagehandlungen zu ermöglichen. Grundlegend hierfür sind die im Rahmen dieser Arbeit entwickelten Produktmodelle, die den Aufbau eines Produkts in einer maschinenverständlichen Form beschreiben. Durch ein zweites, im Assistenzsystem integriertes Planungssystem kann die Koordination, Steuerung und Überwachung des Demontageprozesses durch Planung der einzelnen Handlungen der beteiligten technischen Systeme sowie des Menschen ermöglicht werden. Grundlegend hierfür waren die im Rahmen dieser Arbeit entwickelten Prozessmodelle, die den verbindungspezifischen Demontageprozess eines Bauteils in einer allgemeinen Form sowie mögliche Formen der Arbeitsteilung und der technischen Unterstützung beschreiben. Auf Basis der Prozessmodelle kann das RAS anhand der vorliegenden Situation eine passende Folge von parametrierten Handlungsanweisungen generieren, diese anschließend durch ein Steuerungssystem koordiniert ausführen und überwachen. Mithilfe der Überwachung des Demontageprozesses können Abweichungen erkannt und die Handlungen durch eine Neuplanung angepasst werden. Auf Basis der entwickelten Systemarchitektur, der Beschreibung von geeigneten Modellen und der Integration von Planungs- und Steuerungssystemen, konnte eine höhere technische Autonomie im Assistenzsystem umgesetzt werden, die, in Kombination mit den entwickelten Mensch-Maschine-Kommunikationsschnittstellen, zu einer effizienteren Zusammenarbeit von Mensch und Maschine in Demontageprozessen führte. Die experimentelle Validierung zeigte jedoch auch weiteres Entwicklungspotenzial.

So können als Ausblick für weitere Forschungsarbeiten die Integration eines HTN-Planungssystems oder die Verwendung von graphenbasierten Suchalgorithmen genannt werden. Die Suche in Graphen bietet gegenüber der Suche in Baumstrukturen die Möglichkeit, dass auch mögliche parallel ablaufende Aktionen in der Aktionsplanung berücksichtigen werden. Auch eine dynamische Generierung der Teamzusammensetzung stellt ein herausforderndes wissenschaftliches Forschungsfeld dar. Weitere Forschungsarbeit sollte auch im Bereich der Bewegungsüberwachung des Nutzers erfolgen. Denn die Bewegungsüberwachung stellt die Grundlage für eine effizientere Kommunikation, Koordinierung und Intensionserkennung dar. Letztere wiederum bildet die Voraussetzung zur Entwicklung von proaktiven Assistenzsystemen. Insbesondere mit Hinblick auf die Sicherheit ist eine Überwachung der Bewegung des Menschen notwendig, wobei die Betrachtung der Sicherheit, in einem System, das automatisiert Bewegungshandlungen ermittelt und ausführt, hinsichtlich fehlender Normen und Gesetze eine komplexe Forschungsthematik darstellt. Auch die Anpassung des Assistenzverhaltens an einzelne Nutzer durch maschinelles Lernen könnte die Zusammenarbeit verbessern. Die hierzu notwendigen Daten werden bei der Interaktion des Menschen mit dem RAS erzeugt, liegen in strukturierter Form vor und können zum maschinellen Lernen, zum Beispiel auf Basis von Entscheidungsbäumen (engl. *decision trees*), verwendet werden. Darüber hinaus wäre die Implementierung von Sensoren zur 3D-Umwelterfassung zum Beispiel für die kollisionsfreie Bahnplanung sinnvoll. Denn viele Aktionen, die reine Umsetzbewegungen darstellen, würden dadurch in der Handlungsplanung wegfallen. Die Bahnplanung könnte hierzu in einem eigenständigen Agenten, der den Roboter direkt steuert, im RAS implementiert werden. Weitere Tätigkeitsfelder könnten der Weiterentwicklung der erstellten Softwarewerkzeuge, der Entwicklung von weiteren verbindungsspezifischen Demontageprozessen und der Verbesserung der Arbeitsplatzgestaltung dienen. Die Flexibilität des RAS könnte insbesondere durch die Integration des Kuka LBR iiwa auf einer mobilen Plattform verbessert werden. Denn der Arbeitsraum eines stationären Kuka LBR iiwa ist beschränkt und damit auch seine Einsatzmöglichkeiten.

Abschließend soll angemerkt werden, dass diese Arbeit nur einen kleinen Beitrag zur Vision von generisch einsetzbaren Assistenzsystemen zur Demontage von individuellen Produkten im Bereich der Wartung, Instandhaltung und Refabrikation leisten konnte. Durch die Entwicklung einer geeigneten Systemarchitektur, die ganzheitliche Betrach-

tung des Systems hinsichtlich der Informationsverarbeitung und der in dieser Arbeit gesammelten experimentellen Erkenntnisse konnte jedoch eine solide Basis für weitere Forschung in diesem interdisziplinären Forschungsfeld geschaffen werden.

9 Anhang

Anhang A: Produktmodell im XML Format

In Abbildung 64 ist das Produktmodell (im XML Format) der Beispielbaugruppe von Abbildung 63 dargestellt. Jedes Bauteil der Baugruppe ist als eine Instanz der Basis Bauteilklasse (*Part*) beschrieben. Weiterhin bestehen nur Aufgelegt-Verbinden (*LyingOn*) zwischen den Bauteilen. Die Bauteilmerkmale des gelb markierten Bauteils (in Abbildung 63) sind im Produktmodell geöffnet. Auch die Aufgelegt-Verbindung, die das gelb markierte Bauteil mit dem darunterliegenden Bauteil mit ID 1 herstellt, ist darin zu sehen. In Abbildung 63 sind die Koordinatensysteme der Greiflage der Bauteile sowie das farblich dargestellte BGKS zu erkennen. Die drei Bohrungen in der Grundplatte werden zum direkten Einmessen des BGKS verwendet.

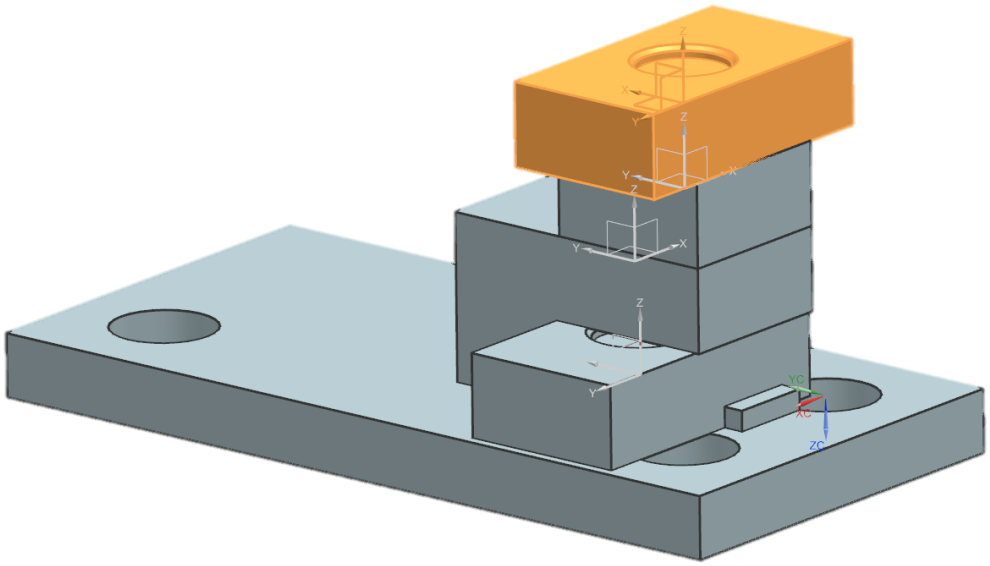

Abbildung 63: Beispielbaugruppe.

```
<?xml version="1.0"?>
<Productmodel xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  <PartList>
    <Part>
      <ID>0</ID>
      <PartName>Bauteil_A 1</PartName>
      <PartFrame>
        <double>1.1657341758564144E-15</double>
        <double>1</double>
        <double>-4.4328702259040069E-16</double>
        <double>22.500000000000014</double>
        <double>0.99999999999999989</double>
        <double>-1.1102230246251565E-15</double>
        <double>2.8034237584964062E-16</double>
        <double>21.50000000000005</double>
        <double>2.8034237584964013E-16</double>
        <double>-4.43287022590401E-16</double>
        <double>-0.99999999999999989</double>
        <double>-69.999999999999972</double>
      </PartFrame>
      <Material>PLA</Material>
      <Mass>0</Mass>
      <Picture>Bauteil_A 1.JPG</Picture>
      <Width>40</Width>
      <Height>30</Height>
      <Length>70</Length>
      <RobotTool>Grp</RobotTool>
      <UserTool />
    </Part>
    <Part>
    <Part>
    <Part>
    <Part>
  </PartList>
  <LinkList>
    <Link xsi:type="LyingOn">
      <ID>0</ID>
      <FromPartID>
        <int>0</int>
      </FromPartID>
      <ToPartID>
        <int>1</int>
      </ToPartID>
    </Link>
    <Link xsi:type="LyingOn">
    <Link xsi:type="LyingOn">
    <Link xsi:type="LyingOn">
  </LinkList>
</Productmodel>
```

Abbildung 64: Produktmodell der Beispielbaugruppe von Abbildung 63.

Anhang B: Prozessmodell in XML-Format

In Abbildung 65 (links) ist das XML-Importformat, mit den darin definierten Prozessmodellen (*ProcessModellSet*) sowie der Objekt- (*ObjectTemplateSet*) und Agentenvorlagen (*AgentTemplateSet*), dargestellt. In der Abbildung 65 (rechts) zu sehen ist ein Prozessmodell (*ProcessModel*) im XLM-Format. An den Zeilennummern ist zu erkennen welchen Umfang diese Dateien besitzen und dass sie nur mithilfe des Prozessmodelleditors und einer grafischen Benutzeroberfläche effizient erstellt werden können. Darin zu erkenn ist der Aufbau eines Prozessmodells, das sich aus einer Menge von Objekten (*ObjectSet*) und Agenten (*AgentSet*) sowie unterschiedlichen Formen von Arbeitsteilungen bzw. technischer Assistenzfunktionen (*BehaviorSet*) zusammensetzt.

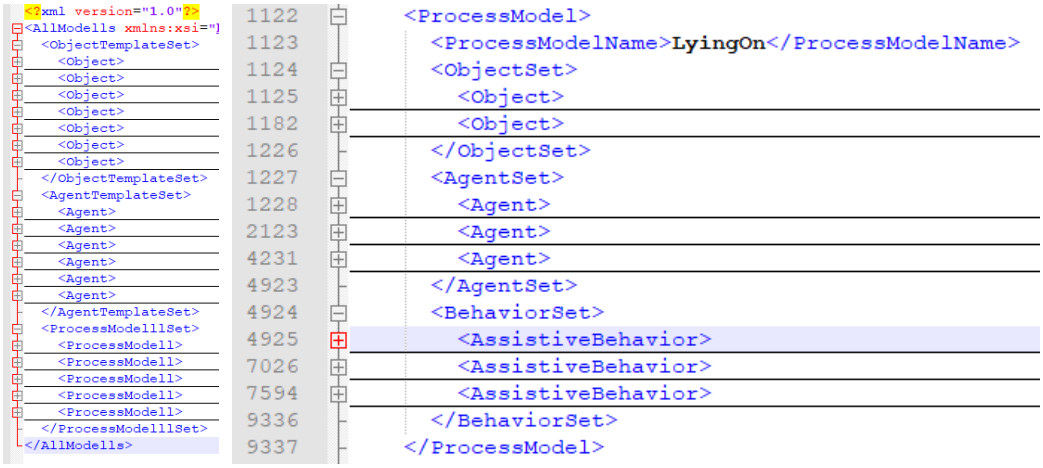

Abbildung 65: XML-Format der Prozessmodelle.

Der Aufbau der Parameter- (*ParameterSet*) und Zustandsmenge (*StateSet*) eines Objektes (*Object*) bzw. eines Agenten (*Agent*) sind in Abbildung 66 und Abbildung 67 dargestellt. In Abbildung 66 sind in der Parametermenge nur dynamische Parameter (*Parameter*) definiert. Zum Beispiel bezieht sich der Parameter (*PartPos*) auf die Basis Bauteilklasse (*Part*) und auf eine Funktion (*KukaPartFrame*) die die Greiflage des Bauteils (*PartFrame*) aus dem Produktmodell (vgl. Abbildung 64) im Kuka Format X,Y,Z,A,B,C übergibt. In Abbildung 67 ist dargestellt, wie sich die Zustandsmenge aufbaut, eine Zustandsvariable ist geöffnet und zeigt deren Namen (*Effector*), deren Domäne (*StateVariableDomain*), den Kommunikationskanal (*MqttStatusTopic*) sowie initialen (*InitialStateValue*) und Zielzustand (*TargetStateValue*).

```
<Object>
  <ObjectName>Part</ObjectName>
  <ParameterSet>
    <ObjectName>Part</ObjectName>
    <ParameterSet>
      <Parameter>
        <ObjectName>Part</ObjectName>
        <ParameterName>PartPos</ParameterName>
        <LinkedObjectClassProperty>KukaPartFrame</LinkedObjectClassProperty>
        <LinkedObjectClass>Part</LinkedObjectClass>
      </Parameter>
      <Parameter>
        <ObjectName>Part</ObjectName>
        <LinkedObjectClassProperty>Heigth</LinkedObjectClassProperty>
        <LinkedObjectClass>Part</LinkedObjectClass>
      </Parameter>
      <Parameter>
        <ObjectName>Part</ObjectName>
        <ParameterName>PartRobotTool</ParameterName>
        <LinkedObjectClassProperty>RobotTool</LinkedObjectClassProperty>
        <LinkedObjectClass>Part</LinkedObjectClass>
      </Parameter>
    </ParameterSet>
  </ParameterSet>
```

Abbildung 66: Aufbau der Parametermenge im XML-Format.

```
<Agent>
  <AgentName>Kuka_iiwa</AgentName>
  <ParameterSet>
  <StateSet>
    <ObjectName>Kuka_iiwa</ObjectName>
    <StateSet>
      <State>
        <AgentName>Kuka_iiwa</AgentName>
        <MqttStatusTopic>iiwaTopic/Status</MqttStatusTopic>
        <StateVariableName>Effector</StateVariableName>
        <StateVariableDomain>
          <string>Lock</string>
          <string>Unlock</string>
          <string>Ball</string>
          <string>GearSetGripper</string>
          <string>Grp</string>
          <string>Pen</string>
          <string>Ratchet</string>
          <string>ScrewDriver</string>
          <string>Unknown</string>
        </StateVariableDomain>
        <InitialStateValue>Unknown</InitialStateValue>
        <TargetStateValue />
      </State>
      <State>
      <State>
    </StateSet>
  </StateSet>
  <ActionSet>
</Agent>
```

Abbildung 67: Aufbau der Zustandsmenge im XML-Format.

Der Aufbau eines Agentenmodells im XML-Format ist in Abbildung 68 dargestellt. Darin ist die Aktionsmenge (*ActionSet*) geöffnet und zeigt die einzelnen Aktionen (*Action*).

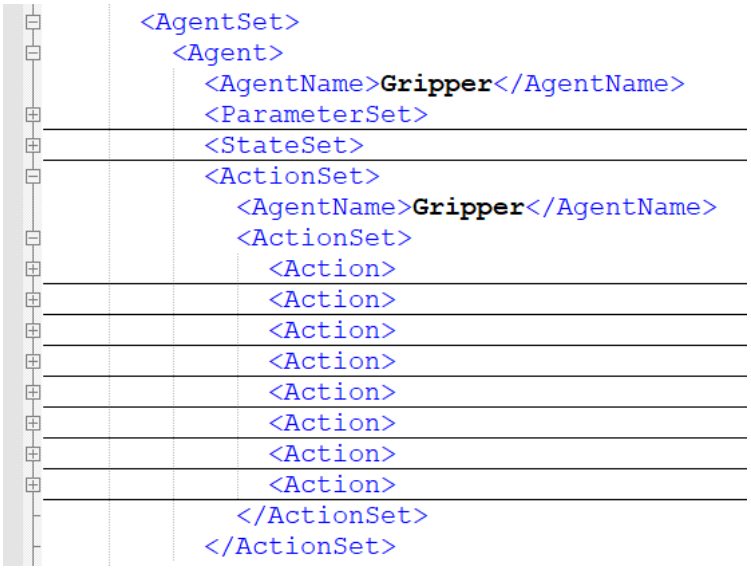

Abbildung 68: Aufbau eines Agentenmodells im XML-Format.

In Abbildung 69 ist weiterhin der Aufbau einer Aktion mit dem Aktionsnamen (*ActionName*) ‚*Open*‘ eines Agenten (*Gripper*) (des Robotiq Greifers) im XML-Formt dargestellt. Der Aktion sind Aktionskosten (*ActionCosts*) sowie der Kommunikationskanal (*MqttCommandTopic*) zugewiesen und sie besitzt keine Parameter (*Parameter*). Als Vorbedingungen (*PreConditionSet*) der Aktion sind zwei Zustandsvariablen (*StateVariable*) definiert, der Greifer muss betriebsbereit und die Finger geschlossen sein. Der Transitionseffekt (*TransitionSet*) beschreibt den Umstand das die Finger sich bei der Aktionsausführung bewegen und der Endeffekt (*EffectSet*), dass die Finger nach Aktionsausführung geschlossen sind.

Der Aufbau einer im Prozessmodell definierten Assistenzfunktionen (*AssistiveBehavior*) ist in Abbildung 70 dargestellt. Sie definiert den Namen (*BehaviorName*) und eine Aktionsmenge (*ActionSet*) der darin vorgesehenen Aktionen.

```
<Agent>
  <AgentName>Gripper</AgentName>
  <ParameterSet>
  <StateSet>
  <ActionSet>
    <AgentName>Gripper</AgentName>
    <ActionSet>
      <Action>
        <AgentName>Gripper</AgentName>
        <ActionName>Open</ActionName>
        <ActionCosts>5</ActionCosts>
        <MqttCommandTopic>GrprTopic</MqttCommandTopic>
        <Parameter />
        <Command>Open</Command>
        <PreConditionSet>
          <StateVariable>
          <StateVariable>
        </PreConditionSet>
        <TransitionSet>
          <StateVariable>
        </TransitionSet>
        <EffectSet>
          <StateVariable>
        </EffectSet>
      </Action>
      <Action>
```

Abbildung 69: Aufbau von Aktionen im XML-Format.

```
<BehaviorSet>
  <AssistiveBehavior>
    <BehaviorName>automatic</BehaviorName>
    <ActionSet>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
      <Action>
    </ActionSet>
  </AssistiveBehavior>
```

Abbildung 70: Aufbau einer Assistenzfunktion im Prozessmodell.

Anhang C: Bedienung der Benutzeroberfläche des intelligenten Demontage Assistenten Agenten

Im derzeitigen Entwicklungsstand erfolgt der Import des Produktmodells sowie der Prozessmodelle durch den Nutzer über das Menü ‚*File->Load->Product Modell*' bzw. ;*File->Load->Process Models*' (vgl. Abbildung 71) über ein sich öffnendes Datei-Dialog Fenster.

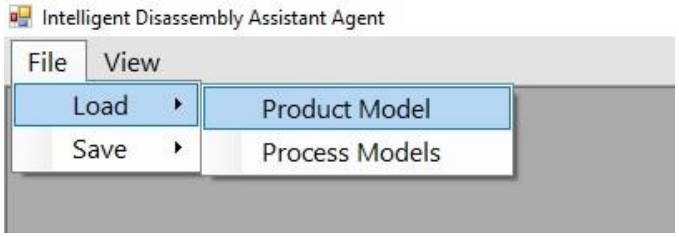

Abbildung 71: Import des Produktmodells und der Prozessmodelle.

Durch den Import des Produktmodells öffnet sich das Ansichtsfenster ‚*Product Modell*' dargestellt in Abbildung 72.

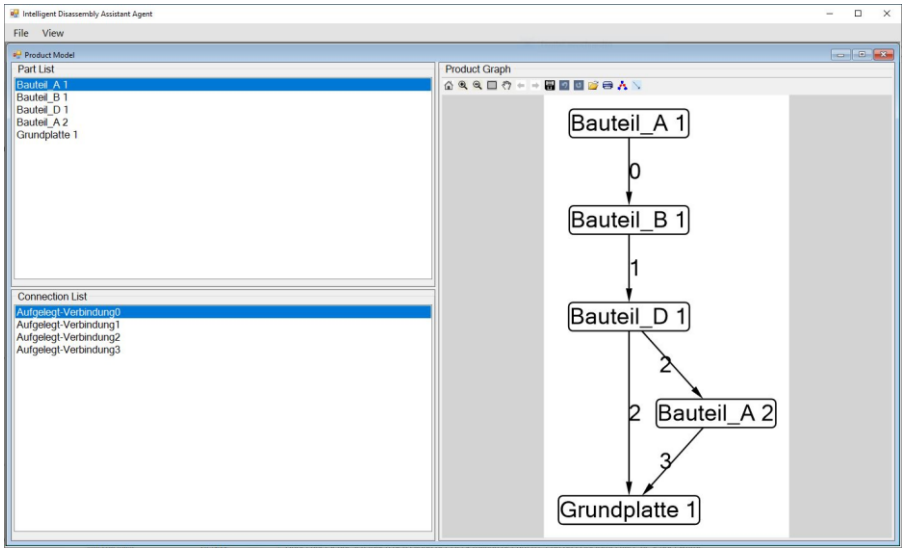

Abbildung 72: Produktmodell Ansichtsfenster.

Darin werden die enthaltenen Bauteile und Verbindungen aufgelistet sowie der Produktgraph abgebildet. Für die Visualisierung des Produktgraphen sowie der Suchbäume wurde die freie Bibliothek *Microsoft Automatic Graph Layout* (MSAGL) [176] verwendet.

Durch den Import der Prozessmodelle wird das Ansichtsfenster des Prozessmodelleditors in IDAA geöffnet (vgl. Abbildung 73).

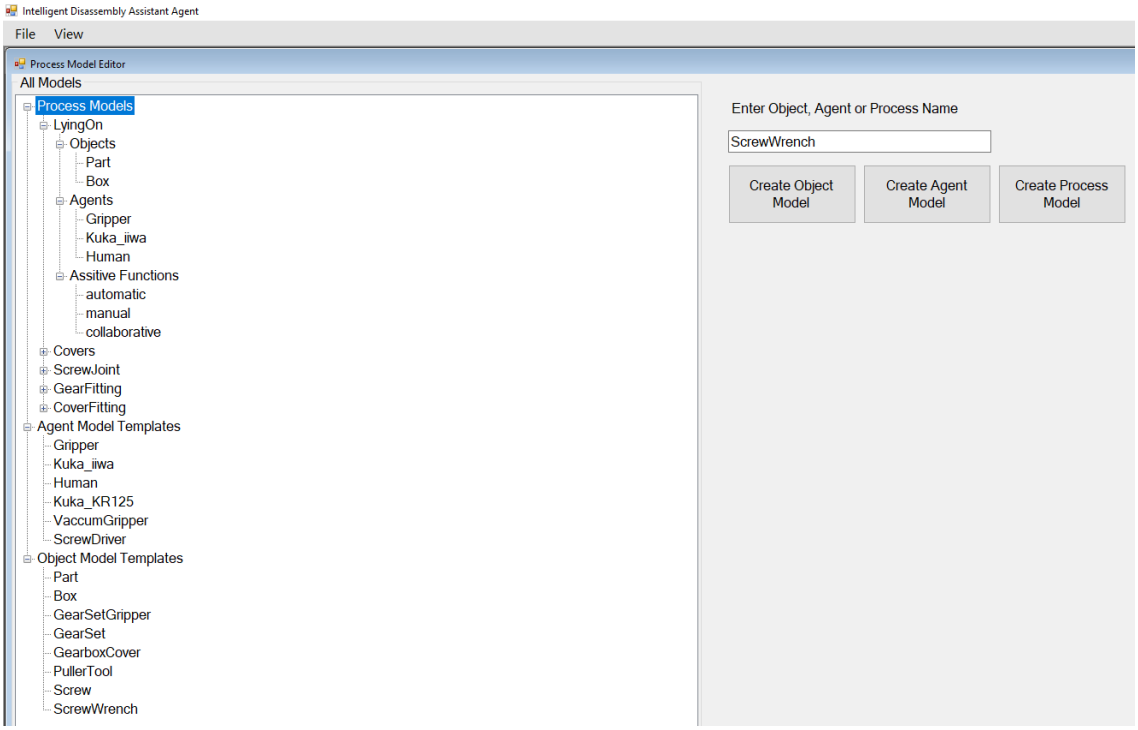

Abbildung 73: Prozessmodelleditor Ansichtsfenster.

Auf der linken Seite der Abbildung werden alle in der XML-Datei definierten Prozessmodelle sowie Agenten- und Objektmodelvorlagen dargestellt. Das in Kapitel 6.3.2.1 beschrieben Prozessmodell für Aufgelegt-Verbindungen (*LyingOn*) ist geöffnet und zeigt die darin definierten Objekte (*Part*, *Box*) und Agenten (*Gripper*, *Kuka_iiwa*, *Human*) sowie die definierten Unterstützungsfunktionen (*automatic*, *manual*, *collaborative*). Weiterhin sind die Prozessmodelle zum Lösen von Schraubenverbindungen (*ScrewJoint*), zum Abziehen des Getriebedeckels (*CoverFitting*) und zum Entnehmen des Getriebesatzes (*GearFitting*) darin enthalten. In den Agentenmodellvorlagen hinterlegt, sind die im RAS integrierten steuerbaren technischen Systeme. Dazu gehören der Robotiq zwei Finger Greifer (*Gripper*), der Kuka LBR iiwa (Kuka_iiwa), der Kuka KR 125 (*Kuka_KR125*), der Sauggreifer (*VaccumGripper*) und das Schraubwerkzeug (*ScrewDriver*). In den Objektmodellvorlagen sind weiterhin Modelle für Bauteile, nicht steuerbare Robotereffektoren und Handwerkzeuge hinterlegt. Als Bauteile sind hierin einfache Bauteile (*Part*), Schrauben (*Screw*), der Getriebesatz (*GearSet*), der Getriebedeckel (*GearCover*) definiert. Weiterhin enthalten sind Objektmodelle für den Robotereffektor zur

Entnahme des Getriebesatzes (*GearSetGripper*), das Werkzeug zum Abziehen des Getriebedeckels (*PullerTool*) sowie Schraubenschlüssel (*ScrewWrench*). Auf der linken Seite der Abbildung 73 sind Funktionen zum Erstellen von neuen Objekt-, Agenten- und Prozessmodellen im Prozessmodell abgebildet.

Nach dem Laden der Prozessmodelle kann über den Menüpunkt ‚*View->Disassembly Task Planning*‘ zur Demontageplanung übergegangen werden. Es öffnet sich das Ansichtsfenster der Demontageplanung (vgl. Abbildung 74).

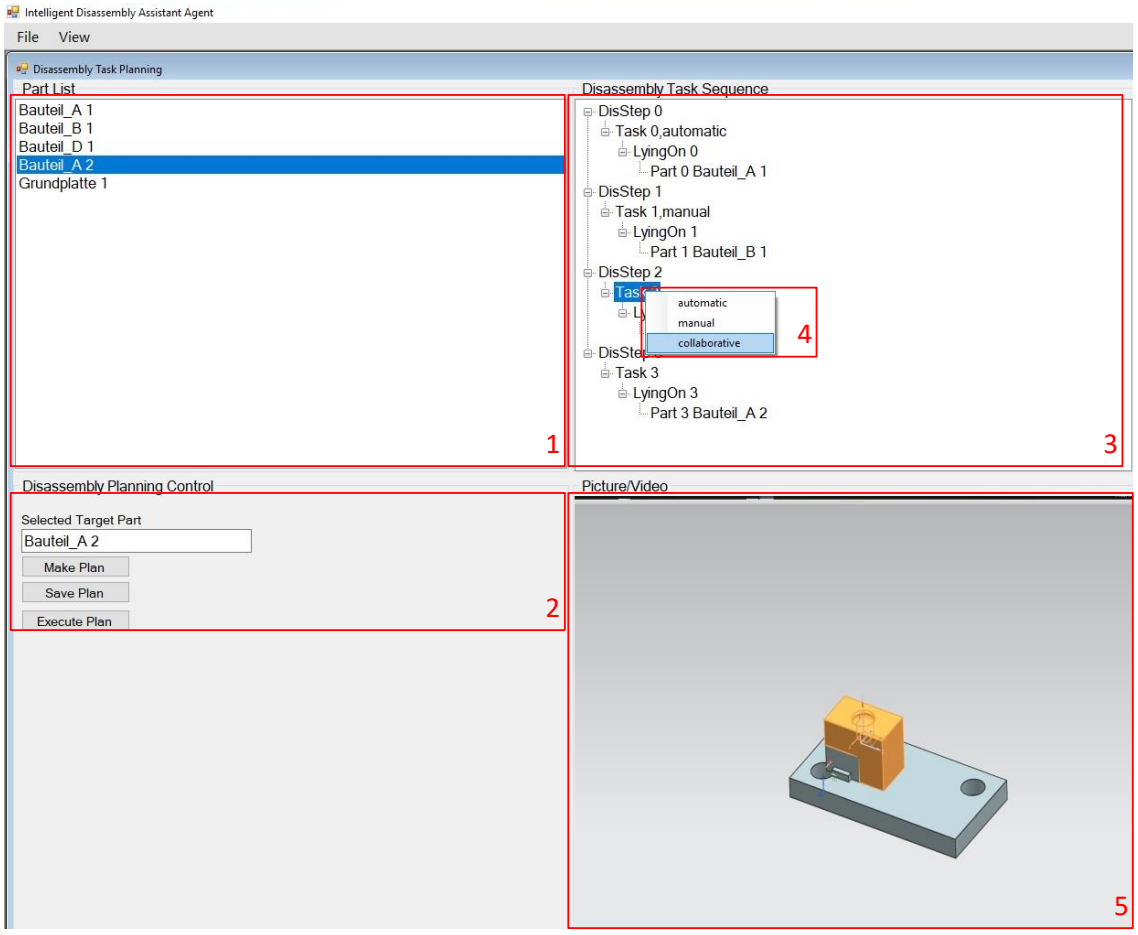

Abbildung 74: Demontageplanung Ansichtsfenster.

Für die Demontageplanung muss der Nutzer ein Zielbauteil aus der Liste (1) auswählen und die Demontageplanung, durch Betätigung des ‚*Make Plan*‘ Steuerelements in (2), ausführen. Der dadurch erzeugte Demontageplan (*Disassembly Task Sequence*) mit einzelnen Demontageschritten (*DisStep*) und Demontageaufgabe (*Task*) wird im Fenster (3)

dargestellt und kann vom Nutzer nachfolgend angepasst werden. Der Nutzer kann hierzu einzelne Demontageschritte oder Demontageaufgabe entfernen oder die Reihenfolge der Demontageaufgaben anpassen. Auch die Form der Unterstützung in jeder Demontageaufgabe kann definiert werden, hierzu selektiert der Nutzer zuerst die entsprechende Demontageaufgabe, durch Betätigung der linken Maustaste erscheint dann ein Dialogfenster (4) zur Auswahl der gewünschten Assistenzfunktion. Die möglichen Assistenzfunktionen werden dabei aus dem der Demontageaufgabe hinterlegten Prozessmodell geladen. Bei der Selektion eines Bauteils durch den Nutzer im Demontageablauf wird auch ein Bild des zu demontierenden Bauteils im Fenster (5) dargestellt. Änderungen am Demontageplan können durch das Steuerelement ‚*Save Plan*' gespeichert und der Demontageprozess durch bestätigen des Steuerelements ‚*Execute Plan*' ausgeführt werden.

Nach dem Start des Demontageprozesses sind keine weiteren Interaktionen des Nutzers mit der grafischen Oberfläche des IDAA notwendig, die einzelnen Demontageaufgaben werden nacheinander geplant, ausgeführt und überwacht. Hierzu werden, für jede Demontageaufgabe, die Softwaremodule: ‚Initialisierung der Detailplanung', ‚Prozessüberwachung' sowie ‚Aktionsplanung' und ‚Steuerung des Demontageprozesses' (vgl. Abbildung 26) (in dieser Reihenfolge) aufgerufen. Diese einzelnen Softwaremodule können aber auch manuell zu Testzwecken aufgerufen werden. Zum besseren Verständnis und zur Dokumentation soll der manuelle Prozess hierzu beschrieben werden.

Das Ansichtsfenster der ‚Initialisierung der Detailplanung' kann über das Menü ‚*View-> Task Initialisation'* aufgerufen werden (vgl. Abbildung 75).

Durch Betätigung des Steuerelement ‚*Initialize Task*' im Kontrollfeld (1) kann die momentane Demontageaufgabe initialisiert werden. Die in einer Demontageaufgabe, bzw. im hinterlegten Prozessmodell, definierten Agenten und Objektmodelle werden geladen und in den Listen (6) aufgeführt. Weiterhin wird der initiale und Zielprozesszustand aus dem Prozessmodell, bzw. den darin definierten Objekt- und Agentenmodelle, gebildet und in den Listenfeldern (3) und (4) dargestellt. Die Darstellung der Zustandsvariablen, z. B. *Effector(Kuka_iiwa)=Unknown* erfolgt hierbei in der Form *Kuka_iiwa,Effector,Unknown*. Die Aktionen der beteiligten Agenten werden initialisiert, Aktionskosten zugewiesen und zusammengefasst im Listenfeld (5) dargestellt. Im Listenfeld (2) werden außerdem die Assistenzfunktionen des Prozessmodells aufgelistet und die vom Nutzer gewählte Assistenzfunktion blau selektiert. Mittels der Steuerelemente ‚*Recent Task*' und

‚*Next Task*‘ kann zur Initialisierung der vorherigen bzw. nächsten Demontageaufgabe gesprungen werden.

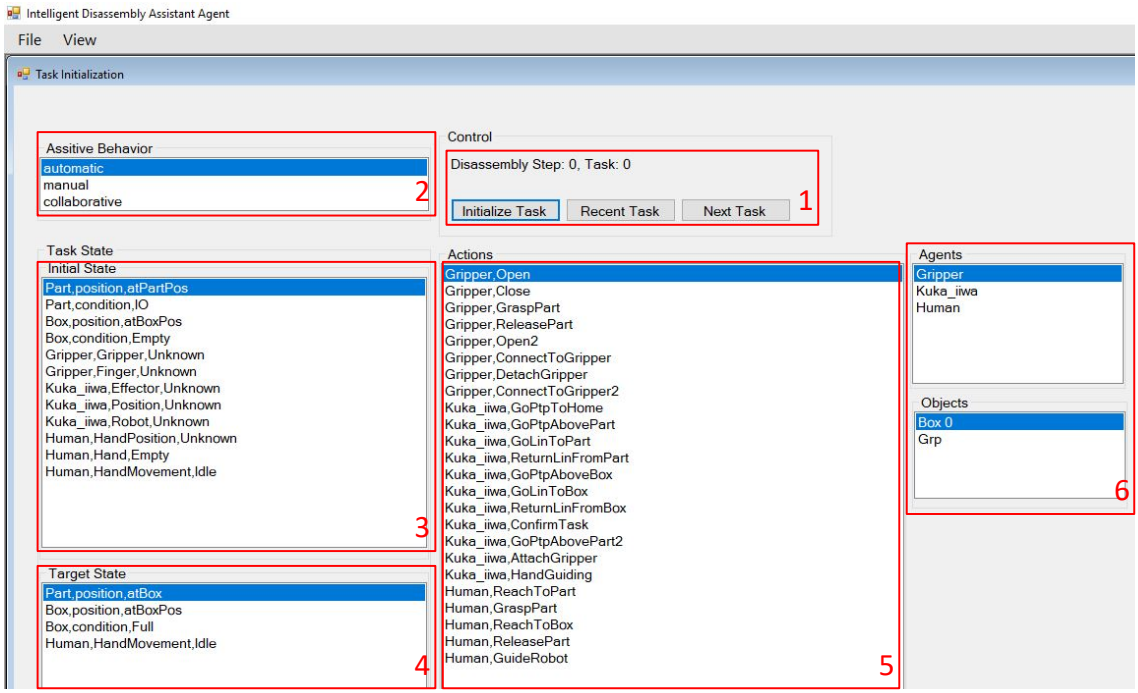

Abbildung 75: Ansichtsfenster der Initialisierung der Detailplanung.

Nach der Initialisierung der Demontageaufgabe muss das Modul zur Prozessüberwachung gestartet werden. Hierzu kann über das Menü ‚*View->Process Monitoring*‘ das Ansichtsfenster der Prozessüberwachung aufgerufen (vgl. Abbildung 76) werden.

Im Kontrollfeld (1) kann über das Steuerelement ‚*Start Process Monitoring*‘ bzw. ‚*Task Monitor Running*‘. die Prozessüberwachung gestartet bzw. beendet werden. Dadurch wird eine Hintergrundprozess gestartet der den Zustandsvariablen zugeordneten Kommunikationskanäle abonniert. Der Empfang von Gerätezustandsänderungen über das Wahrnehmungsnetzwerk wir im Listenfeld (3) dokumentiert und der initiale Prozesszustand entsprechend aktualisiert (vgl. Abbildung 76 Felder (2) und (3) mit Abbildung 75 Feld (3)).

Nach der Aktualisierung des initialen Prozesszustands kann nun das Ansichtsfenster der Aktionsplanung über das Menü ‚*View->Action Planning*‘ aufgerufen werden (vgl. Abbildung 77).

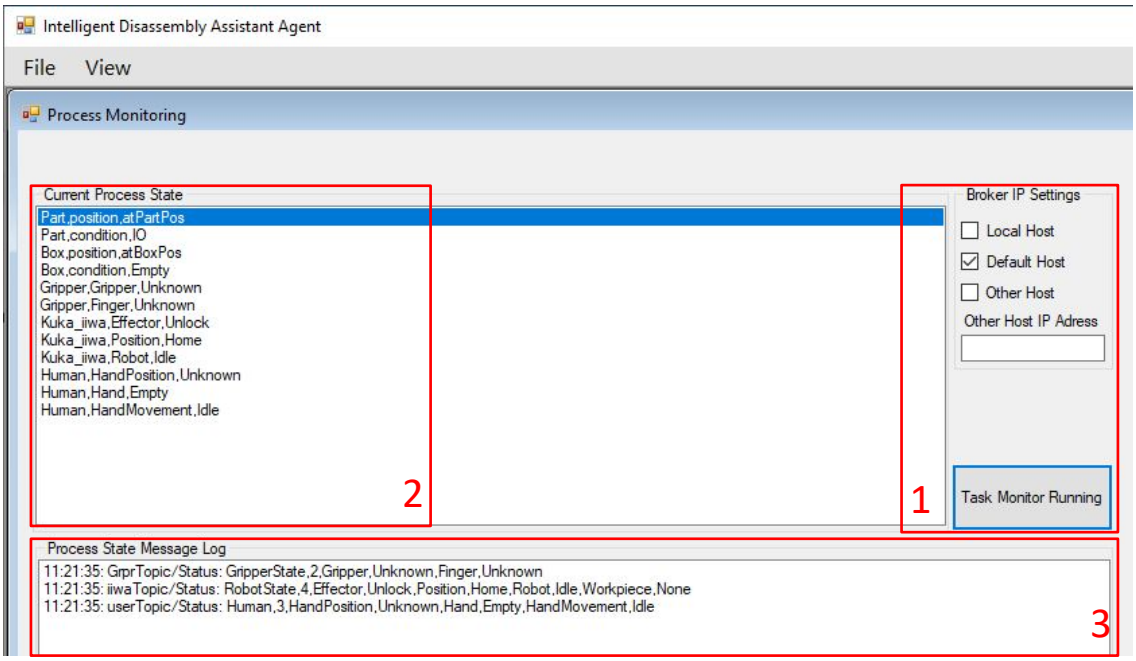

Abbildung 76: Ansichtsfenster der Prozessüberwachung.

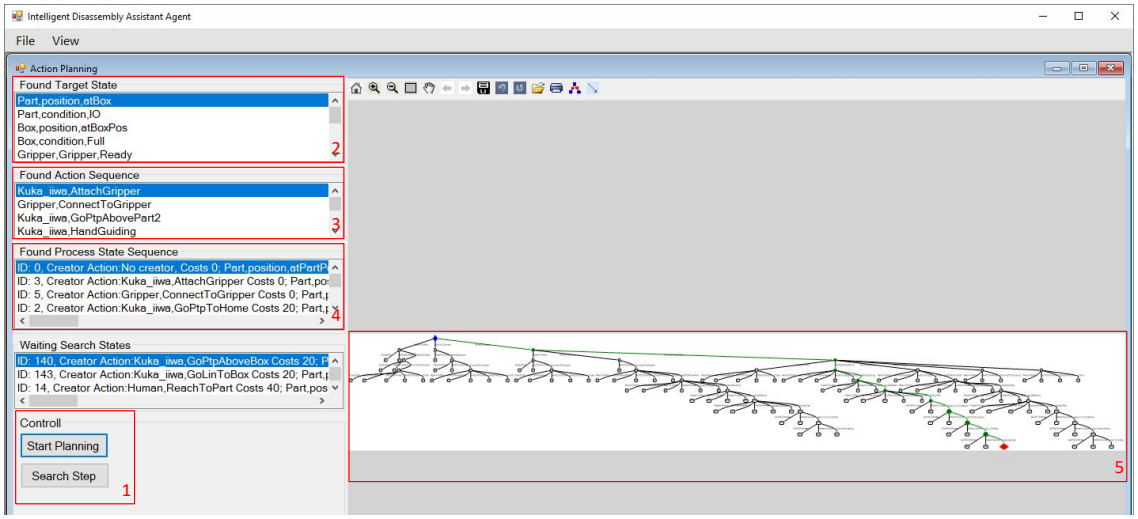

Abbildung 77: Ansichtsfenster der Aktionsplanung.

Mit dem Steuerelement ‚*Start Planning*‘ im Kontrollfeld (1) wird die Handlungsplanung ausgelöst. Das Ergebnis der Handlungsplanung ist eine Sequenz von Aktionen (aufgelistet in Feld 3) und Prozesszuständen (aufgelistet in Feld 4) die zum Zielzustand (dargestellt in Feld 1) führt. Der bei der Handlungsplanung generierte Suchbaum wird in Feld (5) dargestellt. Über das Steuerelement ‚*Search Step*‘ im Kontrollfeld (1) kann weiterhin die

Expansion des Suchbaums schrittweise erfolgen, diese Funktion ist hilfreich beim Design neuer sowie bei der Erweiterung bestehender Prozessmodelle. Ein einfacher Suchbaum ist in Abbildung 78 dargestellt und soll nun kurz erklärt werden. Der aktualisierte initiale Prozesszustand als Ausgangspunkt der Suche wird durch einen blauen Diamanten dargestellt. Von diesem gehen Kanten, welche Aktionen darstellen, aus, die neue Prozesszustände generieren. Der Zielzustand wird durch einen roten Diamanten dargestellt. Prozesszustände (Knoten) und Aktionen (Kanten) zwischen Start- und Zielzustand stellen die Sequenz der zu erwartenden Prozesszustände bzw. der Aktionen dar und sind grün dargestellt.

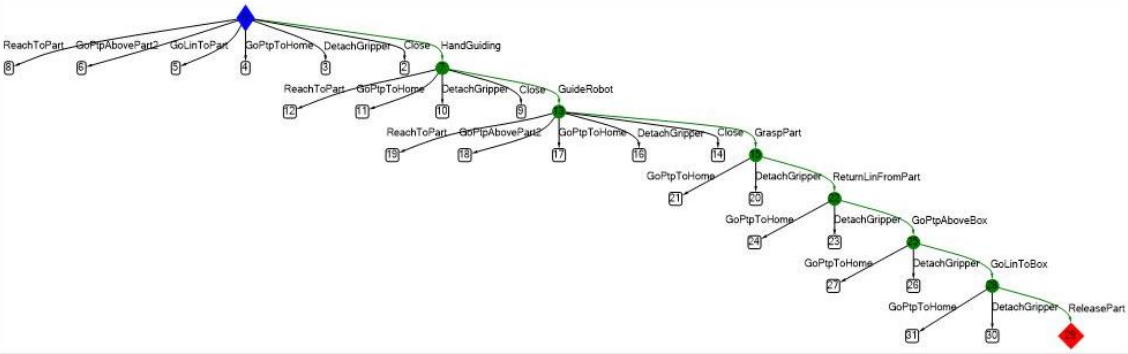

Abbildung 78: Suchbaum aus der Aktionsplanung.

Nach der Handlungsplanung kann über das Menü ‚*View->Action Execution and Synchronization*' zum Ansichtsfenster der Prozesssteuerung (vgl. Abbildung 79) gewechselt werden.

Im Ansichtsfenster der Prozesssteuerung wird die Aktionssequenz in Feld 2 sowie die zu erwartende Prozesszustände im Feld 3 dargestellt. Bei der Aktionsausführung wird das jeweils letzte versendete Aktionskommando sowie der nächste zu erwartende Prozesszustand werden in den Listen blau hervorgehoben. Über das Steuerelement ‚*Create MQTT Command Client*' bzw. ‚*MQTT Comannd Client running*' im Feld 1 kann eine eigene Kommunikationsschnittstelle zum Ausführungsnetzwerk des RAS gestartet bzw. beendet werden. Weiterhin kann über das Steuerelement ‚*Execute Aktion*' bzw. ‚*Execute Action Sequence*' die Ausführung einer oder aller Aktionen in der Aktionssequenz gestartet werden. Die versendeten Aktionskommandos werden dabei im Feld 6 protokolliert. Im Feld 4 wird der aktuelle Prozesszustand durch das Softwaremodule der Prozessüberwachung ständig aktualisiert und dargestellt. Im Feld 5 ist der nächste erwartete Prozesszustand

dargestellt. Bei jeder Aktualisierung des Prozesszustands wird überprüft ob der erwartete nächste Prozesszustand eingetreten ist und ob das nächste Aktionskommando versendet werden kann.

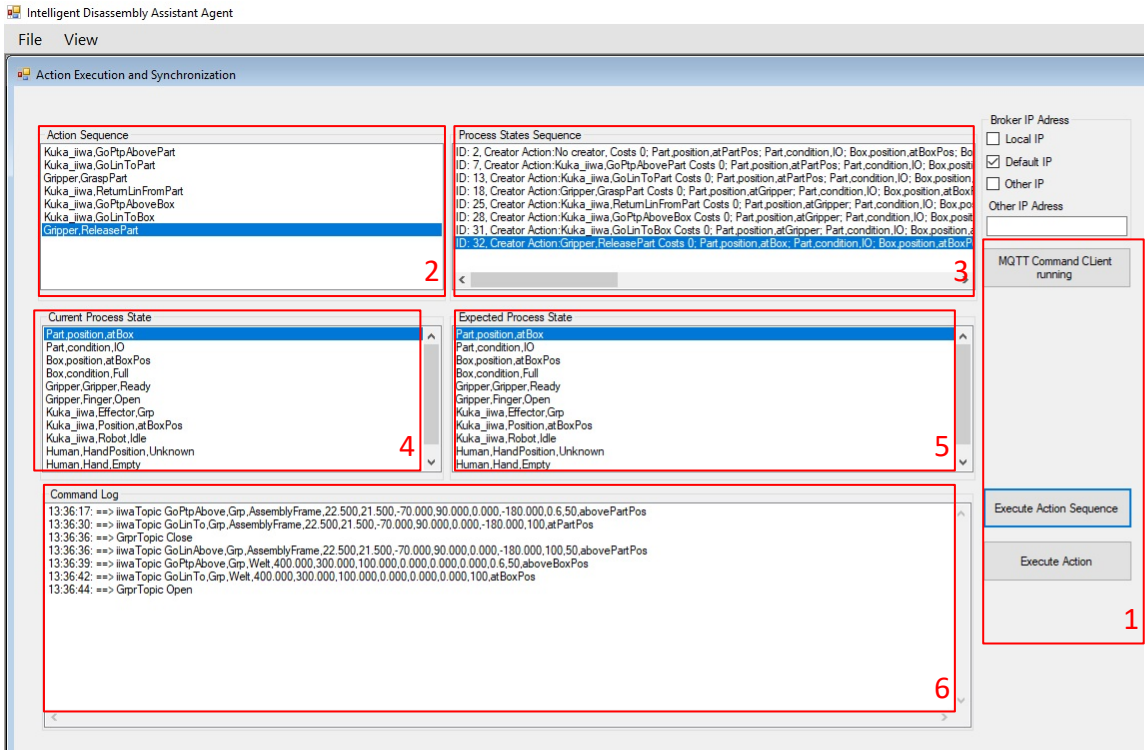

Abbildung 79: Ansichtsfenster der Prozesssteuerung.

Anhang D: Bedienung der Benutzeroberfläche der Geräte steuernden Agenten

Kuka LBR iiwa GSA

Die Benutzeroberfläche des Kuka LBR iiwa GSA ist in die drei folgenden Registerkarten aufgeteilt:

- ‚*MQTT Communication*' (vgl. Abbildung 80)
- ‚*TCP/IP Communication*' (vgl. Abbildung 81)
- ‚*Action Commands Test*' (vgl. Abbildung 82 und Abbildung 83)

In der Registerkarte ‚*MQTT Communication*' kann über das Steuerelement ‚*Start Client*' bzw. ‚*Close Client*' im Feld 1 die Kommunikationsschnittstelle zum RAS aufgebaut bzw. beendet werden. Dadurch werden die Kommunikationskanäle des Ausführungsnetzwerks abonniert sowie der aktuelle Gerätezustand auf dem Kommunikationskanal des Wahrnehmungsnetzwerkes veröffentlicht. Eingehende Aktionskommandos werden nachfolgend in der Listefeld 2 und ausgehende Zustandsänderungen im Listenfeld 3 dokumentiert.

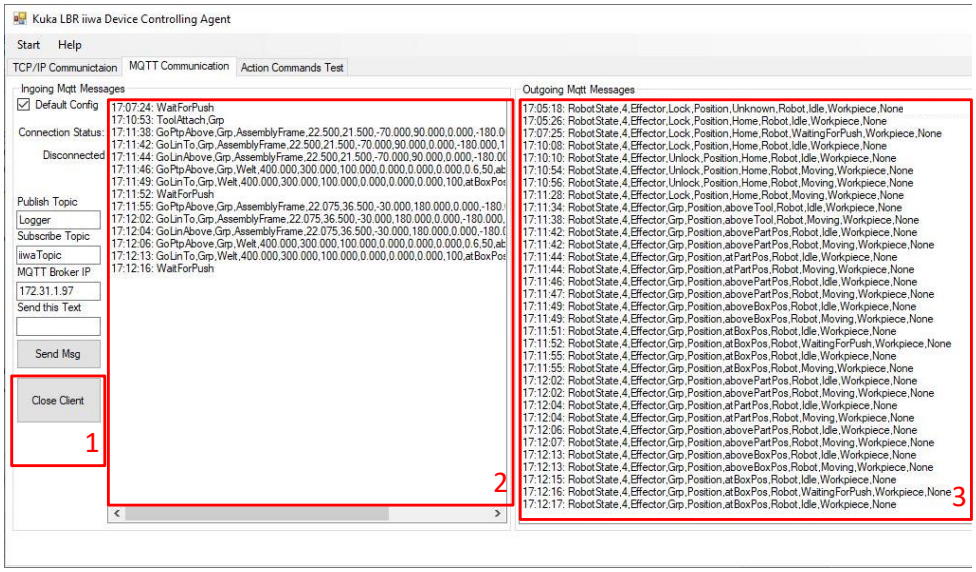

Abbildung 80: Registerkarte MQTT Communication.

Die Kommunikationsschnittstelle mit dem RAS muss zuerst hergestellt werden, nachfolgend kann in der Registerkarte ‚*TCP/IP Communication*‘ über die Steuerelemente ‚*Start Server*‘ bzw. ‚*Close Server*‘ (im Feld 1 und 2) die Kommunikationsschnittstelle mit dem LBR iiwa Roboterprogramm gestartet bzw. beendet werden. Um die Verbindung aufzubauen muss das Roboterprogramm gestartet werden, nach Aufbau der Kommunikation fährt der LBR iiwa in seine Grundstellung und sendet seinen Gerätezustand an den GSA. Die erhaltenen Gerätezustandsänderungen des LBR iiwa werden im Listenfeld 3 und die gesendeten Aktionskommandos im Listenfeld 4 dokumentiert.

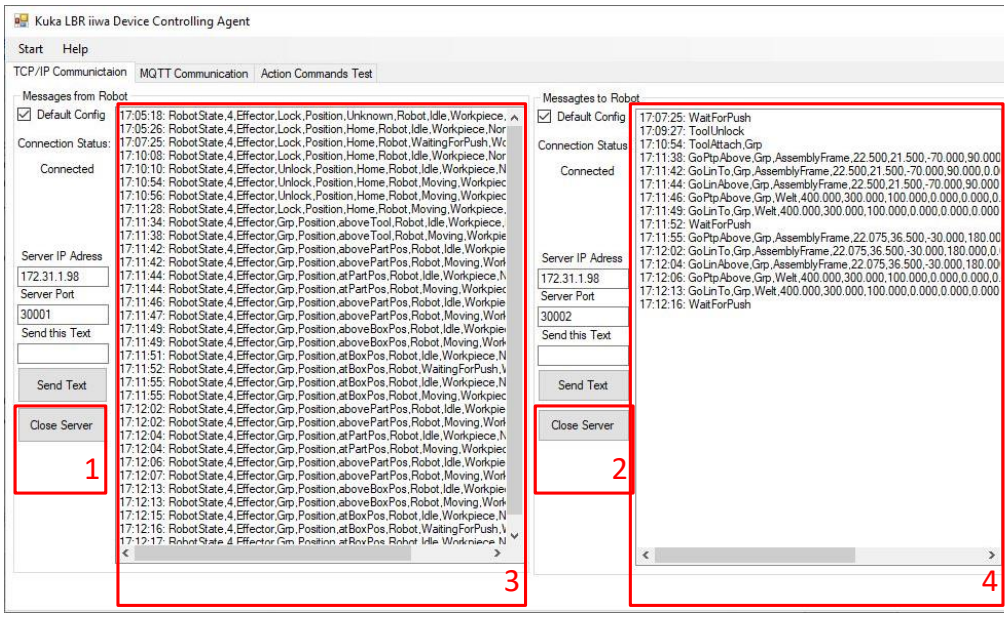

Abbildung 81: Registerkarte TCP/IP Communication.

Nach dem Aufbau beider Kommunikationsschnittstellen können über die Registerkarte ‚*Action Commands Test*‘ auch manuell Aktionskommandos an das Roboterprogramm gesendet werden. Diese Registerkarte dient vor allem zur Erprobung neuer Aktionskommandos ohne übergeordnete Steuerung. Im Feld 1 von Abbildung 82 sind die Bewegungskommandos zusammengefasst die in (T8, N1-7 u. 17)[43] beschrieben sind. In Abbildung

[43] Dieser Ausdruck (TX, NY) steht verkürzt für, vgl. Tabelle 8X, Aktionskommando Nummer Y im Anhang D.

84 ist die Methode im Roboterprogramm für die Ausführung des GoPtpTo -Aktionskommando dargestellt. Im Feld 2 und sind Aktionskommandos zum automatisierten Wechsel der Effektoren zusammengefasst (T8, N8-12).

Im Feld 3 sind Aktionskommandos zusammengefasst um neue Koordinatensysteme oder Bezugskoordinatensysteme zu erstellen (T8, N15-16).

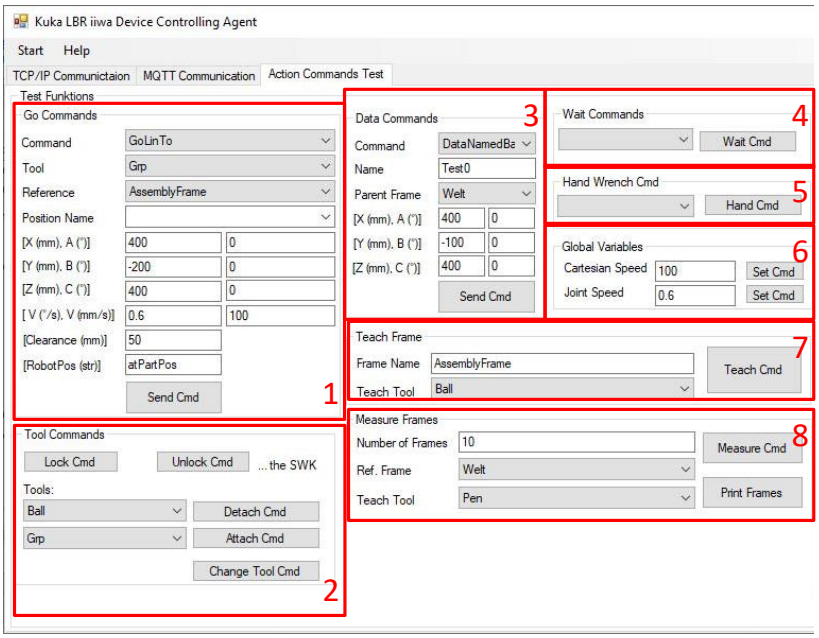

Abbildung 82: Erster Teil der Registerkarte Action Commands Test.

Im Feld 4 sind Warte-Aktionskommandos für die Koordinierung von Handlungen des Menschen mit dem RAS zusammengefasst (T8Tabelle 8, N18-21). In Feld 5 ist eine Aktionskommando zum zureichen von Schraubenschlüsseln in verschieden Schlüsselweiten implementiert (T8, N22). Über die im Feld 6 definierten Aktionskommandos können globale Variablen (kartesische und Achswinkelgeschwindigkeiten) im Roboterprogramm verändert werden (T8, N23-24). Mit dem Aktionskommandos in Feld 7, können bestehende Bezugskoordinatensystem neu eingemessen werden (T8, N13). Im Feld 8 steht ein Aktionskommando zur Verfügung, mit dem Messpunkte mit dem Roboter aufgenommen und diese als CSV-Datei exportieren werden können (T8, N14).

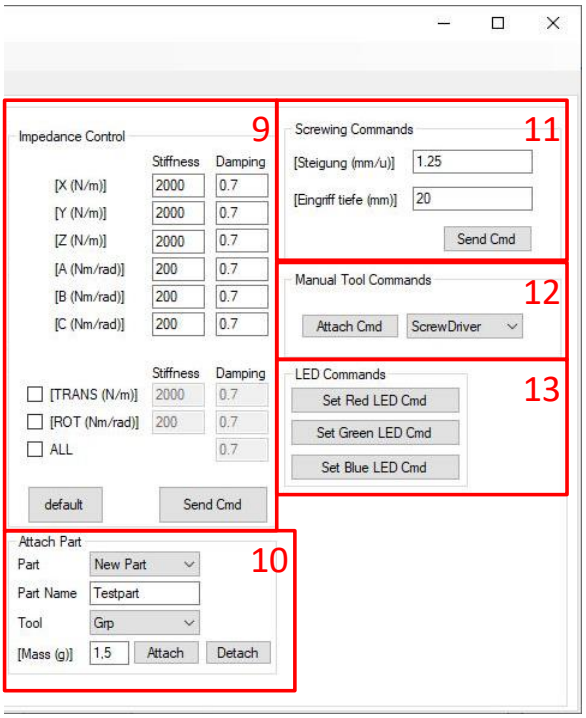

Abbildung 83: Zweiter Teil der Registerkarte Action Commands Test.

Im Feld 9 der Abbildung 83 ist das Aktionskommando für den Impedanzregelung dargestellt (T8, N27) und im Feld 10 die Funktion um Werkstücke softwaretechnisch im Roboter an die Effektoren an- und abzukoppeln (T8, N25-26). Für das Abziehen des Getriebedeckels wird das Aktionskommando in Feld 11 verwendet (T8, N28). Der manuelle Wechsel von Werkezugwechsel kann durch das Aktionskommandos in Feld 12 erfolgen (T8, N29). Weiterhin können Aktionskommandos für die Steuerung des LED-Bands am Roboterflansch im Feld 13 gesendet werden (T8, N30).

Die Aktionskommandos des LBR iiwa GSA müssen im MQTT Topic: „iiwaTopic“ veröffentlicht werden. Die Symbole <> schließen optionale Parameter ein z. B. <optionaler-Parameter>.

Tabelle 8: Aktionskommandos des LBR iiwa GSA.

Nummer	**Aktionskommando**
1	Kommando: GoHome Parameter: RelSpeed Beispiel Aktionskommando: “GoHome,0.6” Beschreibung: Der Roboter fährt seine Grundstellung in einer PTP-Bewegung mit der relativen Gelenkgeschwindigkeit (RelSpeed) an. Der Befehl ist redundant da er im Alexa Skill noch verwendet wird.
2	Kommando: GoPtpToHome Parameter: RelSpeed Beispiel Aktionskommando: “GoPtpToHome,0.6” Beschreibung: Der Roboter fährt seine Grundstellung in einer PTP-Bewegung mit der relativen Gelenkgeschwindigkeit (RelSpeed) an.
3	Kommando: GoPtpTo Parameter: Tool,RefSys,PosOri,RelSpeed,<RobotPosStateValue> Beispiel Aktionskommando: “GoPtpTo,Grp,Welt,400,100,300,0,0,0,0.6” Beispiel mit opt. Parameter: “GoPtpTo,Grp,Welt,400,100,300,0,0,0,0.6,atHandoverPos” Beschreibung: Der Roboter fährt die Position und Orientierung (PosOri) im Referenzkoordinatensystem (RefSys) mit dem Standard TCP des Werkzeugs (Tool) in einer PTP-Bewegung mit der relativen Gelenkgeschwindigkeit (RelSpeed) an. Als zusätzlichen Parameter kann ein Zeichenkette (RobotPosStateValue) übergeben werden. Wenn definiert, nimmt der Roboter nach erfolgreicher Aktionsausführung den Wert von (RobotPosStateValue) an.
4	Kommando: GoPtpAbove Parameter: Tool,RefSys,PosOri,RelSpeed,Dis,< RobotPosStateValue > Beispiel Aktionskommando: “GoPtpTo,Grp,Welt,400,100,300,0,0,0,0.6,50“ Beispiel mit opt. Parameter: “GoPtpTo,Grp,Welt,400,100,300,0,0,0,0.6,50,abovePartPos” Beschreibung: Der Roboter fährt in einer PTP-Bewegung eine Position an, die um den Wert (Dis) über der Zielposition (PosOri) liegt. Die PTP-Bewegung wird mit der relative Achswinkelgeschwindigkeit (RelSpeed) ausgeführt. Die Positionen sind bezogen zum Referenzkoordinatensystem (RefSys) und werden mit dem Standard TCP des Werkzeugs (Tool) angefahren. Wenn definiert, nimmt der Roboter nach erfolgreicher Aktionsausführung den Wert von (RobotPosStateValue) an.
5	Kommando: GoLinTo Parameter: Tool,RefSys,PosOri,CarSpeed,< RobotPosStateValue > Beispiel Aktionskommando: “GoLinTo,Grp,Welt,400,100,300,0,0,0,100” Beispiel mit opt. Parameter: “GoLinTo,Grp,Welt,400,100,300,0,0,0,100,atPartPos”

	Beschreibung: Der Roboter fährt die Position und Orientierung (PosOri) im Referenzkoordinatensystem (RefSys) mit dem Standard TCP des Werkzeugs (Tool) in einer linearen Bewegung mit der kartesischen Werkzeuggeschwindigkeit (CarSpeed) an. Wenn definiert, nimmt der Roboter nach erfolgreicher Aktionsausführung den Wert von (RobotPosStateValue) an.
6	Kommando: GoLinAbove: Parameter: Tool,RefSys,PosOri,CarSpeed,Dis,<RobotPosStateValue> Beispiel Aktionskommando: “GoLinAbove,Grp,Welt,400,100,300,0,0,0,100,50,abovePartBox” Beschreibung: Der Roboter fährt in einer linearen Bewegung eine Position an, die um den Wert (Dis) über der Zielposition (PosOri) liegt. Die lineare Bewegung wird mit der kartesischen Geschwindigkeit (CarSpeed) ausgeführt. Die Positionen sind bezogen zum Referenzkoordinatensystem (RefSys) und werden mit dem Standard TCP des Werkzeugs (Tool) angefahren. Wenn definiert, nimmt der Roboter nach erfolgreicher Aktionsausführung den Wert von (RobotPosStateValue) an.
7	Kommando: GoLinRel: Parameter: RefSys,RelPosOri,CarSpeed,<RobotPosStateValue> Beispiel Aktionskommando: “GoLinRel,Welt,400,100,300,0,0,0,100,relativePos” Beschreibung: Der Roboter verfährt relativ zum Bezugsystem (RefSys) entsprechend (RelPosOri) eine neue Position in einer linearen Bewegung an. Die lineare Bewegung wird mit der kartesischen Geschwindigkeit (CarSpeed) ausgeführt. Wenn definiert, nimmt der Roboter nach erfolgreicher Aktionsausführung den Wert von (RobotPosStateValue) an.
8	Kommando: ToolLock Parameter: Beispiel Aktionskommando: “ToolLock” Beschreibung: Das Schnellwechselsystem wird verriegelt.
9	Kommando: ToolUnlock Parameter: Beispiel Aktionskommando: “ToolUnlock” Beschreibung: Das Schnellwechselsystem wird entriegelt und im Roboterprogramm das momentane Werkzeug gespeichert in der Variable tCur entfernt.
10	Kommando: ToolAttach Parameter: Tool Beispiel Aktionskommando: “ToolAttach,Ratchet” Beschreibung: Der Effektor (Tool) wird gerüstet und als momentanes Werkzeug in der Variable tCur im Roboterprogramm gespeichert.
11	Kommando: ToolDetach Parameter: Tool Beispiel Aktionskommando: “ToolDetach,Grp” Beschreibung: Der Effektor (Tool) wird im Werkzeugbahnhof abgelegt und das momentane Werkzeug in der Variable tCur im Roboterprogramm wird entfern.

12	Kommando: ChangeTool Parameter: Tool1, Tool2 Beispiel Aktionskommando: “ToolDetach,Grp,Ratchet” Beschreibung: Der Effektor (Tool1) wird im Werkzeugbahnhof abgelegt und der Effektor (Tool2) gerüstet.
13	Kommando: TeachFrame Parameter: Tool, RefSys Beispiel Aktionskommando: “TeachFrame,Ball,AssemblyFrame” Beschreibung: Durch Handführen des Roboters und drücken der Applikationstaste am Flansch, kann der Nutzer den Ursprung, ein Punkt auf der X-Achse und ein Punkt in der X-Y-Ebene mit dem Werkzeug (Tool) einmessen und somit das Bezugskoordinatensystem (RefSys) neu vermessen.
14	Kommando: MeasureFrames Parameter: Tool, RefSys,Num Beispiel Aktionskommando: “ MeasureFrames,Ball,DriveFrame,10” Beschreibung: Durch handführen des Roboters und drücken der Applikationstaste am Flansch kann der Nutzer eine Anzahl (Num) von Punkten mit dem Werkzeug (Tool) bzgl. des Bezugsystems (RefSys) messen und als CSV-Datei exportieren.
15	Kommando: DataNamedBase Parameter: BaseName,RefSys,PosOri Beispiel: “NewFrame,Welt,400,100,300,0,0,0” Beschreibung: Im Roboter wird ein neues Arbeitskoordinatensystem mit der Position und Orientierung definiert in (PosOri) bezüglich dem Referenzkoordinatensystem (RefSys) mit dem Namen (BaseName) angelegt und gespeichert.
16	Kommando: DataNamedPosition Parameter: PosName,RefSys,PosOri Beispiel: “NewPosition,Welt,400,100,300,0,0,0” Beschreibung: Im Roboter wird die Position und Orientierung (PosOri) bezüglich dem Referenzkoordinatensystem (RefSys) mit dem Namen (PosName) als Koordinatensystem gespeichert und über den folgenden Befehl angefahren werden.
17	Kommando: GoPositionName Parameter: PosName,RelSpeed Beispiel: “GoPositionName,100” Beschreibung: Der Roboter fährt die benannte Position (PosName) mit dem Standard TCP des momentanen Werkzeugs (tCur) in einer PTP-Bewegung mit der relativen Gelenkgeschwindigkeit (RelSpeed) an.
18	Kommando: WaitForPush Parameter: Beispiel Aktionskommando: “WaitForPush”

	Beschreibung: Der Roboter lässt das LED Band am Flansch blau aufleuchten und wartet, bis der Nutzer mit einer Kraft > 20 Newton den Flansch berührt.
19	Kommando: WaitForButton Parameter: Beispiel: “WaitForButton” Beschreibung: Der Roboter lässt das LED Band am Flansch blau aufleuchten und wartet, bis der Nutzer den Applikationstaster am Roboterflansch betätigt.
20	Kommando: WaitForVoice Parameter: Beispiel: “WaitForVoice” Beschreibung: Der Roboter lässt das LED Band am Flansch blau aufleuchten. Dann wartet der Roboter, bis ein digitaler Eingang am Buskoppler (high) gesetzt wird. Der entsprechende Eingang ist mit einem Relais des Raspberry Pi3 Board verbunden. Das Raspberry Pi Relais wird wiederum durch den Sprachassistenten Alexa gesteuert. Durch ein Sprachkommando des Nutzers wird der Eingang am Buskoppler gesetzt und der Roboter beendet den Wartemodus.
21	Kommando: WaitForAny Beispiel: “WaitForAny” Beschreibung: Der Roboter lässt eine LED blau aufleuchten und wartet, bis der Nutzer den Applikationstaster am Roboterflansch betätigt, über den Sprachassistenten Alexa den entsprechenden Sprachbefehl gibt oder einen Kraftimpuls auf den Roboter ausübt.
22	Kommando: HandWrench Parameter: Size Beispiel: “HandWrench,S10” Beschreibung: Der Roboter greift den in (Size) definierten Schraubenschlüssel aus dem Werkzeugträger und fährt in einer PTP-Bewegung an eine Übergabeposition. Dort wartet der Roboter, bis der Nutzer am Schraubenschlüssel zieht. Dann öffnet sich der Greifer und gibt den Schraubenschlüssel frei. Danach fährt der Roboter zurück zur Grundstellung. Diese Roboterfertigkeit wurde in Verbindung mit dem Sprachassistenten Alexa entwickelt und verwendet.
23	Kommando: SetRelSpeed Parameter: Speed Beispiel: “SetRelSpeed,0.7“ Beschreibung: Die global definierte Variable relSpeed im Roboterprogramm erhält den Wert in (Speed). Die Variable relSpeed gibt einen Prozentsatz [0-1] der maximale Achswinkelgeschwindigkeiten für alle Achsen an.
24	Kommando: SetCarSpeed Parameter: Speed Beispiel: “SetCarSpeed,100“

	Beschreibung: Die global definierte Variable carSpeed im Roboterprogramm erhält den Wert in (Speed). Die Variable carSpeed gibt die kartesische Robotergeschwindigkeit in [mm/s] für den Tool Center Point TCP an.
25	Kommando: AttachPart Parameter:,PartName,Mass Beispiel: “AttachPart,GearSet,3.65” Beschreibung: Ein Werkstück mit der Masse (Mass) in [kg] und dem Namen (PartName) wird erstellt, der variable wCur im Roboterprogramm zugewiesen und an das momentane Werkzeug (tCur) softwaretechnisch angebracht. Dies ist erforderlich um die Gewichtskraft der Bauteile bei der Handführung zu kompensieren.
26	Kommando: DetachPart Parameter: Beispiel: “DetachPart“ Beschreibung: Das am momentanen Werkzeug (tCur) angebrachte Werkstück (wCur) wird entfernt.
27	Kommando: StartImpedance Parameter: StiffX,StiffY,StiffZ,StiffA,StiffB,StiffC,DampX,DampY,DampZ,DampA,Damp,DampC Beispiel: “StartImpedance,2000,2000,2000,200,200,200,0.7,0.7,0.7,0.7,0.7,0.7” Beschreibung: Die Impedanzregelung des Roboters wird an der aktuellen Position aktiviert. Die Steifigkeiten StiffX,Y,Z werden in [N/m] angegeben, StiffA,B,C in [Nm/rad] und die Dämpfungen DampX,Y,Z,A,B,C als double Werte. Die Gewichtskraft des momentanen Werkzeugs (tCur) und des angebrachten Werkstücks (wCur) werden berücksichtigt. Die Impedanzregelung endet mit Knopfdruck auf den Applikationstaster am Roboterflansch.
28	Kommando: Screwing Parameter: threadPitch,threadDepth Beispiel: “Screwing,1.25,20“ Beschreibung: Der Roboter geht in Impedanzregelung und dreht in 320° Schritten das Knarrenschraubwerkzeug entsprechend der Gewindesteigung (threadDepth) und der Gewindetiefe im Eingriff (threadPitch) n-mal bis die Schraube gelöst wurde.
29	Kommando: ManualToolAttach Parameter: Tool Beispiel: “ ManualToolAttach,Screwdriver“ Beschreibung: Der Effektor definiert in (Tool) wird als momentanes Werkzeug in der variable tCur im Roboterprogramm gespeichert.
30	Kommando: SetLED Parameter: Color Beispiel: “SetLED,red“ Beschreibung: Die Farbe definiert in (Color) wird am Roboterflansch befindliche LED-Band aktiviert bzw. deaktiviert (wenn es bereits leuchtet).

```
250    public void GoPtpTo(String msg) {
251
252        // change and send the transition robot state
253        this.robAPI.curRobotState.setStateValue("Robot", "Moving");
254
255        // split action command string
256        String[] subStr = msg.split(",");
257        // set new current tool
258        this.robAPI.tCur = this.getToolFromStr(subStr[1]);
259        // set new current reference coordinate system
260        this.robAPI.bCur = this.getRefFrameFromStr(subStr[2]);
261        // extract position and orientation parameters
262        String[] subPosOriStr = Arrays.copyOfRange(subStr, 3, 9);
263        // set the current target coordinate system
264        this.robAPI.fCur = this.getFrameFromStr(subPosOriStr);
265        // parse the axis speed
266        double speed = Double.parseDouble(subStr[9]);
267        // default robot position after action execution
268        String robotPos = "atCommandedPos";
269        // check for optional parameters
270        if(subStr.length == 11)
271            robotPos = subStr[10]; // set new robot position after action execution
272
273        // attach the current tool to the robot flange
274        this.robAPI.tCur.attachTo(this.robAPI.lbr.getFlange());
275
276        // get the current TCP of the tool
277        ObjectFrame OfCurTcp = this.robAPI.tCur.getDefaultMotionFrame();
278
279        // move to the commanded position
280        try {
281             OfCurTcp.move(ptp(this.robAPI.fCur).setJointVelocityRel(speed));
282        } catch (Exception e) {
283            e.printStackTrace();
284        }
285
286        // change and send the effect robot state
287        String[] States = {"Robot","Position"};
288        String[] Values = {"Idle",robotPos};
289        this.robAPI.curRobotState.setMultipleStateValues(States,Values);
290    }
```

Abbildung 84: Methode GoPptTo im Roboterprogramm.

Robotiq Greifer GSA

Die Benutzeroberflächen der folgenden GSA entspricht dem Aufbau der Benutzeroberfläche des Kuka LBR iiwa GSA, aus diesem Grund werden nun folgenden nur die Aktionskommandos der einzelnen GSA beschrieben. Die Aktionskommandos des Robotiq Greifer GSA müssen im MQTT Topic: „GrprTopic“ veröffentlicht werden. Die verfügbaren Befehle können aus entnommen Tabelle 9 werden.

Tabelle 9: Aktionskommandos des Robotiq Greifers.

Nummer	**Aktionskommando**
1	Kommando: Attached Parameter: Beispiel Aktionskommando: “Attached ” Beschreibung: Der Greifer GSA baut eine Modbus Kommunikation mit dem Greifer auf, aktiviert und initialisiert diesen und aktiviert den Positionsregelkreis des Greifers.
2	Kommando: Detached Parameter: Beispiel Aktionskommando: “Detached ” Beschreibung: Der Greifer GSA deaktiviert den Positionsregelkreis sowie den Greifer und beendet die Modbus Kommunikation mit dem Greifer.
3	Kommando: Activate Parameter: On Beispiel Aktionskommando: “Activate,true” Beschreibung: Bei On = true wird der Greifer aktiviert und startet seine Initialisierungsbewegung. Bei On = false wir der Greifer deaktiviert.
4	Kommando: LoopOn Parameter: On Beispiel Aktionskommando: “LoopOn,false” Beschreibung: Bei On = true wird der Positionsregelkreis des Greifers wird aktiviert, bei On=false wird der Positionsregelkreis des Greifers deaktiviert.
5	Kommando: Open Parameter: Beispiel Aktionskommando: “Open” Beschreibung: Der Greifer bewegt seine Greifbacken in die maximal geöffnete Position mit der maximalen Geschwindigkeit und minimaler Kraft.
6	Kommando: Close

	Parameter: Beispiel Aktionskommando: “Close” Beschreibung: Der Greifer bewegt seine Greifbacken in die geschlossene Position mit der maximalen Geschwindigkeit und minimalen Kraft.
7	Kommando: Move Parameter: GrpPosition, GrpSpeed, GrpForce Beispiel Aktionskommando: “Move,100,250,100” Beschreibung: Der Greifer bewegt seine Greifbacken in die Position (GrpPosition) mit der Geschwindigkeit (GrpSpeed) unter der Kraftentwicklung (GrpForce). Die Werte von GrprPosition, GrpSpeed und GrpForce liegen im Bereich von 0-255, 0 ist der minimale Wert, 255 der maximale Wert.

Kuka KR 125 GSA

Aktionskommandos an den Kuka KR 125 GSA müssen im MQTT Topic: „KukaKR125Topic“ veröffentlicht werden. Die verfügbaren Befehle können aus Tabelle 10 entnommen werden.

Tabelle 10: Aktionskommandos des Kuka KR 125.

Nummer	**Aktionskommando**
1	Kommando: pushDown Parameter: Beispiel Aktionskommando: “pushDown” Beschreibung: Der Kuka KR 125 presst den Getriebesatz aus indem er mit der Abtriebswelle des Antriebssystems auf den Auspressstempel fährt.
2	Kommando: MoveUp Parameter: Beispiel Aktionskommando: “MoveUp” Beschreibung: Der Kuka KR 125 fährt nach dem Auspressvorgang wieder über den Auspressstempel.

Schraubwerkzeug GSA

Aktionskommandos an den GSA des Schraubwerkzeugs müssen im MQTT Topic: „ScrewDriverTopic“ veröffentlicht werden. Die verfügbaren Befehle können aus Tabelle 11 entnommen werden.

Tabelle 11: Aktionskommandos des Schraubwerkzeugs.

Nummer	**Aktionskommando**
1	Kommando: RunAt Parameter: Speed Beispiel Aktionskommando: “RunAt, 500“ Beschreibung: Der Akkuschrauber dreht mit der in Speed definierten Drehzahl.
2	Kommando: RunAtWith Parameter: Speed,mSec Beispiel Aktionskommando: “RunAt,500,1000“ Beschreibung: Der Akkuschrauber dreht mit der in mSec definierten Zeit (in Millisekunden) und mit der in Speed definierten Drehzahl.

Sauggreifer GSA

Aktionskommandos an den GSA des Saugreifers müssen im MQTT Topic: „VacuumGrprTopic“ veröffentlicht werden. Die verfügbaren Befehle können aus Tabelle 11 entnommen werden.

Tabelle 12: Aktionskommandos des Sauggreifers.

Nummer	**Aktionskommando**
1	Kommando: On Parameter: Beispiel Aktionskommando: “On“ Beschreibung: Das Pneumatikventil wird geöffnet und Unterdruck am Sauggreifer zum Greifen erzeugt.
2	Kommando: Off Parameter: Beispiel Aktionskommando: “Off“ Beschreibung: Das Pneumatikventil wird geschlossen

Anhang E: Das MQTT Protokoll

MQTT (engl.: *Message Queuing Telemetry Transport*) ist ein standardisiertes Kommunikationsprotokoll (ISO/IEC 20922:2016), das auf TCP/IP und dem ‚Veröffentlichen/Abonnieren'-Mechanismus (engl.: *Publish*/*Subscribe*) basiert. In dieser Architektur ist ein Vermittler, (engl.: *broker*), für die Organisation des Nachrichtenflusses zwischen allen Klienten (engl.: *clients*) zuständig. Unterschiedliche Kommunikationskanäle werden durch Themen (engl.: *topics*) realisiert. Jedes Thema ist ein Kommunikationskanal, in dem interessierte Klienten in einem bestimmten Kontext miteinander kommunizieren können. Wenn sich ein Klient mit dem Vermittler verbindet, muss er angeben, welche Themen er abonnieren möchte (vgl. Abbildung 85) und welche zusätzlichen Kommunikationseinstellungen verwendet werden. Klienten können Nachrichten zu einem Thema veröffentlichen, indem sie die Nachricht an den Vermittler senden. Der Vermittler verteilt die Nachricht an alle Themenabonnenten (vgl. Abbildung 86 und Abbildung 87). In [177] gibt es eine große Auswahl von freien Software-Bibliotheken für das MQTT Kommunikationsprotokoll für unterschiedliche Programmiersprachen. In dieser Arbeit wurde die M2MQTT Bibliothek [178] von Patierno Paolo und als Vermittler der Open-Source Mosquitto Broker [179] verwendet.

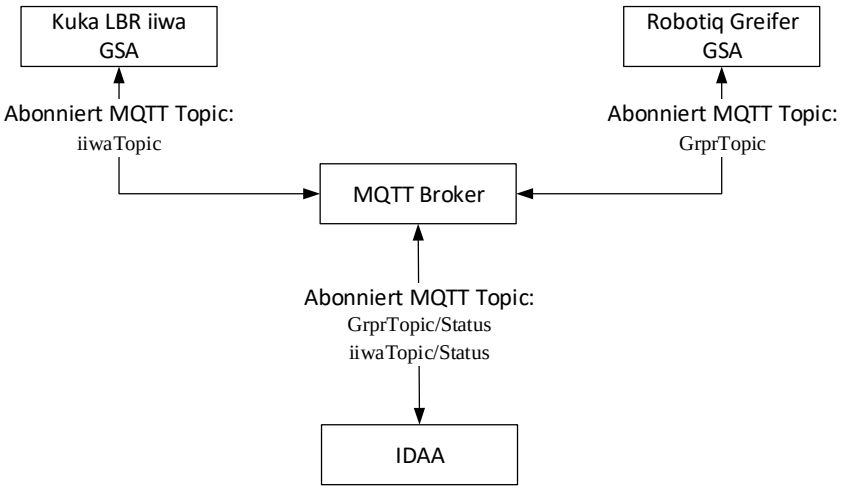

Abbildung 85: Anmeldung am MQTT Broker.

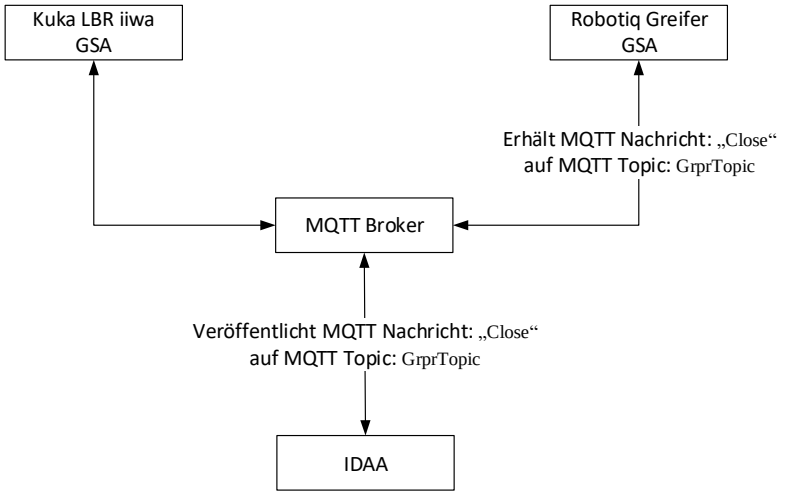

Abbildung 86: Veröffentlichen und erhalten eines Aktionskommandos.

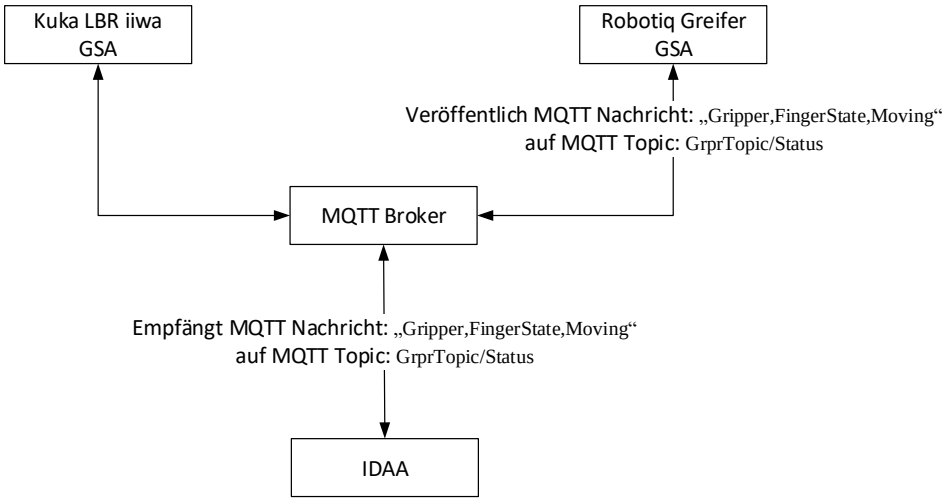

Abbildung 87: Veröffentlichen und erhalten von Gerätezustandsänderungen.

Das MQTT Protokoll bietet die folgenden Funktionen:

– *Last Will*: Im Fall des Verbindungsverlustes eines Klienten wird das hinterlegte Testament (Nachricht) zu einem Thema veröffentlicht.
– *Retained Message*: Die letzte Nachricht in einem Thema wird auf dem Vermittler gespeichert, abonniert oder verbindet sich ein Klient, erhält er automatisch die letzte gespeicherte Nachricht.

- *Quality of Service* (QoS): Es gibt drei QoS Stufen, bei QoS Stufe 0 wird die Nachricht nach dem Senden und „*Fire-and-Forget*“ - Prinzip einmal zu den Abonnenten gesendet. Die Nachricht wird maximal einmal von jedem Klient empfangen. Damit eine Nachricht (bei jedem Abonnenten) mindestens einmal ankommt, wird eine Empfangsbestätigung bei QoS Stufe 1 verwendet. Dabei kann es sein, dass die Nachricht mehrmals gesendet wird und mehrmals bei einem Klienten ankommt. Die QoS Stufe 2 verwendet eine Vier-Schritte-Verifikation um sicherzustellen, dass die Nachricht genau einmal bei jedem Abonnenten eintrifft.
- *Persistent Sessions*: Bei einer persistenten Sitzung speichert der Vermittler alle Daten eines Klienten. Zum Beispiel um im Fall eines Verbindungsverlusts den Konfigurationsaufwand beim Wiederaufbau der Verbindung zu verringern. Zusätzlich speichert der Server noch nicht bestätigten Nachrichten mit dem QoS Stufe 1 oder 2 um diese dem Klienten bei der Wiederanmeldung zu zusenden.
- *Keep Alive*: Jeder Klient überprüft zyklisch die Verbindung mit dem Vermittler, die Zeit zwischen zwei Verbindungsprüfungen wird mit dem Parameter „Keep Alive Time“ festgelegt.
- *Auto Reconnect*: Wird die Verbindung zwischen einem Klienten und dem Vermittler unterbrochen, versucht der Klient automatisch die Verbindung wiederherzustellen.
- SSL/TLS: Die Kommunikation zwischen Broker und Klienten kann verschlüsselt stattfinden.
- *High Availability*: Ist es einem Klienten nicht möglichen eine Verbindung mit dem Vermittler herstellen, kann er sich mit einem alternativen Vermittler verbinden.
- *Blocking and Non-Blocking*: Nachrichten können ereignisbasiert Empfangen und verarbeitet werden oder nach dem Request/Responce-Prinzip durch Abfragen erfolgen.
- In der aktuellen MQTT Version 5 wurden weitere Funktionalitäten für hoch skalierbare Systeme implementiert, für Details zu diesen neuen Funktionalitäten wird auf [180] verwiesen.

Anhang F: Veröffentlichungen

Jungbluth, Jan; Gerke, Wolfgang; Plapper, Peter (2016): Demontage von Elektroantrieben mit Assistenzrobotern zum wirtschaftlichen Recycling. In: Tagungsband AALE 2016. Automatisierung im Fokus von Industrie 4.0 : 13. Fachkonferenz, Lübeck. München: DIV Deutscher Industrieverlag GmbH.

Jungbluth, Jan; Gerke, Wolfgang; Plapper, Peter (2016): Agentenbasierte Steuerung der Mensch-Roboter-Interaktion. Beispiel: Zerstörungsfreie Demontage. In: atp edition - Ausgabe 12 2016 (12), S. 58–66.

Jungbluth, Jan; Gerke, Wolfgang; Plapper, Peter (2017): An Intelligent Agent-Controlled and Robot-Based Disassembly Assistant. In: IOP Conf. Ser.: Mater. Sci. Eng. 235, S. 12005. DOI: 10.1088/1757-899X/235/1/012005.

Jungbluth, Jan; Plapper, Peter; Gerke, Wolfgang (2017): Towards Intelligent Robot Assistants for the nondestructive Disassembly of End of Life Products. In: Rainer Müller, Peter Plapper, Olivier Brüls, Wolfgang Gerke, Gabriel Abba, Bassem Hichri und Matthias Vette-Steinkamp (Hg.): Robotix-Academy Conference for Industrial Robotics (RACIR) 2017. Aachen: Shaker Verlag (Berichte aus der Robotik).

Jungbluth, Jan; Plapper, Peter; Gerke, Wolfgang (2018): Recent Progress Toward Intelligent Robot Assistants for Non-Destructive Disassembly. In: Rainer Müller, Peter Plapper, Olivier Brüls, Wolfgang Gerke, Gabriel Abba, Matthias Vette-Steinkamp und Bassem Hichri (Hg.): Robotix-Academy Conference for Industrial Robotics (RACIR) 2018. 1. Auflage. Herzogenrath: Shaker (Berichte aus der Robotik).

Jungbluth, Jan; Siedentop, Karsten; Krieger, Rolf; Gerke, Wolfgang; Plapper, Peter (2018): Combining Virtual an Robot Assistants - A Case Study about Integrating Amazon's Alexa as A Voice Interface in Robotics. In: Rainer Müller, Peter Plapper, Olivier Brüls, Wolfgang Gerke, Gabriel Abba, Matthias Vette-Steinkamp und Bassem Hichri (Hg.): Robotix-Academy Conference for Industrial Robotics (RACIR) 2018. 1. Auflage. Herzogenrath: Shaker (Berichte aus der Robotik).

Anhang G: Betreute studentische Arbeiten

Awn, Shadi: „Konstruktion eines entkoppelten Robotereffektors zum Lösen von Schraubenverbindungen mit passiver indirekter Kraftregelung“, Bachelorthesis, 2015

Groß, Sebastian: „Sensorgesteuertes Lösen von Schraubenverbindungen mit dem Industrieroboter“, Bachelorthesis, 2015

Bach, Kim Magdalena: „Erstellung eines Tutorials zur Offline-Programmierung von Industrierobotern mit der Software Famos“, Fachprojekt, 2015

Flieg, Christian; Schmidt Erich: „Automatische Sequenzerzeugung für die Roboterunterstützte (De-)Montage“, Interdisziplinäre Projektarbeit, 2016

Groß, Sebastian: „Erstellen und Exportieren von Demontageinformationen mittels NX-Open, Interdisziplinäre Projektarbeit, 2016

Mayer, Manuel; Selzner, Christian: „Konstruktion und Prozesssimulation eines Effektors zum Lösen von Sicherungsringen“, Projektarbeit, 2016

Tilch, Markus: „Untersuchung von Verbindungstechniken zur roboterunterstützen Demontage“, Interdisziplinäre Projektarbeit, 2016

Tilch, Markus: „Ermittlung und Beschreibung von Produkten für die robotergestützte Demontage“, Projektarbeit, 2016

Hohmann, Steven; Bauermann, Simon: „Konstruktion eines Robotereffktors zur Handhabung von elektrischen Motoren“, Projektarbeit 2016

Fuchs, Patrick; Leschhorn, Tobias: „Untersuchung von Verbindungstechniken zur roboterunterstützten Demontage am Beispiel von aufgepressten Kugellagern und Zahnrädern“, Projektarbeit, 2017

Moorbach, Julian; Hecktor, Maximillian: „ Konstruktion einer Entschraubstation“, Interdisziplinäre Projektarbeit, 2017

Groß, Sebastian: „Entwicklung und Aufbau einer Roboter-assistierten Demontageanlage für Zusatzkühlpumpen“, Masterthesis, 2017

Held, Jonas; Schlarb, Markus; Fröhlich, Sebastian: „Planung von Mensch-Roboter Arbeitsplätzen“, Projektarbeit, 2017

Held, Jonas; Schlarb, Markus; Fröhlich, Sebastian: „Prototypentwicklung der intelligenten Kiste“, Interdisziplinäre Projektarbeit, 2017

Sontag, Wladislaw; Ermich, Sonntag: „Entwicklung eines Praktikumsversuchs für den Universal Robot 3“, Projektarbeit, 2017

Siedentopp, Karsten: „Einsatz von Sprachassistenten zur Ansteuerung von Industrieroboter“, Interdisziplinäre Projektarbeit, 2018

Krieger, Jonas: „Ergonomische Gestaltung eines Demontagearbeitsplatzes mit integriertem robotergestützen Assistenzsystem“, Bachelorarbeit 2018

Lyazidi, Hafid: „Entwicklung von Effektoren für die Mensch-Roboter Kollaboration“, Bachelorthesis, 2019

Idlasri, Abdellah: „Entwicklung einer MRK-Applikation zum Entlöten von Bausteinen aus Leiterplatten im Bereich der Leistungselektronik“ Bachelorthesis, 2019

Schmitt, Stefan: „Beschreibung, Entwicklung und Implementierung unterstützender Roboterfertigkeiten in ein intelligentes Roboterassistenzsystem für die Gestalt-erhaltende Demontage von Elektromotoren“ Masterthesis, 2019

Anhang H: Betreute Vorlesungen und Praktika

Elektrische Maschinen Praktikum WS 15/16

Robotik Praktikum WS 15/16

Praktika: Übungen zur Robotik und Mechatronik WS 15/16

Elektrische Maschinen Praktikum WS 16/17

Robotik Praktikum WS 16/17

Vorlesung: Übungen zur Robotik und Mechatronik WS 16/17

Vorlesung: Übungen zur Robotik und Mechatronik SS 17

Elektrische Maschinen Praktikum WS 17/18

Robotik Praktikum WS 17/18

Vorlesung: Übungen zur Robotik und Mechatronik WS 17/18

Vorlesung: Übungen zur Robotik und Mechatronik WS 18/19

Literaturverzeichnis

[1] T. Hafen, *Das sind die Robotik-Trends 2018: Aktuelle Trends in der Robotik.* [Online] Available: https://www.com-magazin.de/praxis/business-it/robotik-trends-2018-1483616.html?page=1_aktuelle-trends-in-der-robotik. Accessed on: Apr. 10 2019.

[2] A. Barnitzke, *MRK richtig umgesetzt: Sechs Tipps aus der Praxis.* [Online] Available: https://automationspraxis.industrie.de/allgemein/mrk-richtig-umgesetzt-sechs-tipps-aus-der-praxis/. Accessed on: Apr. 10 2019.

[3] M. Ciupek, *Tücken bei der Zulassung von Robotern.* [Online] Available: https://www.vdi-nachrichten.com/Technik-Wirtschaft/Tuecken-Zulassung-Robotern. Accessed on: Apr. 10 2019.

[4] *Indstrieroboter - Sicherheitsanforderungen Teil 1:*, DIN EN ISO 10218, 2009.

[5] *Industrieroboter - Sicherheitsanforderungen - Teil 2:*, DIN EN ISO 10218, 2011.

[6] *Sicherheit von Maschinen - Sicherheitsbezogene Teile von Steuerungen - Teil 2:*, DIN EN ISO 13849, 2012.

[7] *Roboter und Robotikgeräte - Kollaborierende Roboter*, DIN ISO/TS 15066, 2017.

[8] Konradin-Verlag Robert Kohlhammer GmbH, *Das K in MRK.* [Online] Available: https://automationspraxis.industrie.de/allgemein/das-k-in-mrk/. Accessed on: Apr. 10 2019.

[9] D. Kremer and S. Herrmann, "Arbeitsplätze für die Mensch-Roboter-Kollaboration inklusionsförderlich und wirtschaftlich gestalten," [Online] Available: https://www.baua.de/DE/Angebote/Publikationen/Fokus/Mensch-Roboter-Kollaboration.html. Accessed on: Apr. 14 2019.

[10] W. Gerke, *Technische Assistenzsysteme: Vom Industrieroboter zum Roboterassistenten*. Berlin: De Gruyter Oldenbourg, 2015.

[11] R. Müller *et al.*, *Handbuch Mensch-Roboter-Kollaboration*. München: Hanser, 2019.

[12] G. D. Hatcher, W. L. Ijomah, and J.F.C. Windmill, “Design for remanufacture: a literature review and future research needs,” *Journal of Cleaner Production*, vol. 19, no. 17-18, pp. 2004–2014, 2011.

[13] Apple, *Environmental Responsibility Report 2016.* [Online] Available: http://images.apple.com/euro/environment/d/generic/pdf/Apple_Environmental_Responsibility_Report_2016.pdf.

[14] Apple, *Meet Liam.* [Online] Available: https://www.youtube.com/watch?v=AYshVbcEmUc.

[15] U. Lange, *Ressourceneffizienz durch Remanufactoring: Industrielle Aufarbeitung von Altteilen.* [Online] Available: https://www.ressource-deutschland.de/fileadmin/user_upload/downloads/kurzanalysen/VDI_ZRE_Kurzanalyse_18_Remanufacturing_bf.pdf. Accessed on: Apr. 11 2019.

[16] D. Parker *et al.*, *Remanufactoring Market Study*. [Online] Available: https://www.remanufacturing.eu/assets/pdfs/remanufacturing-market-study.pdf. Accessed on: Apr. 11 2019.

[17] B. Brouns, *HP Laser Toner - Refurbishment of HP laser toners cartridges.* [Online] Available: http://www.remanufacturing.eu/studies/2ae67d0e001d27a01ca3.pdf. Accessed on: Apr. 11 2019.

[18] S. Butzer, S. Schötz, A. Kruse, and R. Steinhilper, “Managing Complexity in Remanufacturing Focusing on Production Organisation,” in *Enabling manufacturing competitiveness and economic sustainability: Proceedings of the 5th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2013), Munich, Germany, October 6th - 9th, 2013*, M. Zäh, Ed., Cham: Springer, 2014, pp. 365–370.

[19] E. Helmut, *Up2Date: Das Aftermarket Magazin von ZF Services.* [Online] Available: https://files.vogel.de/vogelonline/vogelonline/companyfiles/10839.pdf. Accessed on: Apr. 11 2019.

[20] Boll Automation GmbH, *MRK - Applikationslösungen.* [Online] Available: https://www.bollautomation.de/de/loesungen/mensch-roboter-kollaboration/. Accessed on: Apr. 11 2019.

[21] FANUC Austria GmbH, *STIHL geht mit kollaborativem Roboter von FANUC neue Wege.* [Online] Available: https://www.fanuc.eu/at/de/fallbeispiele-von-kunden/stihl. Accessed on: Apr. 11 2019.

[22] FANUC Austria GmbH, *Automatisierung ohne Berührungsängste: V-Zug setzt in der Bedienfeldmontage einen kollaborativen Roboter von FANUC ein.* [Online] Available: https://www.fanuc.eu/at/de/fallbeispiele-von-kunden/v-zug?return-url=https%3A%2F%2Fwww.fanuc.eu%2Fat%2Fde%2Ffallbeispiele-von-kunden. Accessed on: Apr. 11 2019.

[23] Kuka Systems GmbH, *MRK Lösungsdatenbank.* [Online] Available: https://www.kuka.com/de-de/branchen/loesungsdatenbank?filters=EE8A90CEAE6D45B3BCC8C290B637ABF0|09E98E66CA3944D59468F9075FC6ADC6. Accessed on: Apr. 11 2019.

[24] Yaskawa Motoman, *Fully automatic and electrical assembly machine for Tinkerbots.* [Online] Available: https://www.yaskawa.eu.com/en/case-studies/fully-automatic-and-electrical-assembly-machine-for-tinkerbotsc/. Accessed on: Apr. 11 2019.

[25] Technische Universität Braunschweig - Institut für Werkzeugmaschinen und Fertigungstechnik, "Untersuchungen zur automatisierten Demontage von elektronischen Altgeräten und Bestimmung der Recyclingmöglichkeiten," 1997. [Online] Available: https://www.dbu.de/projekt_01540/01_db_2409.html. Accessed on: Apr. 11 2019.

[26] K. Hohm, H. Muller Hofstede, and H. Tolle, "Robot assisted disassembly of electronic devices," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, Takamatsu, Japan, Oct. 2000, pp. 1273–1278.

[27] A. Weigl, "Requirements for robot assisted disassembly of not appropriately designed electronic products: lessons from first studies," in *1994 IEEE International Symposium on Electronics and The Environment*, San Francisco, CA, USA, May. 1994, pp. 337–342.

[28] J. Li, "Robotic disassembly of electronic components to support end-of-life recycling of electric vehicles," Dissertation, Loughborough University, 2016.

[29] J. Li, M. Barwood, and S. Rahimifard, “Robotic disassembly for increased recovery of strategically important materials from electrical vehicles,” *Robotics and Computer-Integrated Manufacturing*, vol. 50, pp. 203–212, 2018.

[30] M. Barwood, J. Li, T. Pringle, and S. Rahimifard, “Utilisation of Reconfigurable Recycling Systems for Improved Material Recovery from E-Waste,” *Procedia CIRP*, vol. 29, pp. 746–751, 2015.

[31] Technische Universität Braunschweig - Institut für Robotik und Prozessinformatik, *Roboterunterstützte Demonatge von Lithium Ionen Betterien.* [Online] Available: https://www.tu-braunschweig.de/iwf/ueberuns/portal. Accessed on: Apr. 11 2019.

[32] K. Wegener, W. H. Chen, F. Dietrich, K. Dröder, and S. Kara, “Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries,” *Procedia CIRP*, vol. 29, pp. 716–721, 2015.

[33] A. Kwade, T. Sprengler, C. Herrmann, S. Scholl, and M. Kurrat, “Recycling von Lithium-Ionen-Batterien,” TU Braunschweig, 2016. [Online] Available: https://www.erneuerbar-mobil.de/sites/default/files/2017-01/Abschlussbericht_LithoRec_II_20170116.pdf. Accessed on: Apr. 11 2019.

[34] J. Schmitt, H. Haupt, M. Kurrat, and A. Raatz, “Disassembly automation for lithium-ion battery systems using a flexible gripper,” in *15th International Conference on Advanced Robotics (ICAR), 2011: 20 - 23 June 2011, Tallinn, Estonia ; conference digest*, Tallinn, Estonia, 2011, pp. 291–297.

[35] M. Merdan, W. Lepuschitz, T. Meurer, and M. Vincze, “Towards ontology-based automated disassembly systems,” in *IECON 2010 - 36th Annual Conference of IEEE Industrial Electronics*, Glendale, AZ, USA, pp. 1392–1397.

[36] W. H. Chen, K. Wegener, and F. Dietrich, “A robot assistant for unscrewing in hybrid human-robot disassembly,” in *IEEE International Conference on Robotics and Biomimetics (ROBIO), 2014: 5 - 10 Dec. 2014, Bali, Indonesia*, Bali, Indonesia, 2014, pp. 536–541.

[37] S. Vongbunyong and W. H. Chen, *Disassembly automation: Automated systems with cognitive abilities*. Cham: Springer, 2015.

[38] H. J. Kim, S. Kernbaum, G. Seliger, and H.-J. Kim, “Emulation-based control of a disassembly system for LCD monitors,” 3-4.

[39] J. SAENZ, F. PENZLIN, C. VOGEL, and M. FRITZSCHE, “VALERI — A Collaborative mobile manipulator for Aerospace production,” in *Advances in cooperative robotics: London, UK, 12-14 September 2016*, M. O. Tokhi and G. S. Virk, Eds., New Jersey, London, Singapore, Beijing, Shanghai, Hong Kong, Taipei, Chennai, Tokyo: World Scientific, 2017, pp. 186–195.

[40] Fraunhofer IFF, *Valeri - Validation of Advanced, Collaborative Robotics for Industrial Applications: Factories of the future: Mobile manipulators for aerospace production.* [Online] Available: https://portal.effra.eu/project/1038. Accessed on: Apr. 11 2019.

[41] HORSE Consortuim, *HORSE Project.* [Online] Available: http://www.horse-project.eu/The-project.

[42] HORSE Consortuim, *Horse Complete System Design.* [Online] Available: http://www.horse-project.eu/sites/default/files/publications/HORSE-D2.2%20(Public%20Version).pdf.

[43] HORSE Consortuim, *Smart integrated immersive and symbiotic human-robot collaboration system controlled by Internet of Things based dynamic manufacturing processes with emphasis on worker safety.* [Online] Available: http://www.horse-project.eu/sites/default/files/publications/HORSE_D4.2-v1.00.pdf.

[44] SYMBIO-TIC Consortium, *SYMBIOTIC - Symbiotic Human-Robot Collaborative Assembly.* [Online] Available: www.symbio-tic.eu. Accessed on: Apr. 11 2019.

[45] C. Kardos, A. Kovács, and J. Váncza, “Decomposition approach to optimal feature-based assembly planning,” *CIRP Annals*, vol. 66, no. 1, pp. 417–420, 2017.

[46] CRO-INSPECT Consortium, *CRo-Inspect - Collaborative Robotic Solution for Advanced Inspection of Complex Composie parts.* [Online] Available: http://cro-inspect.eu/. Accessed on: Apr. 11 2019.

[47] CogIMon Consortium, *CogIMon - Cognitive Interaction in Motion.* [Online] Available: https://cogimon.eu/. Accessed on: Apr. 11 2019.

[48] Hochschule Bohn-Rhein-Sieg - Institut für Sicherheitsforschung, *BEYOND SPAI - Sichere Personendetektion im Arbeitsbereich von Industrierobotern durch ein aktives NIR-Kamerasystem.* [Online] Available: https://www.h-brs.de/de/beyond-spai. Accessed on: Apr. 11 2019.

[49] RIF e.V. – Institut für Forschung und Transfer, *MANUSERV - Vom Manuellen Prozess zum industriellen Serviceroboter.* [Online] Available: www.manuserv.de. Accessed on: Apr. 11 2019.

[50] Deutsches Zentrum für Künstliche Intelligenz (DFKI) GmbH, *Hybr-iT - Mensch Roboter Kollaboration: Hybride Teams in wandlungsfähigen, cyber-physischen Produktionsumgebungen.* [Online] Available: https://hybr-it-projekt.de/. Accessed on: Apr. 11 2019.

[51] PROFACTOR GmbH, *AssistMe - Mensch-zentrierte Assistenzrobotik in der Produktion.* [Online] Available: https://www.profactor.at/forschung/industrielle-assistenzsysteme/robotische-assistenz/projekte/assistme/. Accessed on: Apr. 11 2019.

[52] PROFACTOR GmbH, *KoMoProd - Kooperationsmodelle für assistive Mensch-Maschine-Interaktion im Produktionsprozess.* [Online] Available: https://www.profactor.at/forschung/industrielle-assistenzsysteme/robotische-assistenz/projekte/komoprod/. Accessed on: Apr. 11 2019.

[53] PROFACTOR GmbH, *FlexRoP - Flexible, assistive robot for the customized production.* [Online] Available: https://www.profactor.at/forschung/industrielle-assistenzsysteme/robotische-assistenz/projekte/flexrop/. Accessed on: Apr. 11 2019.

[54] PROFACTOR GmbH, *LERN4MRK: Modellieren, Erlernen und Abstrahieren von Prozessen für die Mensch-Roboter Kooperation.* [Online] Available: https://www.profactor.at/forschung/industrielle-assistenzsysteme/robotische-assistenz/projekte/lern4mrk/. Accessed on: Apr. 11 2019.

[55] Zentrum für Mechatronik und Automatisierungstechnik, *Effektive und sichere Mensch-Roboter-Kollaboration (4by3).* [Online] Available: https://www.zema.de/de/4by3/. Accessed on: Apr. 11 2019.

[56] Zentrum für Mechatronik und Automatisierungstechnik, *Effiziente Montage- und innovative Inbetriebnahmeprozesse in der Produktion moderner Fahrzeuge (AUTO*

IBN²). [Online] Available: https://www.zema.de/de/auto-ibn%c2%b2/. Accessed on: Apr. 11 2019.

[57] Zentrum für Mechatronik und Automatisierungstechnik, *Teilautomatisierter Roboterschweißplatz für die Einzelteilfertigung (TRSE).* [Online] Available: https://www.zema.de/de/trse/. Accessed on: Apr. 11 2019.

[58] Zentrum für Mechatronik und Automatisierungstechnik, *Cyber-physische IT-Systeme zur Komplexitätsbeherrschung einer neuen Generation multiadaptiver Fabriken.* [Online] Available: https://www.zema.de/de/smartf-it/. Accessed on: Apr. 11 2019.

[59] Zentrum für Mechatronik und Automatisierungstechnik, *Grenzüberschreitender Forschungscluster für industrielle Robotik.* [Online] Available: https://www.zema.de/de/robotix-academy/. Accessed on: Apr. 11 2019.

[60] *CHARM - Collaborative, Human-focused, Assistive Robotics for Manufacturing.* [Online] Available: http://charm.sites.olt.ubc.ca/.

[61] University of Britisch Columbia, *CHARM: Collaborative, Human-Focused, Assitive Robotics for Manufacturing.* [Online] Available: http://charm.sites.olt.ubc.ca/. Accessed on: Apr. 11 2019.

[62] R. Alami *et al.*, "Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006: Oct. 2006, [Beijing, China*, Beijing, 2006, pp. 1–16.

[63] Universität La Sapienza, *PHRIENDS: Physical Human-Robot Interaction: DepENDability and Safety.* [Online] Available: https://www.dis.uniroma1.it/~labrob/research/PHRIENDS.html. Accessed on: Apr. 11 2019.

[64] G. Ropohl, *Allgemeine Technologie: eine Systemtheorie der Technik*. s.l.: KIT Scientific Publishing, 2009.

[65] J. Rambo, H. Weber, and T. Dorsch, *Systems Engineering: Die Klammer in der technischen Entwicklung*, 0th ed., 2017.

[66] *Systems and software engineering — Architecture description*, 42010.

[67] Brockhaus Enzyklopädie, *Technik.* [Online] Available: https://brockhaus.de/ecs/enzy/article/technik-20. Accessed on: Apr. 11 2019.

[68] *IEEE International Symposium on Assembly and Task Planning*, Aug. 1995.

[69] R. Weidner, T. Redlich, and J. P. Wulfsberg, Eds., *Technische Unterstützungssysteme*. Berlin, Heidelberg: Springer Vieweg, 2015.

[70] H. Wandke, "Assistance in human–machine interaction: A conceptual framework and a proposal for a taxonomy," *Theoretical Issues in Ergonomics Science*, vol. 6, no. 2, pp. 129–155, 2005.

[71] B. Ludwig, "Interaktive Assistenzsysteme," in *Xpert.press, Planbasierte Mensch-Maschine-Interaktion in multimodalen Assistenzsystemen*, B. Ludwig, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 5–46.

[72] S. Henkel, "Entwicklung von Assistenzkonzepten unter verschieden ressourcenreichen Bedingungen," Institut für Psychologie, Humboldt-Universität zu Berlin, Berlin, 2007.

[73] K.-P. Timpe and R. Baggen, Eds., *Mensch-Maschine-Systemtechnik: Konzepte, Modellierung, Gestaltung, Evaluation,* 2nd ed. Düsseldorf: Symposion Publ, 2002.

[74] B. Ludwig, *Planbasierte Mensch-Maschine-Interaktion in multimodalen Assistenzsystemen*. Berlin: Springer Vieweg, 2015.

[75] M. R. Endsley and D. B. Kaber, "Level of automation effects on performance, situation awareness and workload in a dynamic control task," (eng), *Ergonomics*, vol. 42, no. 3, pp. 462–492, 1999.

[76] J. Frohm, V. Lindström, J. Stahre, and M. Winroth, "Levels of automation in manufacturing," in *Ergonomia - an International journal of ergonomics and human factors, ISSN 0137-4990, Vol. 30, no 3*.

[77] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Syst., Man, Cybern. A*, vol. 30, no. 3, pp. 286–297, 2000.

[78] J. M. Beer, A. D. Fisk, and W. A. Rogers, “Toward a framework for levels of robot autonomy in human-robot interaction,” (eng), *Journal of human-robot interaction*, vol. 3, no. 2, pp. 74–99, 2014.

[79] D. Surdilovic, A. Bastidas-Cruz, J. Radojicic, and P. Heyne, “Interaktionsfähige intrinsisch sichere Roboter für vielseitige Zusammenarbeit mit dem Menschen,”

[80] Deutsche Akademie der Technikwissenschaften, Ed., “Fachforum Autonome Systeme: Langfassung des Abschlussberichts,” Deutsche Akademie der Technikwissenschaften 29.10.2018, 2016. [Online] Available: http://www.hightech-forum.de/fileadmin/PDF/autonome_systeme_abschlussbericht_langversion.pdf.

[81] L. Schmidt, J. Grosche, and C. M. Schlick, *Ergonomie und Mensch-Maschine-Systeme*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[82] L. Urbas, M. Graube, J. Pfeffer, and J. Ziegler, *Einführung in die Mensch-Maschine-Systemtechnik*. [Online] Available: http://www.et.tu-dresden.de/ifa/uploads/media/MMST_001-Einfuehrung.pdf. Accessed on: Apr. 11 2019.

[83] M. A. Goodrich and A. C. Schultz, “Human-Robot Interaction: A Survey,” *FNT in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[84] Linda Onnasch, Xenia Maier, and Thomas Jürgensohn, “Mensch-Roboter-Interaktion - Eine Taxonomie für alle Anwendungsfälle,” 2016.

[85] L. Camarinha-Matos and H. Afsarmanesh, “Concept of Collaboration,” in *Encyclopedia of Networked and Virtual Organizations*, G. d. Putnik and M. M. Cruz-Cunha, Eds.: IGI Global, 2008, pp. 311–315.

[86] R. Mueller, M. Vette, and O. Mailahn, “Empowering of Assembly Processes for Human-Robot-Cooperation in Terms of Task Assignment,” in *SAE Technical Paper Series*, 2016.

[87] J. Kallweit, *BMW: Der Mensch bleibt in der Produktion unersetzbar*. [Online] Available: https://www.automobil-produktion.de/technik-produktion/produktionstechnik/bmw-der-mensch-bleibt-in-der-produktion-unersetzbar-118.html. Accessed on: Apr. 11 2019.

[88] K. F. MacDorman and H. Ishiguro, “The uncanny advantage of using androids in cognitive and social science research,” *IS*, vol. 7, no. 3, pp. 297–337, 2006.

[89] H. A. Yanco and J. Drury, “Classifying human-robot interaction: an updated taxonomy,” in *2004 IEEE international conference on systems, man & cybernetics theme: Impacts of emerging cybernetics and human-machine systems : conference proceedings*, The Hague, Netherlands, op. 2004, pp. 2841–2846.

[90] L. Mingyue Ma, T. Fong, M. J. Micire, Y. Kim Kyung, and K. Feigh, “Human-Robot Teaming: Concepts and Components for Design,” in vol. 5, *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds., Cham: Springer International Publishing, 2018.

[91] R. Watson, “Uncanny Valley – Das Phänomen des „unheimlichen Tals,” in *Spektrum Sachbuch, 50 Schlüsselideen der Zukunft*, R. Watson and R. Schneider, Eds., Berlin: Springer Spektrum, 2014, pp. 136–137.

[92] A. Gallala, “A survey: The usage of Augmented Reality in Industry,” in *Berichte aus der Robotik, Robotix-Academy Conference for Industrial Robotics (RACIR) 2018*, R. Müller et al., Eds., 1st ed., Herzogenrath: Shaker, 2018.

[93] F. Zeller, *Mensch-Roboter-Interaktion: eine sprachwissenschaftliche Perspektive*. Zugl.: Kassel, Univ., Diss, 2005. Kassel: Kassel Univ. Press, 2005.

[94] J. Blume, “Methoden und Anwendungen zur intuitiven Mensch-Roboter-Interaktion,” Lehrstuhl für Mensch-Maschine-Kommunikation, Technischen Universität München.

[95] T. Tendrik, “Communicative Aspects of Human-Robot Interaction,” Prifysgol Bangor University, 2001.

[96] J. Jungbluth, K. Siedentop, R. Krieger, W. Gerke, and P. Plapper, “Combining Virtual and Robot Assistants - A Case Study about Integrating Amazon's Alexa as a Voice Interface in Robotics,” in *Berichte aus der Robotik, Robotix-Academy Conference for Industrial Robotics (RACIR) 2018*, R. Müller et al., Eds., 1st ed., Herzogenrath: Shaker, 2018.

[97] R. C. Stalnaker, *Context and Content: Essays on intentionality in speech and thought*: Oxford University Press, 1999.

[98] H. Huettenrauch, K. Eklundh, A. Green, and E. Topp, “Investigating Spatial Relationships in Human-Robot Interaction,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006: Oct. 2006, [Beijing, China*, Beijing, 2006, pp. 5052–5059.

[99] S. Satake *et al.*, “How to approach humans?,” in *Proceedings of the 4th ACMIEEE international conference on Human robot interaction*, La Jolla, California, USA, 2009, p. 109.

[100] L. Takayama and C. Pantofaru, “Influences on proxemic behaviors in human-robot interaction,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems: IROS 2009 ; St. Louis, Missouri, USA, 10 - 15 [i.e. 11 - 15] October 2009*, St. Louis, MO, USA, 2009, pp. 5495–5502.

[101] J. Scholtz, “Theory and evaluation of human robot interactions,” in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences: Abstracts and CD-ROM of full papers : 6-9 January, 2003, Big Island, Hawaii*, Big Island, HI, USA, 2003, 10 pp.

[102] BGIA – Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung, Ed., “BG/BGIA-Empfehlungen für die Gefährdungsbeurteilung nach Maschinenrichtlinie: Gestaltung von Arbeitsplätzen mit kollaborierenden Robotern,” 2009. [Online] Available: https://publikationen.dguv.de/dguv/pdf/10002/bg_bgia_empf_u001d.pdf. Accessed on: Apr. 10 2019.

[103] T. Jacobs, *Herausforderung bei der sicheren Gestaltung von autonomen Servicerobotern für Assistenzfunktionen.* [Online] Available: https://www.baua.de/DE/Angebote/Veranstaltungen/Dokumentationen/Neue-Technologien/pdf/Mensch-Roboter-Zusammenarbeit-2018-2.pdf?__blob=publicationFile&v=2. Accessed on: Apr. 11 2019.

[104] *Instandhaltung: Begriffe der Instandhaltung*, 13306, 2001.

[105] *Instandhaltung: Grundlagen der Instandhaltung*, 31051, 2003.

[106] A. P. Barquet, H. Rozenfeld, and F. A. Forcellini, “An integrated approach to remanufacturing: Model of a remanufacturing system,” *J Reman*, vol. 3, no. 1, p. 1, 2013.

[107] R. Steinhilper, *Remanufacturing: The ultimate form of recycling*. Stuttgart: Fraunhofer-IRB-Verl., 1998.

[108] G. Biggs and B. MacDonald, "A Survey of Robot Programming Systems," in *Proceedings of the Australasian Conference on Robotics and Automation, CSIRO*.

[109] M. Xie, *Fundamentals of robotics: Linking perception to action*. Singapore: World Scientific, 2003.

[110] R. C. Arkin, *Behavior-based robotics*, 3rd ed. Cambridge, Mass.: MIT Press, 2000.

[111] R. R. Murphy, *Introduction to AI robotics*. New Delhi: Prentice-Hall of India, 2004, c2000.

[112] V. Graefe and R. Bischoff, "From ancient machines to intelligent robots — A technical evolution —," in *Instruments (ICEMI)*, Beijing, China, p. 3.

[113] K. Rajan and A. Saffiotti, "Towards a science of integrated AI and Robotics," *Artificial Intelligence*, vol. 247, pp. 1–9, 2017.

[114] R. Hartanto, *A hybrid deliberative layer for robotic agents: Fusing DL reasoning with HTN planning in autonomous robots*. Berlin: Springer, 2011.

[115] N. J. Nilsson, "The Physical Symbol System Hypothesis: Status and Prospects," in *Festschrift*, vol. 4850, *50 years of artificial intelligence: Essays dedicated to the 50th anniversary of artificial intelligence*, M. Lungarella, Ed., Berlin: Springer, 2007, pp. 9–17.

[116] B. Kuipers, E. A. Feigenbaum, P. E. Hart, and N. J. Nilsson, "Shakey: From Conception to History," in *AI Magazin*, pp. 88–103.

[117] Shanahan and Murray, *The Frame Problem.* [Online] Available: https://plato.stanford.edu/entries/frame-problem/.

[118] J. Sequeda, *Introduction to: Open World Assumption vs Closed World Assumption.* [Online] Available: https://www.dataversity.net/introduction-to-open-world-assumption-vs-closed-world-assumption/#. Accessed on: Apr. 16 2019.

[119] L. M. Aufderklamm and C. Gurschler, “The Symbol Grounding Problem,” Institut für Psychologie Leopold Franzens Universität Innsbruck, 2007. [Online] Available: https://www.uibk.ac.at/psychologie/mitarbeiter/leidlmair/the_symbol_grounding_problem.pdf. Accessed on: Apr. 23 2019.

[120] T. R. Fitz-Gibbon, *Brooks' Subsumption Architecture.* [Online] Available: http://wwwdh.informatik.uni-erlangen.de/IMMD8/Lectures/RobertinoTrento/Presentation_RyanFitzGibbon_SubsumptionArchitecture.pdf.

[121] E. Gat and R. R. Murphy, “On Three-Layer Architectures,” 2001. [Online] Available: https://pdfs.semanticscholar.org/4b84/efafce8578dfdee7d757b11b00c8b52e6135.pdf?_ga=2.116199695.809958509.1556030228-972246091.1556030228. Accessed on: Apr. 23 2019.

[122] M. Wooldridge *et al., Eds., Experiences with an architecture for intelligent, reactive agents: Intelligent Agents II Agent Theories, Architectures, and Languages*: Springer Berlin Heidelberg, 1996.

[123] Peter Bonasso, *3T Architecture.* [Online] Available: http://autsys.aalto.fi/fsr/attach/Material/Prassler_3T.pdf.

[124] C. Schlegel, *Separation of Levels and Separation of Concerns.* [Online] Available: https://robmosys.eu/wiki/general_principles:separation_of_levels_and_separation_of_concerns#fnt__1. Accessed on: Apr. 23 2019.

[125] S. J. Russell and P. Norvig, *Künstliche Intelligenz: Ein moderner Ansatz*, 3rd ed. München, Harlow, Amsterdam: Pearson Higher Education, 2012.

[126] M. Ghallab, D. S. Nau, and P. Traverso, *Automated planning and acting*. New York: Cambridge University Press, 2016.

[127] M. Ghallab, D. S. Nau, and P. Traverso, *Automated planning: Theory and practice*. Amsterdam: Elsevier/Morgan Kaufmann, 2004.

[128] J. Ferber, *Multi-agent systems: An introduction to distributed artificial intelligence*. Harlow: Addison-Wesley, 1999.

[129] J. P. Müller, A. S. Rao, and M. P. Singh, *Intelligent agents V: Agent theories, architectures, and languages ; 5th International Workshop, ATAL '98 Paris, France, July 4-7, 1998 proceedings*. Berlin, Heidelberg: Springer, 1999.

[130] A. Haddadi and K. Sundermeyer, “Belief-desire-intention agent architectures,” in *Sixth-generation computer technology series, Foundations of distributed artificial intelligence*, G. M. P. O'Hare and N. R. Jennings, Eds., New York: Wiley, 1996, pp. 169–185.

[131] M. J. Wooldridge, *An introduction to multiagent systems*, 2nd ed. Chichester: Wiley, 2009.

[132] J. Zhang, *Multi-agent based production planning and control*. Hoboken, NJ, USA: John Wiley & Sons Inc, 2017.

[133] J. Lunze, *Ereignisdiskrete Systeme: Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen*, 2nd ed. München: De Gruyter, 2012.

[134] J. Lunze, *Automatisierungstechnik: Methoden für die Überwachung und Steuerung kontinuierlicher und ereignisdiskreter Systeme : mit 416 Abbildungen, 93 Anwendungsbeispielen und 101 Übungsaufgaben*, 4th ed. Berlin, Boston: De Gruyter Oldenbourg, 2016.

[135] A. C. Schultz and L. E. Parker, Eds., *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*. Dordrecht, s.l.: Springer Netherlands, 2002.

[136] D. Vernon, *Artificial cognitive systems: A primer*. Cambridge, MA: The MIT Press, 2014.

[137] J. Laird, *The Soar cognitive architecture*. Cambridge,Mass, London, England: MIT Press, 2012.

[138] Westerkamp, *Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation.* [Online] Available: https://www.vdi.de/uploads/media/Stellungnahme_Cyber-Physical_Systems.pdf. Accessed on: Nov. 08 2017.

[139] A. L. Nicole Schmidt, *AutomationML - The Glue for Seamless Automation Engineering: AutomationML in a Nutshell*.

[140] B. H. M. Gerritsen and I. Horváth, “Current Drivers and Obstacles of Synergy in Cyber-Physical Systems Design,” in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - 2012: Presented at ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, August 12 - 15, 2015 [i.e. 2012], Chicago, Illinois, USA*, Chicago, Illinois, USA, 2012, pp. 1277–1286.

[141] S. Weyer and T. Meyer, “Open semantic meta-model as a cornerstone for the design, engineering and management of CPS-based Factories,” in *Automation ML Conference*.

[142] J. Lin, S. Sedigh, and A. Miller, “Modeling Cyber-Physical Systems with Semantic Agents,” in *IEEE 34th Annual Computer Software and Applications Conference workshops (COMPSAC-W), 2010: 19 - 23 July 2010, Seoul, Korea ; proceedings*, Seoul, Korea (South), 2010, pp. 13–18.

[143] J. Lee, B. Bagheri, and H.-A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[144] T. Bauernhansl, M. ten Hompel, and B. Vogel-Heuser, Eds., *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung, Technologien, Migration*. Wiesbaden: Springer Vieweg, 2014.

[145] W. Mahnke, A. Gossling, M. Graube, and L. Urbas, “Information modeling for middleware in automation,” in *IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA), 2011: Toulouse, France, 5 - 9 Sept. 2011*, Toulouse, France, 2011, pp. 1–7.

[146] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, “A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT),” in *2015 internet technologies and applications (ITA): Proceedings of the sixth international*

conference : Tuesday 8th-Friday 11th September 2015, Glyndwr University, Wrexham, Wales, UK, Wrexham, United Kingdom, 2015, pp. 219–224.

[147] N. Shahid and S. Aneja, “Internet of Things: Vision, application areas and research challenges,” in *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC 2017): 10-11 February 2017*, Palladam, Tamilnadu, India, 2017, pp. 583–587.

[148] ICAPS, *International Conference on Automated Planning and Scheduling.* [Online] Available: http://www.icaps-conference.org/. Accessed on: Apr. 23 2019.

[149] V. Strobel and A. Kirsch, “Planning in the Wild: Modeling Tools for PDDL,” in vol. 8736, *KI 2014: Advances in Artificial Intelligence*, C. Lutz and M. Thielscher, Eds., Cham: Springer International Publishing, 2014, pp. 273–284.

[150] S. M. LaValle, *Planning algorithms*. Cambridge: Cambridge University Press, 2006.

[151] Q. Yang, *Intelligent planning: A decomposition and abstraction based approach*. Berlin: Springer, 1997.

[152] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*. Boston, Columbus, Indianapolis: Pearson, 2016.

[153] G. Dini, F. Failli, and M. Santochi, “A disassembly planning software system for the optimization of recycling processes,” (en), *Production Planning & Control*, vol. 12, no. 1, pp. 2–12, 2001.

[154] R. E. Fikes, P. E. Hart, and N. J. Nilsson, “Learning and Executing Generalized Robot Plans,” Stanford Research Institute, Menlo Park, California.

[155] J. Provost, B. Lennartson, M. Fabian, A. Fasth, and J. Stahre, “Planning in assembly systems — A common modeling for products and resources,” in *2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, Krakow, Poland, pp. 1–8.

[156] U. Thomas, T. Stouraitis, and M. A. Roa, “Flexible assembly through integrated assembly sequence planning and grasp planning,” in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Gothenburg, Sweden, pp. 586–592.

[157] T. de Fazio and D. Whitney, “Simplified generation of all mechanical assembly sequences,” (en), *IEEE J. Robot. Automat.*, vol. 3, no. 6, pp. 640–658, 1987.

[158] L. M. Galantucci, G. Percoco, and R. Spina, “Assembly and Disassembly Planing by using Fuzzy Logic & Genetic Algorithms,” Dipartimento di Ingegneria Meccanica e Gestionale, 1 2, 2004. [Online] Available: http://www.intechopen.com/download/pdf/4082. Accessed on: Jul. 27 2015.

[159] L. S. Homem de Mello and A. C. Sanderson, “A correct and complete algorithm for the generation of mechanical assembly sequences,” in *Proceedings, 1989 International Conference on Robotics and Automation: A correct and complete algorithm for the generation of mechanical assembly sequences*, Scottsdale, AZ, USA, May. 1989, pp. 56–61.

[160] L. S. Homem de Mello and A. C. Sanderson, “AND/OR graph representation of assembly plans,” (en), *IEEE Trans. Robot. Automat.*, vol. 6, no. 2, pp. 188–199, 1990.

[161] J. Yu, L. D. Xu, Z. Bi, and C. Wang, “Extended Interference Matrices for Exploded View of Assembly Planning,” (en), *IEEE Trans. Automat. Sci. Eng.*, vol. 11, no. 1, pp. 279–286, 2014.

[162] Y. Jyh-Cheng and L. Yi-Ming, “The Structure Representation for the Concurrent Analysis of Product Assembly and Disassembly,” Department of Mechanical and Automation Engineering, 2005.

[163] N. S. Ong and Y. C. Wong, “Automatic Subassembly Detection from a Product Model for Disassembly Sequence Generation,” (en), *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 6, pp. 425–431, 1999.

[164] L. Johannsmeier and S. Haddadin, “A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes,” (en), *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 41–48, 2017.

[165] Y. F. Huang and C.S.G. Lee, “Precedence knowledge in feature mating operation assembly planning,” in *Precedence knowledge in feature mating operation assembly planning*, Scottsdale, AZ, USA, May. 1989, pp. 216–221.

[166] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, and S. Gupta, “An industrial robotic knowledge representation for kit building applications,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura-Algarve, Portugal, pp. 1365–1370.

[167] Xu, C. Wang, Z. Bi, and J. Yu, “Object-Oriented Templates for Automated Assembly Planning of Complex Products,” (en), *IEEE Trans. Automat. Sci. Eng.*, vol. 11, no. 2, pp. 492–503, 2014.

[168] *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005*.

[169] S. Chen, J. Yi, H. Jiang, and X. Zhu, “Ontology and CBR based automated decision-making method for the disassembly of mechanical products,” (en), *Advanced Engineering Informatics*, vol. 30, no. 3, pp. 564–584, 2016.

[170] K.-Y. Kim, D. G. Manley, H. Yang, and B. O. Nnaji, “Ontology-based virtual assembly model for collaborative virtual prototyping and simulation,” in *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005*, St Louis, MO, USA, pp. 251–258.

[171] N. Lohse, H. Hirani, S. Ratchev, and M. Turitto, “An ontology for the definition and validation of assembly processes for evolvable assembly systems,” in *(ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005*, Montreal, Quebec, Canada, Jul. 2005, pp. 242–247.

[172] World Wide Web Consortium (WC3), *Extensible Markup Language (XML)*. [Online] Available: https://www.w3.org/XML/. Accessed on: Apr. 23 2019.

[173] SEW EURODIRVE GmbH & Co KG, *Mechatronisches Antriebssystem MOVIGEAR*. [Online] Available: https://www.sew-eurodrive.de/produkte/dezentrale_antriebe_mechatronik/energiesparende_mechatronische_antriebe_ie4/mechatronisches_antriebssystem_movigear/mechatronisches_antriebssystem_movigear.html.

[174] S. Roßmann, *EasyModBus*. [Online] Available: http://easymodbustcp.net/en/. Accessed on: Apr. 23 2019.

[175] D. S. Nau *et al.*, "SHOP2: An HTN Planning System," *jair*, vol. 20, pp. 379–404, 2003.

[176] L. Nachmanson, S. Pupyrew, T. Dwyer, and T. Hart, *Microsoft Automatic Graph Layout.* [Online] Available: https://www.microsoft.com/en-us/research/project/microsoft-automatic-graph-layout/. Accessed on: Apr. 23 2019.

[177] Eclipse Foundation, *Eclipse Paho.* [Online] Available: https://www.eclipse.org/paho/. Accessed on: Apr. 23 2019.

[178] P. Patierno, *M2MQTT*. [Online] Available: https://m2mqtt.wordpress.com/. Accessed on: Apr. 23 2019.

[179] Eclipse Foundation, *Eclipse Mosquitto: A open source MQTT broker.* [Online] Available: https://mosquitto.org/. Accessed on: Apr. 23 2019.

[180] OASIS Consortium, *MQTT Version 5.0.* [Online] Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.