PaRTs: Automatische Programmierung in der robotergestützten Fertigung

Ludwig Nägele

Dissertation
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

Fakultät für Angewandte Informatik

2020

PaRTs: Automatische Programmierung in der robotergestützten Fertigung

Erstgutachter:	Prof. Dr. Wolfgang Reif
Zweitgutachter:	Prof. Dr. Bernhard Bauer
Drittgutachter:	Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl

Tag der mündlichen Prüfung: 08. September 2020

„Die Kunst, Pläne zu machen, besteht darin, den Schwierigkeiten ihrer Ausführung zuvorzukommen."

— Luc de Clapiers, Marquis de Vauvenargues

Danksagung

Diese Dissertation ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Softwaretechnik der Universität Augsburg entstanden. Sie wäre ohne die großartige Unterstützung vieler Menschen für mich nicht möglich gewesen. An dieser Stelle möchte ich mich bei ihnen allen bedanken.

Meinem Doktorvater Prof. Dr. Wolfgang Reif danke ich für seine Unterstützung und sein Vertrauen, sowie für viele inspirierende Gespräche – sei es auf der für mich äußerst gern besuchten jährlichen Lehrstuhl-Hütte oder spontan zwischen Tür und Angel zum Büro. Er hat mir bei meiner Forschung viel Freiraum gewährt und eine hervorragend ausgestattete technische Forschungsumgebung bereitgestellt. Darüber hinaus bedanke ich mich bei den weiteren Prüfungskommissionsmitgliedern zu meiner Promotion, Prof. Dr. Bernhard Bauer, Prof. Dr. Jörg Hähner und Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl. Für die Unterstützung bei der Anfertigung formalisierter Beschreibungen danke ich Prof. Dr. Alexander Knapp.

In meiner Zeit als wissenschaftlicher Mitarbeiter haben mich zahlreiche Kolleginnen und Kollegen auf meinem Weg zur Promotion begleitet. Das Team der Forschungsgruppe Robotik trug mit regem fachlichem Austausch und konstruktiven Ideen maßgeblich zum Gelingen meiner Dissertation bei. Dafür danke ich Christian Eymüller, Michael Filipenko, Julian Hanke, Miroslav Macho, Alexander Poeppel, Martin Schörner, Matthias Stüben und Constantin Wanninger. Besonders dankbar bin ich Dr. Andreas Angerer, Dr. Alwin Hoffmann, Dr. Andreas Schierl und Dr. Michael Vistein, die mich von Beginn meiner Arbeit an lenkend an die Hand nahmen, mir soziale und wissenschaftliche Kompetenzen in Lehre und Forschung vermittelten und mich mit ihrem herausragenden Wissen bei meiner Arbeit und meinen Publikationen unterstützten. Ohne ihre großartige Mithilfe und ihre wissenschaftlichen Beiträge, auf die ich aufbauen durfte, wäre meine Dissertation in der jetzigen Form für mich nicht möglich gewesen.

Im Kollegenkreis haben sich über die Jahre nicht nur gute Arbeitsbeziehungen entwickelt, sondern es sind – wohl auf Basis offenbarender Gespräche am Mittagstisch – familiäre Freundschaften entstanden, die selbst im Privaten sehr wichtig für mich geworden sind. Hierfür und für eine unvergessliche Zeit am Lehrstuhl danke ich der Robotik-Gruppe wie auch den weiteren Kolleginnen und Kollegen des Lehrstuhls, von denen ich Dr. Benedikt Eberhardinger, Dr. Axel Habermaier und Dr. Hella Ponsar stellvertretend nennen möchte.

Während meiner Beschäftigung am Lehrstuhl durfte ich mit engagierten Studentinnen und Studenten zusammenarbeiten. Mit ihren Abschlussarbeiten oder im Rahmen ihrer Tätigkeiten am Lehrstuhl haben sie auch mich bei meiner Forschung unterstützt. Dafür bedanke ich mich besonders bei Thomas Eibel, Jonas Geschke, Raban Popall, Luitpold Reiser und Leon Schneider.

Mit Dr. Alwin Hoffmann und Dr. Andreas Schierl durfte ich zahlreiche kreative und ebenso inspirierende Gespräche über das Themengebiet meiner Dissertation führen und die gewonnenen Erkenntnisse mit ihrer Unterstützung evaluieren. Viele Ideen

und Konzepte, die zur Lösungsfindung in meiner Dissertation geführt haben, sind im gemeinsamen Austausch mit ihnen entstanden. Dafür bin ich ihnen sehr dankbar.

Für viele hervorragende Ideen, konstruktive Anregungen und ausdauerndes Korrekturlesen meiner Arbeit danke ich ganz besonders Dr. Andreas Angerer und seiner Frau Veronika. Ihnen danke ich auch für regelmäßig festgelegte Besprechungstermine und den dadurch „sanft“ ausgeübten Druck, der für den notwendigen Fortschritt meiner Arbeit gesorgt hat. Durch ihre Unterstützung hat sich meine Dissertation sowohl inhaltlich als auch äußerlich geformt. Außerdem danke ich ihnen sowie meinen weiteren Freunden Lucas Hoffmann, Clarissa und Dr. Benedikt Eberhardinger, Constantin Wanninger, Jasmin Urbanski und Elfriede Berger für umsorgende Essenseinladungen und spontane Grillabende, bei denen mich aufbauende Gespräche zur Weiterarbeit an meiner Dissertation ermutigt und motiviert haben.

Mein größter Dank gilt meinen Eltern Regine und Dr. Rudolf Nägele. Bei Ihnen war ich zu jeder Zeit willkommen und habe stets uneingeschränkte Unterstützung erhalten. Meiner Mutter danke ich besonders für die spontan gezauberten Abendessen bei meinen meist unangekündigten Besuchen in der Schreib-Phase meiner Arbeit. Meinem Vater danke ich für die vielen Stunden, die er in die akribische, sowohl inhaltliche als auch sprachliche Korrektur meiner Dissertation investiert hat. Danke an meine Familie mit Rudolf, Regine, Margarete, Andreas, Martin, Christian, Doris, Simone, und ganz besonders Philipp, Oliver, Franz und Evi. In der für mich schwierigen Endphase der Dissertation, die mit den einschneidenden Beschränkungen der gleichzeitigen Corona-Pandemie zusammengefallen ist, haben sie für viele positive und ausgleichende Momente gesorgt.

Ludwig Nägele

Zusammenfassung

Roboter sind in der heutigen Zeit kaum mehr wegzudenken. In großen Fabriken erfüllen die leistungsfähigen Maschinen rund um die Uhr schwere und hochpräzise Aufgaben und erzielen bei der Fertigung insbesondere von Serienprodukten mit hohen Stückzahlen eine effiziente Produktion mit konstantem Durchsatz. Im Zeitalter der Industrie 4.0 soll Automatisierung auch in denjenigen Branchen Einzug erhalten, die aufgrund komplexer Fertigungsprozesse und kleiner Stückzahlen mit hoher Variabilität bis heute von manuellen Arbeitsschritten geprägt sind. Ermöglicht wird dies durch den Einsatz intelligenter und flexibler Fertigungsanlagen, die über neue Fähigkeiten und Freiheitsgrade zur Fertigung anspruchsvoller Bauteile verfügen. Mit der Flexibilität wächst jedoch auch die Komplexität bei der Programmierung. Daher werden neue Maßstäbe an die Steuerungssoftware solcher Systeme gesetzt, um den vollen Umfang an Freiheitsgraden nutzbar zu machen. Einen möglichen Ansatz bieten Methoden der automatischen Programmierung, wonach optimale Roboterprogramme vollständig vom Computer geplant werden. Doch aufgrund der vielen kombinatorischen Möglichkeiten der zur Verfügung stehenden Aktionen wächst die Planungskomplexität exponentiell. Schon bei vergleichsweise einfachen Planungsproblemen der Fertigung führt dies schnell zu einer Überschreitung der Kapazitäten eines Planers und der Ressourcen eines Computers.

Das zentrale Ergebnis dieser Dissertation ist ein allgemeiner Planungsansatz ($PaRTs$), der für ein vorliegendes Computermodell des zu fertigenden Produkts eine vollständige Automatisierung der Fertigung in einer flexiblen Roboterzelle ermöglicht. Einen wichtigen Aspekt stellt dabei die Einbeziehung von Know-How der unterschiedlichen Fachexperten dar, das abstrakt beigetragen und auf verschiedenen Ebenen zur automatischen Planung verfügbar gemacht wird. Bei der Fertigung in einer flexiblen Roboterzelle bedarf es oft der gegenseitigen Unterstützung mehrerer Roboter. Die Planung über $PaRTs$ erarbeitet derartige Kooperationen und strebt darüber hinaus eine maximale Parallelisierung gleichzeitiger Fertigungsschritte mit mehreren Robotern an, um die Gesamt-Ausführungszeit der Fertigung zu verkürzen. Ein zentraler Schwerpunkt des Planungsansatzes ist die Bewältigung der exponentiellen Planungskomplexität, die durch die Anzahl der benötigten Fertigungsschritte und der anwendbaren Fähigkeiten der Roboterzelle gegeben ist. Durch die Kombination mehrerer Strategien verfolgt $PaRTs$ eine zielgerichtete Planung, wodurch der Planungsaufwand deutlich reduziert werden kann. Anhand zweier unterschiedlicher Fallstudien werden die Ergebnisse dieser Arbeit evaluiert und über die simulative sowie die reale Ausführung der automatisch geplanten Roboterprogramme ausgewertet. Im Vergleich zu üblichen Vorgehensweisen bei der Automatisierung von Fertigungsprozessen erlaubt $PaRTs$ eine signifikante Verkürzung der benötigten Entwicklungszeit und minimiert die Ausführungszeit der Fertigung durch den Einsatz paralleler Fertigungsabläufe mit mehreren Robotern. $PaRTs$ ermöglicht dadurch eine effiziente und ökonomische Automatisierung gerade von Produktionen mit hoher Variabilität und geringen Stückzahlen und stellt somit einen vielversprechenden Ansatz für die zukünftige Roboterprogrammierung dar.

Inhaltsverzeichnis

Zusammenfassung. In diesem Kapitel wird dargelegt, welches Potential die automatische Programmierung in der industriellen Fertigung besitzt, aber auch, welche offenen Fragestellungen hierzu bestehen. Darauf aufbauend werden die Forschungsergebnisse dieser Dissertation vorgestellt, die dazu beitragen, das Potential der automatischen Programmierung nutzbar zu gestalten. In diesem Kapitel wird zudem ein Überblick über die Struktur dieser Arbeit gegeben.

1 Einleitung

Ziel dieser Arbeit ist ein ganzheitlicher Ansatz zur automatischen Programmierung von Fertigungsaufgaben mit Robotern. Anhand einer Beschreibung der Roboterzelle wird zu einem geometrischen Modell, das das zu fertigende Produkt beschreibt, automatisch ein Roboterprogramm zur Fertigung des Produkts in der Roboterzelle geplant.

Abschnitt 1.1 motiviert zunächst die Notwendigkeit der automatischen Programmierung und zeigt die Ziele auf, die mit dem in der vorliegenden Arbeit vorgestellten Planungsansatz verfolgt werden. Die erzielten Forschungsergebnisse dieser Dissertation werden in Abschnitt 1.2 vorgestellt. Abschließend gibt Abschnitt 1.3 einen Überblick über die Struktur der Arbeit.

1.1 Motivation und Ziele

Der Einsatz von Robotern ist heutzutage in der produzierenden Industrie weit verbreitet. So zählt die International Federation of Robotics (IFR) im Jahr 2018 über $2{,}4$ Millionen weltweit eingesetzte Roboter im industriellen Umfeld. Mit $422\,000$ weltweit neuen Roboterinstallationen im Jahr 2018 wurde von der IFR ein neuer Rekord verzeichnet. Für das Jahr 2022 werden bereits deutlich mehr als eine halbe Million neue Roboterinstallationen jährlich prognostiziert, mit weiterhin steigender Tendenz [62]. Industrieroboter sind darauf spezialisiert, sich wiederholende Aufgaben in der Produktion präzise und effizient in einer gleichbleibend hohen Taktung auszuführen. Sie ersetzen damit manuelle Arbeitsschritte am Fließband und ermöglichen eine zuverlässige Produktion mit großen Stückzahlen und hohen Durchsätzen. Die Programmierung von Produktionsanlagen, in denen oft viele Roboter in hintereinander geschalteten Produktionszellen zur schrittwei-

Inbetriebnehmer

sen Produktion zum Einsatz kommen, erfolgt meist über **Inbetriebnehmer**, die jeden einzelnen Fertigungsprozess oft aufwendig und zeitintensiv in die Robotersteuerung integrieren. Ein gängiges Verfahren ist dabei das Teachen von Roboterprogrammen. Hier wird der gesamte Prozessablauf einmal manuell am Roboter durchgeführt und dabei gespeichert. Laut einer Schätzung der Firma Irobiq, Kanada, zu den Kosten bei der Automatisierung macht die Anschaffung von Robotern lediglich rund 25 % der gesamten Investitionssumme aus. Der weitaus größte Anteil entfällt auf die Integrationskosten und die darin enthaltene Herstellung der Steuerungssoftware [80]. Die oft immensen Initial-Investitionen lohnen sich allerdings schnell, wenn die Automatisierungsanlage in großen Stückzahlen produziert.

Im Zeitalter der Industrie 4.0 rückt die Variabilität der automatisiert herstellbaren Produkte zunehmend in den Fokus. Fabriken werden aus intelligenten und vernetzten Fertigungsmaschinen bestehen, die durch den flexiblen Einsatz von Robotern die Fertigung variabler Produkte möglich machen. Dabei existiert nicht nur mehr ein einzelner Prozessablauf in der Roboterzelle; stattdessen durchwandern oft mehrere Werkstücke zeitgleich ihre Prozesskette in der intelligenten Fabrik und nutzen deren sogenannte cyber-physischen Produktionssysteme, die ihre Fähigkeiten über Services zur Verfügung stellen [52]. Gemäß den Anforderungen an ein solches Produktionssystem genügt der Einsatz eines einzelnen Roboters dafür oft nicht. Komplexe Fertigungsschritte oder großformatige Bauteile erfordern multifunktionale Roboterzellen, die eine dynamische Bildung von Roboterteams möglich machen, und durch die Kooperation der Roboter neue Fähigkeiten erlangen [6]. Der Zugewinn an Flexibilität und Freiheitsgraden stellt jedoch neue Herausforderungen an die Programmierung solcher Systeme: Bei der Entwicklung der Steuerungssoftware reicht es nicht aus, jeden Roboter isoliert zu betrachten; vielmehr nimmt die Koordination mehrerer Roboter des Produktionssystems einen zentralen Stellenwert ein. Dies erhöht die Komplexität der dafür notwendigen Steuerungssoftware erheblich.

Eine besondere Herausforderung sind all diejenigen Produkte, deren Herstellung von der Flexibilität multifunktionaler Roboterzellen profitiert, die aber nur in geringen Stückzahlen benötigt werden. Als Beispiel sei an dieser Stelle die Luft- und Raumfahrt genannt. Hier spielt der Einsatz von carbonfaserverstärkten Kunststoffen (**CFK**) bei der Herstellung von Flugzeugen und Raumfähren eine bedeutende Rolle. Aufgrund des geringen spezifischen Gewichts und der hohen mechanischen Belastbarkeit von CFK-Bauteilen wird dadurch eine besonders leichte Bauweise ermöglicht. Die einzelnen Fertigungsprozesse der CFK-Verarbeitung sind oft sehr komplex; um die Produktion zu automatisieren, sind hochgradig spezialisierte Werkzeuge nötig, die von Robotern geführt werden. Die Entwicklung der Steuerungssoftware hierfür ist nicht trivial und nimmt viel Zeit in Anspruch. Da die zu produzierenden Stückzahlen gleicher Komponenten nur gering ist, steht allein die Entwicklungszeit der Steuerungssoftware meist in keinem ökonomischen Verhältnis zur dadurch gegenüber der manuellen Fertigung eingesparten Produktionszeit. Aus diesem Grund werden in der CFK-Verarbeitung auch heute noch viele der Fertigungsschritte in manueller Arbeit erledigt [3, 4].

CFK

Wirft man einen Blick in andere Bereiche und Branchen, so sind auch hier viele Fertigungen nach wie vor von manuellen Arbeitsschritten geprägt. Dies ist auch im Bereich

der Montage und des dabei oft ausgeführten Fügens, also des dauerhaften Verbindens von Teilen, der Fall. Zwar wachsen die internationalen Verkaufszahlen an Industrieroboter stetig, doch besagt eine Analyse in einem Artikel des Fachmediums Computer & Automation diesbezüglich:

> *„Die Montage ist hier bisher allerdings nicht der hauptsächliche Treiber, denn nur jeder zehnte der verkauften Industrieroboter ist aktuell für Fügeprozesse im Einsatz, wenn auch mit steigender Tendenz. Dies liegt an den dynamischen und vielfältigen Herausforderungen von Montageprozessen wie etwa kleinen Losgrößen, vielen Produktvarianten und kurzen Taktzeiten." [11]*

Das zeigt, dass die Automatisierung gerade im Bereich der Montage noch großes Potential besitzt. Der Roboterhersteller KUKA ist zudem davon überzeugt, dass das Montieren zu den zentralen Aufgaben in der industriellen Fertigung zählt [73]. Der Internetseite von KUKA lässt sich die weiterführende Begründung dafür entnehmen:

> *„Das Montieren ist ein entscheidender Prozessschritt in der industriellen Fertigung. Die Gründe dafür sind immer breitere Produktpaletten, komplexere Anlagen und kürzere Innovationszyklen. Wer im Zeitalter von Industrie 4.0 wettbewerbsfähig sein möchte, muss schnell und vor allem flexibel sein." [73]*

Firmen, die aktuell noch auf manuelle Arbeitsschritte bei ihren Fertigungsprozessen angewiesen sind, sehen sich mit der fundamentalen Frage konfrontiert: Wie kann unter den erwähnten Gesichtspunkten der Variabilität und der kleinen Losgrößen eine Umsetzung der automatisierten Produktion ökonomisch und gewinnbringend funktionieren? Einen vielversprechenden Lösungsansatz stellt die bereits seit den 70er Jahren untersuchte automatische Programmierung aus dem Gebiet der Künstlichen Intelligenz [104] dar – die ideelle Vision, die Programmerstellung erfolge ohne menschliches Zutun durch den Computer selbst. In der Forschung werden schon lange Mechanismen zur **Planung** von Montageprozessen durch den Einsatz von Robotern untersucht. Als besonders erfolgversprechend haben sich dabei u. a. die zustandsbasierte Planung (state-space planning) [45] sowie die planbasierte Planung (plan-space planning) [45] mit dem wohl bekanntesten Vertreter, den Hierachical-Task-Networks (HTN [43]), herausgestellt. Beide Ansätze verfolgen das Ziel, eine adäquate Folge von Aktionen zu finden, die eine bestimmte Gesamtaufgabe erfüllt und dabei das System in einen gewünschten Zielzustand überführt. Dieser ist meist über ein Computermodell des zu fertigenden Produkts definiert, das zuvor z. B. von einem **Ingenieur** des Fachgebiets über entsprechende CAD-Software erstellt wurde. Als Aktionen stehen bei der Planung die jeweils von den Robotern durchführbaren Prozesse zur Verfügung, die einen bestimmten Systemübergang (Veränderung am Produkt und der Roboterzelle) bewirken.

Planung

Ingenieur

Für kleine zu lösende Planungsaufgaben funktionieren diese Ansätze einwandfrei. Mit der Zunahme an Komplexität des zu fertigenden Produkts wächst allerdings auch die Anzahl an Möglichkeiten, aus deren kombinatorischen Zusammensetzungen bei der Planung eine gültige Lösung gefunden werden soll. Auch mit der Hinzunahme weiterer Roboter, die neue Optionen zur Umsetzung von Montageprozessen eröffnen, vergrößert sich der Planungsraum exponentiell. Das Problem dabei ist: Ab einer gewissen Komplexität reichen die Ressourcen eines Computers nicht mehr aus, um eine Lösung

(a) Die Ausbezahlung der Reiskörner: Nur wenige Reiskörner auf den ersten sechs Feldern des Schachbretts, © Paolo Friz [9]

(b) Eine bereits immense Menge an Reiskörnern auf den ersten 24 Feldern eines großen Freiluft-Schachfelds [15]

Abbildung 1.1. Das Problem des exponentiellen Wachstums wird im Märchen zum Reiskorn auf dem Schachbrett veranschaulicht. Sehr schnell entstehen große Zahlen, die für den Mensch nur noch schwer vorstellbar sind.

in angemessener Zeit berechnen zu können. Und diese Komplexitätsgrenze ist schon bei relativ kleinen Montageaufgaben und nur wenigen eingesetzten Robotern erreicht. Die Planung in einem exponentiell wachsendem Raum an Möglichkeiten zählt zu den Problemen, die NP-schwer sind [41, 99]. Zur Verdeutlichung des exponentiellen Wachstums sei ein Märchen herangezogen, wonach ein kluger Untertan einst seinem Kaiser ein Schachbrett schenkte. Dieser, glücklich über den abwechslungsreichen Zeitvertreib, sagte daraufhin dem Untertanen zu, ihm aus Dank einen Wunsch zu erfüllen. Der antwortete: „Ich wünsche mir, dass Ihr mir das Schachbrett mit Reis auffüllet. Legt auf das erste Feld ein Reiskorn, und auf jedes weitere die jeweils doppelte Menge. Also zwei auf das zweite Feld, vier auf das dritte und acht Reiskörner auf das vierte Feld. Dieses führet fort, bis alle 64 Felder mit Reis belegt sind.“ Im Glauben, des Untertans Wunsch zeuge von größter Bescheidenheit, befahl der Kaiser seinen Dienern, diesen sogleich zu erfüllen. Doch der erste herbeigeholte Sack Reis reichte nicht aus. Auch der zweite Sack ging schnell zur Neige. Bald wurde dem Kaiser klar, dass der gesamte Vorrat in seinen Kornkammern, sogar der Ertrag des ganzen Jahres, ja der gesamte Ertrag aus hundert Jahren nicht ausreichen würde, um den einfachen Wunsch des Untertans zu erfüllen.

In der Tat ist der Verlauf exponentiellen Wachstums für den Menschen nur schwer vorstellbar. Die Anzahl von Reiskörnern auf dem i-ten Feld sind mit der Formel 2^{i-1} berechenbar, wobei die Summe der in allen vorherigen Feldern liegenden Reiskörner immer genau um Eins kleiner ist. Für alle 64 Felder ergibt sich somit eine Gesamtanzahl von $2^{64} - 1$ Reiskörnern. Das sind $18\,446\,744\,039\,484\,029\,952$ (18 Trillionen, 446 Billiarden, 744 Billionen, 39 Milliarden, 484 Millionen, 29 Tausend, 952) Reiskörner mit einem geschätzten Gewicht von 540 Milliarden Tonnen [118]. Diese Zahl ist kaum mehr vorstellbar: Würde man diese Menge Reis einen Meter hoch auslegen, so wäre die gesamte Fläche der Bundesrepublik Deutschland damit bedeckt [15].

Bezogen auf unsere Problemstellung bezüglich der automatischen Planung bedeutet dies: Ein Produkt, das mit 64 Montageschritten hergestellt wird, und für dessen Mon-

tageschritte es jeweils zwei unterschiedliche Optionen der Umsetzung gibt, führt bei der Planung gleichermaßen zu den oben erwähnten 18 Trillionen kombinatorischen Möglichkeiten. Oft soll von all diesen Möglichkeiten derjenige Plan ausfindig gemacht werden, der die höchste Zeittaktung erlaubt, am ressourcenschonendsten ist, die geringsten Betriebskosten verursacht oder die beste Produktqualität erzielt. Zur Sicherstellung eines solchen Optimalitätskriteriums muss im Zweifelsfall jeder einzelne der 18 Trillionen Pläne untersucht und mit den anderen verglichen werden. Doch diese Aufgabe überschreitet im Allgemeinen die Kapazitäten eines Planers und die Ressourcen eines Computers. Auch wenn die Entwicklung von Computern so weit voranschreitet, dass eine bestimmte Planungsaufgabe erfüllt werden kann, so übersteigt ein Bauteil, das aus nur einem zusätzlichen Einzelteil mehr besteht, bereits wieder bei Weitem die handhabbare Kapazität. Am Beispiel des Schachbretts wäre dies ein 65. Feld, das die Reismenge aller bisherigen Felder nochmals verdoppelt. Das oben beschriebene Montageproblem bestehend aus 64 Arbeitsschritten ist dabei noch vergleichsweise klein: In realistischen Anwendungsszenarien besteht die Montage eines Produkts aus oft weit mehr Arbeitsschritten, und jeder Montageschritt ist aufgrund der Flexibilität der Roboterzelle meist auf deutlich mehr als zwei Arten umsetzbar. In der bisherigen Betrachtung sind noch viele weitere Aspekte unberücksichtigt, die nochmals deutlich zur exponentiellen Zunahme der Planungskomplexität beitragen würden. So sind z. B. die inhärent kontinuierlichen Variablen des Raums und der Zeit über geeignete Diskretisierungen abzubilden, wenn etwa Kollisionen in der Ausführung der geplanten Montageschritte erkannt werden sollen. Hier ist nicht zuletzt in einem Trade-off zwischen grober Diskretisierung mit unpräzisen Kollisionsergebnissen und feiner Diskretisierung mit enormer Zunahme der Planungskomplexität abzuwägen.

Das Vorhaben zur Automatisierung von Montageprozessen in flexiblen Roboterzellen wirft viele Fragen auf. Motiviert durch die bisher ungelösten Herausforderungen bei der automatischen Programmierung von Roboterprogrammen liegt dieser Dissertation die folgende zentrale Forschungsfrage zugrunde:

> **Wie wird die Programmierung von Montageaufgaben in flexiblen Roboterzellen auf effiziente Weise ermöglicht, auch wenn die Produktion von hoher Variabilität und nur geringen Stückzahlen geprägt ist?**

Im Rahmen der vorliegenden Dissertation wird ein Lösungsansatz vorgestellt, der die in der zentralen Forschungsfrage implizit enthaltenen und zu meisternden Herausforderungen analysiert und beantwortet.

1.2 Forschungsergebnisse

Die formulierte zentrale Forschungsfrage berührt eine Vielzahl unterschiedlicher Disziplinen in den Bereichen der Robotik, der Algorithmik und Planung sowie der Softwaretechnik. Um das Ziel der Forschungsfrage erreichen zu können, sind zunächst viele untergeordnete Problemstellungen und Herausforderungen in diesen Themengebieten zu beleuchten und zu lösen. Im Rahmen der vorliegenden Arbeit ist eine Reihe an Forschungsergebnissen entstanden, die im Einzelnen die untergeordneten Problemstel-

lungen beantworten. In Kombination zu einem ganzheitlichen Lösungsansatz, der als probate Antwort auf die übergeordnete Forschungsfrage vorgestellt wird, ermöglichen sie die automatische Bestimmung von Programmen beginnend beim Computermodell des Produkts bis hin zur angestrebten automatisierten Fertigung mit Robotern. Der Lösungsansatz trägt den Namen „Planungsansatz für Roboter-Teams“ bzw. „Planning approach for Robot Teams“, kurz: $PaRTs$, in Anlehnung an die englische Namensgebung für die Montage von Bauteilen: *The assembly of parts*.

PaRTs

Die Grundidee des Planungsansatzes ist dabei die folgende: Für das zu fertigende Produkt liegt eine maschinenlesbare Beschreibung vor, die vom Computer eingelesen und von der Planung als zu erreichendes Ziel vermerkt wird. Die Produktbeschreibung wird eingehend daraufhin analysiert, welche grundsätzlichen Arbeitsschritte vorzunehmen sind, um das Produkt herzustellen. Die Liste an Arbeitsschritten wird nun in die flexible Roboterzelle mitgenommen. Für jeden einzelnen Arbeitsschritt wird aus den dort zur Verfügung stehenden Fähigkeiten eine passende Zusammensetzung gesucht, die den Arbeitsschritt konkret mit Robotern umsetzen kann. Im Gesamtergebnis erhält man ein ausführbares Programm zur automatisierten Fertigung des Produkts. Damit der Planungsansatz allgemein auf unterschiedliche Fertigungsprobleme anwendbar ist, tragen mehrere **Experten**, die sich in bestimmten Fachgebieten im Zusammenhang mit der Fertigung besonders gut auskennen, ihr Wissen und ihre Erfahrung zu einem gemeinsamen Ökosystem an Expertenmodulen bei, auf das sich die Planung stützt. Wurden die Expertenmodule einmal entwickelt, kann die Planung ohne großen Aufwand erneut und für modifizierte Bauteile angewandt werden. Die wichtigsten im Rahmen dieser Arbeit entstandenen Forschungsergebnisse sind im Einzelnen:

Experte

Modellierung des zu fertigenden Produkts: Die Fertigung spielt in vielen Fachbereichen – wie z. B. dem Automobilbau oder der Smartphoneproduktion – eine wichtige Rolle. Der jeweilige Fachbereich wird im Gebiet der Planung auch **Domäne** genannt. Zur Herstellung eines konkreten Produkts bedarf es eines domänenspezifischen Bauplans, der das zu fertigende Produkt ausführlich genug beschreibt.
In der vorliegenden Arbeit wird dafür das Format der *einheitlichen Produktbeschreibung* vorgestellt, das die Einzelteile des Produkts, deren herzustellende Verbindungsarten untereinander sowie bereits vorgegebene Prozessschritte der Montage benennt. Da der Planungsansatz $PaRTs$ unabhängig von einem zu fertigenden Produkt oder einer konkreten Domäne konzipiert ist, erfüllt die einheitliche Produktbeschreibung die Anforderung, dass sie einerseits ein allgemeines, domänenunabhängiges Schema enthält, andererseits aber die spezielle Angabe aller prozessrelevanten Daten erlaubt, die für die konkrete Fertigung benötigt werden. Der Ansatz zur Planung der Fertigung auf Basis eines abstrakten Bauplans wurde in [47] veröffentlicht.

Domäne

Modellierung der Roboterzelle und ihrer Fähigkeiten: Damit für ein gegebenes Produkt eine robotergestützte Fertigung geplant werden kann, bedarf es einer Beschreibung aller möglichen und ausführbaren Aktionen der Roboterzelle. Sowohl **Automatisierungs-** als auch **Prozessexperten** arbeiten zusammen, um die von den Robotern ausführbaren Prozesse über anwendbare *Fähigkeiten* zu beschreiben.
Die vorliegenden Arbeit spezifiziert *Fähigkeitenmodule*, die es den Experten erlauben, die von den Robotern ausführbaren Prozesse abstrakt zu beschreiben. Die Schnittstelle

Automatisierungs-experte

Prozessexperte

der Fähigkeitenmodule ermöglicht einerseits die Überprüfung, ob die Fähigkeit in einer gegebenen Situation anwendbar ist, und andererseits die Bestimmung einer konkreten Aktion, die eine optimale Ausführung gewährleistet. $PaRTs$ ist damit in der Lage, einen Fertigungsablauf bestehend aus adäquaten Roboteraktionen zu bestimmen. Die Gültigkeit des Ergebnisses wird dabei jederzeit anhand eines *Modells der Roboterzelle* sichergestellt. Die Einbeziehung von Fähigkeitenmodulen in die Planung und deren Anwendung auf eine reale Fertigung wurde in [88] und [90] veröffentlicht.

Modell der Planung: Im Zuge der Planung werden verschiedene Abfolgen von Aktionen untersucht, die sich bei ihrer Ausführung mit unterschiedlichen Änderungen auf das Produkt und die Roboterzelle auswirken. Eine automatische Planung ist nur dann realistisch, wenn sie diese Änderungen und die daraus resultierenden Zustände präzise genug kennt. Grundsätzlich besteht hier eine Abwägung zwischen Konkretion und Abstraktion: Einerseits müssen alle relevanten Eigenschaften des konkreten Planungsproblems abbildbar sein und andererseits sollen universelle Konzepte und Berechnungen (z. B. Kollisionsüberprüfungen) einheitlich darauf angewandt werden können.
$PaRTs$ stellt für die Planung ein universelles Modell bestehend aus *Attributen* und *Situationen* vor, das gleichzeitig die Konkretisierung auf ein vorliegendes Fertigungsszenario erlaubt. Eine Kernkomponente stellt die *modulare Strukturanalyse* dar, die eine automatische Ableitung solcher Attribute aus einer einheitlichen Produktbeschreibung des zu fertigenden Bauteils ermöglicht. Das der Planung zugrundeliegende Modell über Attribute und Situationen sowie deren Ableitung aus der einheitlichen Produktbeschreibung wurde in [87], [88], [54], [90] und [89] veröffentlicht.

Einbeziehung von Expertenwissen: Bereits die einheitliche Produktbeschreibung und die Fähigkeitenmodule bedürfen bei ihrer Modellierung dem Zutun verschiedener Fachexperten. **Domänenexperten**, die sich gut mit der Domäne der jeweiligen Fertigung auskennen, können meist noch weitere Vorgaben identifizieren, die die Planung weiter einschränken und dadurch den Planungsaufwand nochmals entscheidend reduzieren. Oft ist dieses Know-How abstrakt formulierbar, manchmal entsteht die Einschätzung eines Experten jedoch basierend auf seiner Intuition.

Domänenexperte

$PaRTs$ bietet mehrere Schnittstellen für die Einbeziehung von Expertenwissen nach dem Prinzip der Trennung der Zuständigkeiten an. Abstrakt formuliertes Know-How wird dabei im Zuge der Planung bedarfsgemäß automatisch angewandt, um die Planung zu spezialisieren und ungültige Teilpläne (z. B. fehlerhafte Montagereihenfolgen) frühzeitig zu verwerfen. Um Experten auch während der Planung einzubeziehen, stellt $PaRTs$ Schnittstellen zur grafischen Aufbereitung des vorliegenden Planungszustands zur Verfügung, worüber eine intuitive Eingabe der gesuchten Prozessparameter durch den Experten ermöglicht wird. Die separierte Einbeziehung von Experten unterschiedlicher Fachrichtungen in einem modularen System sowie Beispiele eines interaktiv während der Planung geführten Dialogs sind in [78], [86], [87] und [47] veröffentlicht.

Reduktion der Planungskomplexität unter Beibehaltung des Lösungsraums: Mit den bisher genannten Forschungsergebnissen ist es möglich, ein Fertigungsproblem, dessen Domäneneigenschaften und die Fähigkeiten der Roboterzelle unabhängig voneinander zu beschreiben. Die Komplexität des daraus resultierenden Planungsproblems bestimmt sich dabei maßgeblich aus der Größe des zu fertigenden Produkts und den

Freiheitsgraden, die über die Fähigkeitenmodule beschrieben sind.
In der vorliegenden Arbeit werden Ansätze vorgestellt, die den Planungsaufwand durch eine zielgerichtete Suche signifikant reduzieren, ohne dabei den prinzipiellen Lösungsraum einschränken. Erst damit wird eine effiziente Planung der Fertigung auch von anspruchsvolleren Produkten in einer flexiblen Roboterzelle möglich. Dabei leisten jeweils die Auftrennung der Planung in zwei Ebenen, die frühestmögliche Erkennung von ungültigen Plänen, die Vereinfachung von Planungssträngen über Aktionskandidaten und die Abwägung des Planungserfolgs über die Pfadzusammenführung einen signifikanten Beitrag. Ergebnisse zur Komplexitätsreduktion unter Beibehaltung des prinzipiellen Lösungsraums sind in [88], [90], [47] und [54] veröffentlicht.

Kooperation von Robotern: Die Flexibilität von intelligenten Produktionszellen wird oft dadurch erreicht, dass mehrere Roboter durch ihre Kooperation ein größeres Spektrum an ausführbaren Prozessen anbieten können. Die Art und Weise der Kooperation kann dabei unterschiedlichen Charakter haben. Während manche Prozesse in explizit definierten Kooperationen gemeinsam ausgeführt werden, existieren auch implizite, zunächst ggf. nicht offensichtliche Kooperationen, wie z. B. das Festhalten oder Unterstützen bei der Montage. Auch das parallele und unabhängige Agieren der Roboter am Bauteil zählt zu den Formen der Zusammenarbeit.
Der Planungsansatz $PaRTs$ identifiziert über eine tragfähige Suchstrategie automatisch Kombinationen von unabhängigen Fähigkeiten, die gemeinsam eine für die Herstellung des Bauteils notwendige implizite Kooperation ergeben. Um nicht – wie bei klassischen Planungsansätzen der Fall – einen rein sequentiellen Ablaufstrang an Aufgaben als Lösung zu erhalten, erarbeitet $PaRTs$ optimierte Pläne, die durch eine geeignete Parallelisierung der Roboteraktionen eine minimale Gesamtausführungszeit besitzen. Ergebnisse hierzu sind in [88], [90] und [89] veröffentlicht.

Reale Ausführung offline geplanter Programme: Die Offline-Planung von Roboterprogrammen hat viele Vorteile: Softwarebausteine helfen u. a. bei der Berechnung komplexer räumlicher Zusammenhänge und unterstützen bei der Bestimmung geeigneter Trajektorien. Zudem wird die Roboterzelle für die Programmierung nicht belegt und kann deshalb für die Fertigung anderer Produkte betrieben werden. Soll ein offline programmiertes Programm nun in der realen Roboterzelle zur Ausführung gebracht werden, besteht oft das Problem, dass die idealen Verhältnisse, die der Planung zugrunde liegen, nicht exakt mit denen der realen Roboterzelle übereinstimmen.
In der vorliegenden Arbeit werden Techniken zum Abgleich zwischen realer und modellierter Roboterzelle beschrieben, über die stets die Konsistenz der Planungsergebnisse garantiert werden kann. Dafür ist die Annotation des Roboterzellenmodells mit exakt angefertigten Vermessungsdaten möglich, die die realen Verhältnisse in der Planung widerspiegeln. Unter Erhaltung gewisser Topologieinformationen der Roboterzelle ermöglicht $PaRTs$ selbst für ein bereits fertig erstelltes Roboterprogramm die konsistente Anpassung an z. B. nachträgliche Änderungen der Roboterzelle. Ergebnisse hierzu sind in [53], [107], [106], [86] und [78] veröffentlicht.

Einige der in dieser Arbeit erzielten Forschungsergebnisse sind ganz oder teilweise im Rahmen von zwei Forschungsprojekten entstanden. Über das Verbundprojekt *AZIMUT* („Automatisierung zukunftsweisender industrieller Methoden und Technologien für

CFK Rümpfe“) mit u. a. der Premium Aerotec GmbH und dem Deutschen Zentrum für Luft- und Raumfahrt e. V. konnten erste Erkenntnisse und Ergebnisse im Bereich der automatischen Programmierung erzielt werden. Zudem war es im Rahmen des Projekts möglich, die automatisch geplanten Roboterprogramme zur CFK-Verarbeitung in einer realen Roboterzelle am Zentrum für Leichtbauproduktion (ZLP) des DLR zu evaluieren. Das anschließend gestartete DFG-Forschungsprojekt *TeamBotS* hat die Entwicklung einer toolunterstützten Methodik zur Entwicklung von Software für dynamische Roboterteams zum Ziel, wodurch eine Nutzung flexibler Fertigungszellen für die Produktion mit kleinen Stückzahlen effizient gestaltet werden soll.

1.3 Struktur der Arbeit

Die vorliegende Arbeit beginnt mit der Vorstellung zweier Fallstudien in Kapitel 3, für die über den Planungsansatz $PaRTs$ ein Roboterprogramm zur automatisierten Fertigung zu planen ist. Zu jeder Fallstudie werden die Grundlagen sowie der Stand der Technik erklärt, um anschließend die Anforderungen herauszustellen, die der Planungsansatz $PaRTs$ im Zusammenhang mit der Montageplanung für Roboter zu erfüllen hat. Untersucht wird die automatische Programmierung von Roboteranlagen zur Fertigung einer Brücke aus LEGO und zur Herstellung eines Lagenaufbaus aus CFK. In Kapitel 2 werden verwandte Arbeiten vorgestellt, die sich ebenfalls mit Themen der Planung und ähnlichen Fragestellungen auseinandersetzen.

Kapitel 4 erklärt die grundsätzliche Funktionsweise des Planungsansatzes und gibt einen Überblick über die drei Phasen der Planung sowie deren Zusammenspiel bei der Planung. Anschließend werden die drei Phasen in einem jeweils eigenen Kapitel detailliert beleuchtet: Die Strukturanalyse, beschrieben in Kapitel 5, stellt ein einheitliches Format vor, über das ein zu Montierendes Produkt beschrieben werden kann. Ebenso ist ein Verfahren vorgestellt, das aus einer auf diese Weise definierten Produktbeschreibung automatisch sämtliche Daten analysiert, die für die Planung relevant sind. Kapitel 6 stellt die Vorgehensweise der in Phase 2 stattfindenden Prozessplanung vor, die das Planungsproblem zunächst von der Automatisierung abstrahiert und lediglich aus Sicht der Domäne betrachtet und löst. In der anschließenden Phase der Fertigungsplanung werden nun auch Roboter einbezogen, um die bisher abstrakten Pläne zu konkretisieren und ausführbare Fertigungspläne zu erhalten. Kapitel 7 beschreibt hierzu die Einbeziehung eines Modells der Roboteranlage sowie von Fähigkeiten, die sämtliche in der Roboterzelle ausführbaren Prozesse darstellen. Es wird außerdem dargelegt, auf welche Weise zu jeder Phase das Know-How der unterschiedlichen Fachexperten einbezogen und der Planung verfügbar gemacht wird. In Kapitel 8 wird anschließend darauf eingegangen, wie ein über $PaRTs$ erzeugtes Fertigungsprogramm simuliert und anschließend in einer realen Roboterzelle zur Ausführung gebracht werden kann.

Die zu Beginn vorgestellten Fallstudien der Domänen LEGO und CFK werden in Kapitel 9 und 10 wieder aufgegriffen, um anhand derer den erläuterten Planungsansatz $PaRTs$ zu evaluieren. Dabei wird sowohl die Anwendung des Planungsansatzes als auch das Ergebnis der simulierten und teils realen Ausführungen dargelegt und ausgewertet. Die Arbeit schließt mit einem Fazit und einem Ausblick in Kapitel 11.

Zusammenfassung. In diesem Kapitel werden Arbeiten beleuchtet, die mit den Forschungsergebnissen der vorliegenden Arbeit verwandt sind. Dabei wird insbesondere auf den bisherigen Stand und die noch offenen Fragestellungen zu den Themen der Planung für (Multi-)Roboter, der Einbeziehung von Expertenwissen zur Optimierung der Planung sowie der Simulation bei der Offline-Programmierung eingegangen. Das Kapitel schließt mit einem Fazit für den Ansatz der vorliegenden Arbeit.

2 Verwandte Arbeiten

Die Planung von Steuerungssoftware zur Ausführung in Roboteranlagen wird in der Forschung mit verschiedenen Ansätzen untersucht. Einige Ansätze konzentrieren sich hauptsächlich auf diskrete Planungsverfahren zur Lösung der Aufgabenzuteilung an die einzelnen Roboter. Dabei wird das Planungsproblem oft über ein zustandsbasiertes System im *state-space* über Zustände und Aktionen (Transitionen) beschrieben. Die hierbei kombinatorisch möglichen Abfolgen an Aktionen werden über entsprechende Algorithmen (z. B. SAT-Solver) untersucht, um eine gültige und ggf. optimale Lösung zu finden. Ein anderer, ebenso weit verbreiteter Ansatz ist die Planung über Hierarchical Task Networks (HTN). Hier erfolgt die Suche im *plan-space*; im Gegensatz zur zustandsbasierten Planung werden keine Zustände sondern Pläne untersucht. Dabei erfolgt eine Verfeinerung eines zunächst abstrakten Plans (oder einer abstrakten Aktion) anhand von Dekompositionsregeln schrittweise zu atomaren Einzelaktionen [43, 46, 56, 75].

2.1 Symbolische Planung

Eine Möglichkeit, ein über Zustände und Aktionen formuliertes Planungsproblem optimal zu lösen, bietet die informierte Suche über A* [75]. A* ist ein in der Forschung weit verbreiteter Ansatz und in unterschiedlichen Bereichen für das effiziente Finden optimaler Pläne gut geeignet. Die Suche fokussiert sich auf das optimale Ergebnis anhand eines Fitnesswerts, der sich aus den bisherigen Kosten aller Aktionen sowie den geschätzten Kosten der noch zu erwartenden Aktionen zusammensetzt. Da eine solche Schätzung

insbesondere für zurückzulegende Strecken über die euklidische Distanz einfach zu bestimmen ist, wird A* oft zur optimalen Routenplanung mobiler Roboter (z. B. durch die Korridore einer Fabrik) eingesetzt [29, 120]. Es werden auch modifizierte Varianten von A* eingesetzt, um mit dynamischen Hindernissen umgehen zu können [124, 132].

In anderen Bereichen der Robotik – besonders bei der industriellen Anwendung von Robotern – ist eine Abschätzung der Kosten der noch ausstehenden Roboteraktionen jedoch deutlich komplexer und hängt zudem maßgebend vom Kriterium ab, nach dem die Optimalität eines Plans bewertet werden soll. Die immense Komplexität bei der Planung optimaler Abfolgen kann hier mit klassischen Suchalgorithmen allein nicht universell reduziert werden; es sind zusätzliche Problem- und Produkt-spezifische Heuristiken erforderlich, die spezifisches Know-How über die Domäne und die Prozesse zur Vereinfachung der Planung integrieren [46].

Ein Ansatz, der keine explizite Angabe einer Heuristik fordert, ist der in Stanford entwickelte STRIPS-Formalismus [37] und die darauf aufbauende Planning Domain Definition Language (PDDL, [82]). In einer einheitlichen Sprache werden die Domäne und das zu lösende Planungsproblem beschrieben und intern für die Planung in eine Zustandsmaschine übersetzt.. Die möglichen Aktionen sind mit Vorbedingungen annotierbar, um ihre Anwendbarkeit auf bestimmte Zustände einzuschränken. Über Nachbedingungen können die Effekte einer Aktion definiert werden. Es stehen verschiedene hochgradig optimierte SAT-Solver zur Verfügung, über die das Planungsproblem anschließend gelöst werden kann. PDDL hat sich in der Forschung als ein Standard im Bereich der symbolischen Planung etabliert und spielt u. a. bei der International Planning Competition (IPC) eine tragende Rolle [65]. Auch in der Robotik findet der Formalismus Anwendung: Huckaby et al. [59] schlagen eine Taxonomie von Basisfähigkeiten vor, die sie in die Planung über klassisches PDDL einbeziehen. Auch im EU-Projekt SkillPro erfolgt die Aufgabenplanung klassisch mit PDDL [97]. Dabei wird jedoch eine unmittelbare Lösung ausgewählt und keine global optimale Anordnung der Fertigungsschritte betrachtet. Eine interessante Erweiterung um weiche und harte Deadlines, die Nicht-Linearitäten in der Kostenfunktion während der Planung erlaubt, wird von Benton et al. [17] vorgeschlagen. Diese können eingesetzt werden, um qualitative Aspekte in Fertigungsplänen auszudrücken, wie z. B. eine besondere Behandlung empfindlicher Bauteile. Auch unabhängig von PDDL werden auf der Ebene der diskreten Aufgabenplanung oft SAT-Solver eingesetzt (vgl. [27, 63, 64]).

Die symbolische Planung erfordert eine Modellierung all derjeniger Eigenschaften, die bei der Planung des optimalen Ergebnisses berücksichtigt werden sollen. Da die Fertigung mit Robotern üblicherweise hochgradig dynamisch und nicht-linear ist, kann sie nicht mit allen ihren Eigenschaften in einer symbolischen Planung ausgedrückt werden [67, 70]. Wird das Planungsproblem sehr detailliert modelliert, führt dies schnell zu einem nicht mehr handhabbaren Zustandsraum. Eine detailarme Modellierung hingegen verringert zwar den Zustandsraum, reduziert jedoch auch die Qualität der Planung und resultiert in ggf. nicht-optimalen oder sogar unrealistischen Ergebnissen. Daher reicht es bei der Planung von Roboteraufgaben meist nicht aus, die symbolische Planung isoliert zu betrachten.

2.2 Kombinierte Planung und Reduktion der Planungskomplexität

In den letzten Jahren hat sich ein Trend entwickelt, die Aufgabenplanung für roboterbasierte Fertigungsprozesse nicht isoliert, sondern in Kombination mit der Bewegungsplanung zu betrachten [130]. Sacerdoti [105] schlägt erstmals eine hierarchische Untergliederung in Ebenen unterschiedlicher Abstraktion vor, um Lösungen effizienter berechnen zu können. Höller et al. [55] erweitern den oben erwähnten Formalismus PDDL zu HDDL, womit auch hierarchische Planungsprobleme modelliert werden können. Ein vielzitierter Ansatz wurde von Kaelbling und Lozano-Pérez [66] vorgestellt und später u. a. von Levihn et al. [76] erweitert. Dabei wird die Planung auf Aufgabenebene nur in begrenztem Umfang durchgeführt. Die Ergebnisse werden anschließend ausgeführt und ausgehend von der dann entstandenen Situation auf Aufgabenebene weiter geplant. Schoen and Rus [109] führen eine Aufgabenplanung während der Montage durch, um abhängig von den jeweiligen Gegebenheiten (Verfügbarkeit von Teilen) die zu erledigenden Aufgaben ideal auf die zur Verfügung stehenden mobilen Roboter aufzuteilen. Durch das schrittweise Vorgehen wird die Komplexität der Planung gering gehalten und es ist jederzeit eine dynamische Neuplanung auf Basis der tatsächlich auftretenden Effekte möglich. Da die Planung bei beiden Ansätzen schrittweise erfolgt, ist der erzielte Plan aus globaler Sicht jedoch nicht garantiert optimal.

Bhatia et al. [18] schlagen stattdessen eine zweischichtige Planung mit einer diskreten Aufgabenplanung und einer sampling-basierten Bewegungsplanung vor, sodass eine durchgängige Planung schon vor der Ausführung erfolgen kann. Von He et al. [51] wird dieser Ansatz um eine zusätzliche Koordinationsebene erweitert. Ein weiterer Ansatz zur Beschreibung von Fertigungsabläufen wird von Stenmark und Malek [115] über Assembly Graphs vorgeschlagen, die sie in ein Framework zur Planung und Ausführung basierend auf einer industriellen Toolchain einbetten. In [114] erweitern die Autoren ihren Ansatz um Eingaben von Fertigungsaufgaben in natürlicher Sprache. Das Konzept der HTNs wird z. B. von Wolfe et al. [127] aufgegriffen, um die Aufgabenplanung für Roboter bis auf kinematischer Ebene durchzuführen. Weser et al. [125] erweitern in ihrem Ansatz die HTNs, um für einen Serviceroboter in einer dynamischen Umwelt zu planen. AND/OR-Graphen [57] werden von Thomas et al. [119] für die Planung von Montageaufgaben zur Ausführung über einen Roboter eingesetzt. Xiao und Ji [129] präsentieren speziell für kraftbasierte Manipulationsaufgaben sogenannte Contact State Graphs, die z. B. von Meeussen et al. [83] zur Erzeugung von Bewegungsplänen für kraftgeregelte Aktionen angewandt werden. Neben den bisher vorgestellten integrierten Ansätzen existieren weitere Arbeiten (vgl. [28, 42, 112]), die mittels Definition einer allgemeinen Schnittstelle die Integration beliebiger Bewegungsplaner mit verschiedenen Ausführungsplanern ermöglichen wollen. In allen erwähnten Arbeiten beschränkt sich die Betrachtung jedoch auf einen einzelnen Roboter, der die Ausführung eines Ergebnisses der Planungsphase übernimmt. Der Einsatz mehrerer Roboter und deren Koordination wird nicht betrachtet.

2.3 Planung für Multi-Roboter

Auch die Fertigung mit mehreren Robotern wird in der Forschung betrachtet: Tercio [48] ist ein Algorithmus zur Zuweisung von Aufgaben an mehrere gleichartige Robotersysteme, die am selben Bauteil arbeiten. Um Kollisionen der Roboter entgegenzuwirken, werden große Überlappungen der Arbeitsräume vermieden. Yun und Rus [131] betrachten die Aufgabenverteilung auf zwei verschiedene Robotertypen zur Fertigung von Bauteilen mit vielen Einzelteilen. Dabei werden zunächst auf Basis einer Gleichverteilung die optimalen Positionen der Fertigungsroboter festgelegt. Anschließend wird auf dieser Grundlage für Zulieferroboter bestimmt, an welchen Fertigungsroboter sie das nächste Einzelteil ausliefern. Knepper et al. [69] stellen ihre Planung zur Montage von Möbeln mit mehreren mobilen Robotern vor, die in bestimmten Fertigungsschritten gemeinsame Aufgaben (z. B. Umdrehen des Möbelstücks) ausführen. Sie verwenden dazu eine objektorientierte Erweiterung von PDDL, wobei die gemeinsam ausgeführten Aufgaben in konkreten Kooperationen vorgegeben sind.

Während in diesen Ansätzen die kombinierte geometrische Bewegungsplanung nicht betrachtet wird, versuchen von Bourne et al. [21] zur Fertigung großer Strukturen mit Roboterteams eine Verbindung von Planung und Scheduling herzustellen. Dabei wird der oben erwähnte Ansatz von Kaelbling und Lozano-Pérez [66] um einen dynamischen Scheduler erweitert, der sich auf die von der Planung generierten Assembly Graphs stützt. Im Fokus steht hier die Erreichung der für die Fertigung nötigen Präzision mit Teams aus mobilen Robotern zur Laufzeit. Variierende Kooperations-Aufgaben werden nicht betrachtet, sodass dieser Aspekt nicht in die Planung einbezogen wird. Im Rahmen des EU-Projekts SMErobotics [33, 95] untersuchen Nottensteiner et al. [92] den automatisierten Nachbau vorgegebener item-Strukturen. Für die von einem Anwender zusammengesetzte Struktur wird zunächst eine optische Erkennung durchgeführt und anschließend dafür mögliche Montageabfolgen über eine AND/OR-Graphen erstellt. Es werden parametrisierbare Fähigkeiten angewandt, um mit zwei Robotern die item-Konstruktion nachzubauen. Ein Roboter hält dabei die Teilkonstruktion, während der andere Roboter ein weiteres Teil anbringt. Die Kooperation der Roboter ist hierbei in eine einzelne Fähigkeit codiert und wird deshalb vom Planungsalgorithmus nicht explizit behandelt. Kast et al. [67, 68] verfolgen einen hierarchischen Ansatz, um Montageaufgaben für zwei Roboter anhand von deklarativem und prozeduralem Wissen sowie anhand einer besonderen Backtracking-Strategie zu planen. In ihrer Evaluation untersuchen die Autoren ein Beispiel zur Montage von bis zu vier Komponenten auf einer Hutschiene.

Die erwähnten Arbeiten planen in erster Linie die Aufgaben der Fertigung und deren Zuteilung zu mehreren Robotern. Kooperativ auszuführende Aufgaben werden dabei entweder nicht betrachtet oder sind nicht Teil der globalen Planung, da sie in atomaren Fähigkeiten fest vorgegeben werden. Die existierenden Freiheitsgrade flexibler Roboterzellen sind allerdings selten allumfänglich über explizite Kooperationsmöglichkeiten abzubilden, da sie jede Form von Kooperation für unterschiedliche Roboter-Konstellationen im Detail implementieren müssten. Das Potential flexibler Roboterzellen kann

nur dann ausgeschöpft werden, wenn auch dynamisch zu bildende Kooperationen von Robotern universell bei der Planung berücksichtigt werden.

Dies untersuchen z. B. Suárez-Ruiz et al. [117] mit ihrem Ansatz zur Montageplanung eines Stuhls über kooperativ auszuführende Montageaufgaben zweier Roboter. Während die Abfolge der Montageschritte vorgegeben ist, werden die auszuführenden Prozesse automatisch berechnet. Für die Erstellung eines Plans benötigt der Ansatz in etwa 11 Minuten, was die Ausführungszeit der Montage bereits übersteigt. Würde die Ermittlung der optimalen Abfolge der Montageschritte in die Planung einbezogen werden, wäre der benötigte Zeitaufwand immens. Rodrıguez et al. [101] erweitern die zuvor bereits genannte Arbeit zur Montage von Konstruktionen aus item-Profilen [92] um einen mehrschichtigen Planungsansatz, der zunächst die Plausibilität von schnell zu berechnenden abstrakten Plänen überprüft und diese nur im Bedarfsfall in einer konkreteren und teureren Planungsebene verfeinert. Auf diese Weise wird die Gesamtzeit der Planung deutlich reduziert. Mit der optionalen Angabe eines zusätzlichen produktspezifischen Regelsets, das bestimmte Montagereihenfolgen einschränkt, kann die Planungsdauer nochmals verkürzt werden. Dennoch bleibt die Planungskomplexität NP-schwer. Bei einer zu montierenden Struktur bestehend aus fünf Profilen und sechs Winkeln liegt die Planungszeit trotz des produktspezifischen Regelsets bereits im Minutenbereich.

Die kombinierte Planung auf Aufgaben- und Bewegungsplanungsebene hat immer mit der exponentiellen Komplexität zu kämpfen. Daher adressieren alle Arbeiten zur Montage lediglich Montageaufgaben mit vergleichsweise nur wenigen Einzelteilen. Verglichen mit aktuellen Produktionen der Industrie und der Anzahl von Teilen, aus denen sich z. B. ein Smartphone, ein Auto oder ein Flugzeug zusammensetzt, reicht die Performanz der heutigen Ansätze zur automatischen Montageplanung noch nicht aus. Viele Arbeiten, die symbolische Planung und Bewegungsplanung für mehrere Roboter integrieren, betrachten zudem noch keine Optimierung anhand parallel ausführbarer Fertigungsschritte im Wechsel mit kooperativ ausgeführten Aktionen. In der vorliegenden Arbeit wird eine im Rahmen der Montage universelle Planungsstrategie vorgeschlagen, die Roboterkooperationen und Parallelausführungen berücksichtigt und das beherrschbare Komplexitätslevel mit signifikant mehr handhabbaren Einzelteilen deutlich anhebt.

2.4 Einbeziehung von Expertenwissen und Fähigkeiten

Bei der manuellen Fertigung arbeiten oft Fachexperten aus unterschiedlichen Bereichen zusammen, die ihr jeweiliges Know-How in entsprechenden Fertigungsschritten beitragen. Roboter werden als effiziente Alternative zu den manuellen Arbeitsschritten angesehen, die eine ohne Ermüdungserscheinungen ablaufende, schnellere und präzisere Ausführung der Fertigungsaufgaben versprechen. Allerdings beherrscht ein Roboter nur diejenigen Aufgaben, die ihm zuvor (etwa über explizite Programmierung) beigebracht wurden. In der Forschung werden Montageaufgaben für Roboter im Rahmen der automatischen Planung erzeugt, indem auf irgend eine Form von Wissens-Datenbasis zurückgegriffen wird. Diese Wissens-Datenbasis wird von einschlägigen Experten aufgefüllt und anschließend über entsprechende Algorithmen zur Ableitung adäquater Roboteraktionen angewandt. Darin können z. B. domänenspezifische Analyse- und

Berechnungsgrundlagen enthalten sein, die einen automatisch ableitbaren Rückschluss auf notwendige Fertigungsschritte des Bauteils erlauben. Oft werden bei der Planung Roboterfähigkeiten eingesetzt, die die vom Roboter ausführbaren Aktionen oder Prozesse meist in parametrisierbarer Form beschreiben. Björkelund et al. [20] verwenden zur Modellierung von Roboterfähigkeiten erstmals das PPR-Schema (Produkt, Prozess, Ressource). Eine Fähigkeit wird hierbei in Form von Zustandsmaschinen vorgeschlagen. Im EU-Projekt SkillPro wird dieser Ansatz um eine detaillierte Modellierung von Fähigkeiten (Skills) einzelner Automatisierungsanlagen erweitert [96]. Dabei werden auch komplexere Parameter wie Werkstückgröße und Materialtyp berücksichtigt.

In vielen Forschungsarbeiten werden Fähigkeiten als atomare und ggf. parametrisierbare „Black-Boxen“ angesehen, die im Rahmen der Planung als einzelne Aktionen berücksichtigt werden und erst in einer anderen Abstraktionsebene entsprechenden ausführbaren Programmcode für den oder die jeweiligen Roboter zur Verfügung stellen (vgl. [69, 97, 101, 131]). In einigen Ansätzen werden das Verhalten von Geräten sowie abstrakte Fertigungsprozesse mit SysML, einer auf UML basierenden, einheitlichen Modellierungssprache zur Beschreibung von Strukturen und Verhalten, betrachtet [36, 79, 108]. Die zuvor bereits erwähnte Taxonomie [59] beschreibt Basisfähigkeiten, die als atomare Aktionen bei der Planung berücksichtigt werden. Merdan et al. [84] schlagen die Beschreibung über eine Ontologie vor. Ein auf dem OPC UA-Standard basierendes Fähigkeitenmodell wird von Profanter et al. [98] vorgestellt. Einzelne Fähigkeitenmodule können dabei hierarchisch verschachtelt werden, sodass die Entwicklung weiterer Fähigkeiten unter Abstützung auf existierende (Basis-)Fähigkeiten möglich ist. Einen Cloud-basierten Ansatz verfolgen Waibel et al. [123] mit RoboEarth. Über eine zentrale Wissens-Datenbank werden die von den Robotern erfasste Daten (z. B. Sprachbefehle oder Bilddaten) analysiert und es werden entsprechend von den Robotern auszuführende Aktionen bereitgestellt. Im oben beschriebenen Projekt SMErobotics kommen Fähigkeiten in Form von Basisoperationen zum Einsatz, die zu komplexen Tasks kombiniert werden können. Die Autoren unterscheiden zudem zwischen universellen Fähigkeiten, die in unterschiedlichen Domänen wiederverwendbar sind, üblichen Fähigkeiten für bestimmte Domänen und spezifischen Fähigkeiten zur Fertigung einzelner Produkte. Andere Ansätze beschäftigen sich z. B. mit dem Erlernen von Fähigkeiten, wie etwa Fan et al. [34], die einen Roboter für ein zuverlässiges Greifen von Objekten trainieren oder Haarnoja et al. [50], die ihr Lern-Framework für Manipulationsaufgaben vorstellen.

Es existieren sehr viele Möglichkeiten, wie Fähigkeiten für Roboter modelliert werden können. Für eine effiziente Planung ist es weiterhin notwendig, dass die Modellierung der Fähigkeiten nicht nur ihre Ausführungssemantik kennt; vielmehr muss eine Fähigkeit für das Planungsmodell auch möglichst exakt die Effekte benennen können, die sie bei ihrer Anwendung in einer konkreten Situation auf das Bauteil und die Roboterzelle hat. In der vorliegenden Arbeit wird die Aufteilung nach PPRS (Product, Process, Resource, Skill) aufgegriffen und modifiziert, wobei eine Fähigkeit sowohl Ausführungssemantik als auch ihren situationsbedingten Effekt auf symbolischer Abstraktionsebene enthält.

Im Zuge der Planung können über Fähigkeiten hinaus auch weitere Informationen dazu beitragen, dass sich das Planungsergebnis verbessert und sich dadurch ggf. die Produktqualität signifikant erhöht, oder dass die Planungskomplexität reduziert wird

und das Finden einer Lösung deutlich weniger Zeit in Anspruch nimmt. Dazu werden in vielen Forschungsarbeiten u. a. Benutzerschnittstellen angeboten, damit der Anwender in der Modellierungs- aber auch in der Planungsphase sein Know-How beisteuern kann. Stenmark et al. [116] stellen eine grafische Benuzteroberfläche vor, die eine intuitive Programmierung von Roboteraktionen durch den Anwender ermöglicht. Die Aktionen können anschließend mehrfach wiederverwendet und, z. B. wenn sich die Position des zu bearbeitenden Objekts verändert, umparametrisiert werden. Ansätze, die HTNs zur Dekomposition von Tasks anwenden, setzen in jeder Hierarchieebene Dekompositionsregeln ein. Dabei bestimmen die vom Anwender bereitgestellten Dekompositionsregeln maßgeblich die Komplexität der Planung (vgl. [43]). Rodrıguez et al. [101] beziehen ein Regelset in die Planung ein, das die kombinatorischen Montagereihenfolgen einschränkt. Das oben erwähnte Projekt SMErobotics [33] zielt darauf ab, durch intuitiv zu bedienende Benutzerschnittstellen für vereinfachte Aufgabendefinitionen für Roboter die Einstiegshürde für die Automatisierung zu senken. Wildgrube et al. [126] stellen in diesem Rahmen eine graphische Bedienoberfläche vor, über die ein Anwender geometrische Constraints zur Definition eines Bauteils spezifizieren kann (vgl. [93, 95, 111]). Die Constraints werden automatisch zu Montageaufgaben ausgewertet, die anschließend über einen Roboter ausgeführt werden. Von Perzylo et al. [94] wird dieses Verfahren u. a. auf Fertigungsprozesse der Holzverarbeitung angewandt. Die Autoren beschreiben den Einsatz kognitiver Ansätze wie Objekterkennung, Objekt-Hervorhebungen durch Projektionen sowie Gestenerkennung zur Unterstützung des Anwenders bei der Aufgabendefinition für Roboter. Steinmetz et al. [113] stellen das Tool RAZOR vor, eine Mensch-Roboter-Schnittstelle zur visuellen Programmierung von Tasks. Dabei können Experten auf intuitive Weise Fähigkeiten modellieren, die von den Anwendern auf die konkret auszuführende Aufgabe parametrisiert und angewandt werden können. Das Tool integriert zudem das Konzept „Programming by Demonstration“ (PbD) [19], das es dem Anwender erlaubt, Roboterbewegungen durch Vormachen (z. B. über einen Sensor) oder intuitives Führen des Roboters aufzunehmen. Skoglund et al. [110] setzen PbD ein, um dem Anwender die Programmierung von Pick-and-Place-Aufgaben durch Vormachen über einen 6D-Sensor zu ermöglichen. Von Lu und Chen [77] wird ein Ansatz verfolgt, der es dem Anwender erlaubt, die vom Roboter zu erledigenden Aufgaben per Sprachbefehl mitzuteilen.

Die Einbeziehung von Expertenwissen in den Planungs- und Fertigungsprozess ist ein wichtiger Baustein für eine effiziente Automatisierung mit Robotern, insbesondere bei nur kleinen Losgrößen oder anspruchsvollen Prozessen. In der Forschung werden viele Arten der Einbeziehung von Know-How vorgeschlagen, die von der computergestützten Eingabe von Prozessdaten im Rahmen der Planung und Aufgabendefinition bis hin zur Interaktion zwischen Anwender und Roboter während der Ausführung reichen. Dabei werden die Eingabeelemente oft auf die Kernkompetenz des jeweiligen Experten zugeschnitten und intuitiv (z. B. grafisch) aufbereitet, sodass eine einfache, schnelle und präzise Aufgabendefinition selbst für schwierig zu automatisierende Prozesse möglich ist. In der vorliegenden Arbeit wird die constraintbasierte Definition zur Beschreibung des zu fertigenden Produkts aufgegriffen. Zur Planung der robotergestützten Fertigung stellt der Planungsansatz an unterschiedlichen Stellen entsprechende Schnittstellen für

die Einbeziehung von Expertenwissen bereit, die je nach Funktion eine große Bandbreite an Interaktionen mit dem Anwender während der Planung ermöglichen.

2.5 Simulation bei der Offline-Programmierung

Bei der Planung von Roboteranwendungen spielt die Offline-Programmierung eine besonders tragende Rolle, da die Effekte einer Roboteraktion nicht im echten Ablauf betrachtet werden müssen, sondern offline berechenbar sind. Hierbei kommen spezielle Simulationsumgebungen zum Einsatz, die eine Programmierung von Prozessen für neue herzustellende Produkte ermöglichen, ohne dabei die Roboteranlage zu belegen und die Fertigung anderer Produkte einstellen zu müssen. Damit jedoch die auf idealen Berechnungen basierenden Roboterprogramme auf eine reale Roboteranlage übertragen und dort ausgeführt werden können, ist eine möglichst nahe virtuelle Repräsentation der realen Roboterzelle notwendig. Als typische Vertreter von Offline-Simulationsumgebungen werden von Gan et al. [40] und Banks [13] DELMIA Robotics Offline Programming von Dassault Systèmes, Tecnomatix RoboCad von Siemens oder CimStation Robotics von AC&E genannt. Keines dieser Tools ist als universell einsetzbare Offline-Programmierumgebung ausgelegt, sondern unterstützt jeweils im Besonderen bei der Prozessdefinition, Optimierung und Simulation spezifischer Aufgaben wie Lackieren, Fräsen, Punkt- und Bahnschweißen. Eine Kooperation mehrerer Roboter bei der Ausführung solcher Aufgaben wird in diesen Programmen kaum betrachtet (vgl. [40]). Einfache Roboterkooperation durch Master-Slave-Beziehungen bzw. Synchronisationssignale sind hingegen z. B. über FastSuite von CENIT modellierbar. Aber auch Roboterhersteller bieten eigene Offline-Programmierumgebungen an, in denen spezielle Formen der Kooperation möglich sind und zudem spezifisches Wissen der jeweiligen Robotersteuerung z. B. zur Optimierung von Roboterbewegungen berücksichtigt werden kann. Beispiele sind KUKA.Sim von KUKA, Robotics Suite von Stäubli, Robot Studio von ABB oder Motoman Motosim von Yaskawa. Da diese auf der jeweiligen proprietären Roboterprogrammiersprache basieren, sind auch hier die Möglichkeiten der Kooperationen beschränkt. Zudem verfügen sie über keinen Support für eine automatische Planung automatisierter Fertigungsprozesse in verschiedenen Domänen. Auf dem Markt werden auch domänenspezifische Offline-Programmierumgebungen angeboten, z. B. Vericut von CGTech und FibreSim von Siemens, die auf besondere Fertigungsverfahren von carbonfaserverstärktem Kunststoff spezialisiert sind und im Besonderen in der Luft- und Raumfahrt Anwendung finden. Diese Programme eignen sich für die Definition der hochgradig spezifischen Produktionsprozesse, sie sind allerdings nicht erweiterbar, sind nicht auf andere Domänen übertragbar und integrieren keine Konzepte für Roboterkooperationen. Simulationsumgebungen, die häufig im akademischen Umfeld eingesetzt werden, sind CoppeliaSim [102] (Nachfolger von V-Rep [103]), MORSE [30], Gazebo [71] oder USARSim [22]. Als reine Simulationsprogramme, die weder Bahnplanung noch Roboterprogrammierung integrieren, bieten sie von sich aus keine Konzepte der automatischen Planung von Fertigungsprozessen.

Die aktuell existierenden Offline-Programmierumgebungen haben in der Regel die Einschränkung, dass sie nicht universell für eine Planung von Fertigungsprozessen

in unterschiedlichen Domänen ausgelegt sind und dahingehend keine entsprechenden Erweiterungsmöglichkeiten und Schnittstellen bieten. Zudem erzeugen sie aus den programmierten Roboterbahnen meist herstellerspezifischen Robotercode, wobei ursprüngliche Informationen, z. B. über Modelle der Bauteile und der Roboterzelle, verloren gehen. Durch Unterschiede zwischen virtueller und realer Anlage können bei der tatsächlichen Ausführung Abweichungen auftreten, die nicht mehr in das virtuelle Modell zurück übertragen werden können. Ein weiterer Nachteil ist, dass aufgrund der Codegenerierung lediglich diejenigen Kooperationsformen verfügbar sind, die über die Programmiersprache des jeweiligen Herstellers angeboten werden.

2.6 Fazit bisheriger Forschung und Wegweisung

In der Forschung existieren zahlreiche Ansätze, die das Know-How von Fachexperten in die Programmdefinition von Fertigungsanlagen einbeziehen und somit die automatische Planung von Automatisierungsaufgaben ermöglichen bzw. deren Effizienz und Qualität erhöhen. Bei allen Ansätzen, die eine globale Betrachtung der Montageplanung verfolgen, ist grundsätzlich immer derselbe limitierende Faktor zu erkennen: Bereits vergleichsweise kleine Planungsaufgaben mit wenigen zu berücksichtigenden Freiheitsgraden und wenigen Einzelteilen sind bereits nicht mehr effizient und nicht in adäquater Zeit lösbar, da die Ressourcen heutiger Computer für die kombinatorisch bedingte exponentielle Komplexität nicht ausreichen. Ein durchgängiger Planungsansatz, der aufgrund abstrahierender Konzepte und entsprechender Erweiterbarkeit universell für die Automatisierung von Montageaufgaben einsetzbar ist und gleichzeitig komplexe Aufgabenstellungen mit vielen Einzelteilen effizient lösen kann, wurde bisher nicht vorgestellt. Die Exponentialität wird vermutlich niemals gänzlich als beschränkender Faktor aus der automatischen Planung herauslösbar sein, doch können tragfähige Konzepte dazu beitragen, die Komplexität um einen exponentiellen Faktor zu reduzieren. Die vorliegende Arbeit adressiert nun diese Vision mit einem universell für die Montage einsetzbaren Planungsansatz, der das beherrschbare Komplexitätslevel für größere Montageaufgaben signifikant anhebt und dabei dank entsprechender Erweiterungskonzepte universell für unterschiedliche Domänen und Produktfertigungen einsetzbar ist. Dabei sind die vom Planungsansatz zu erfüllenden Eigenschaften im Einzelnen:

- **Durchgängige Offline-Planung** mit Konzepten zum Abgleich mit den real existierenden Gegebenheiten für präzise Montagepläne.
- **Einbeziehung von Expertenwissen** an unterschiedlichen Stellen und nach Zweck gekapselt (Trennung der Verantwortlichkeiten).
- **Modulares System** als Informationsbasis der Planung, bestehend aus **generisch anwendbaren Berechnungsmodulen** für eine automatische Planung und **Interaktionsmodulen** zur individuellen Erfassung von schwer abstrahierbarem Experten-Know-How.
- **Berücksichtigung von Roboterkooperationen** im Rahmen der Planung zur Entfaltung des Potentials flexibler Roboterzellen.

- **Parallelisierte Ausführungspläne** zur effizienten und ideal verteilten Ausführung von Montageprozessen mit einem dynamischen Wechsel zwischen Kooperationen und parallelem Arbeiten mehrerer Roboter.

Mit diesen Eigenschaften wird das Potential für die effektive Planbarkeit um Größenordnungen erhöht und es sind Montageaufgaben für Bauteile automatisch planbar, die aus erheblich mehr Einzelteilen bestehen.

Zusammenfassung. Der in dieser Arbeit vorgestellte Ansatz wird anhand von zwei Fallstudien auf die technische Umsetzbarkeit sowie seine Vorteile durch die automatische Programmierung evaluiert. Beide Fallstudien sowie deren Domänen werden in diesem Kapitel beschrieben. Um allgemeine Probleme und Herausforderungen der Montage zu adressieren, wird der Ansatz zum einen auf ein Fallbeispiel mit LEGO angewandt und damit eine Brücke aus Legosteinen geplant und errichtet. Zum anderen wird der Ansatz auf die Herstellung von Bauteilen aus kohlefaserverstärktem Kunststoff übertragen. Als konkretes Bauteil wird der Lagenaufbau einer Druckkalotte des Airbus A350 automatisiert gefertigt.

3 Fallstudien

Die Zunahme an Variabilität und zugleich die vermehrt fokussierte Individualisierbarkeit bei der Herstellung von Produkten sind ein stetig wachsender Trend vor allem auf dem Konsumentenmarkt. Dabei ist das Potential zur Automatisierung der Fertigungsabläufe heutzutage in vielen Branchen der Industrie gegeben. Doch der hohe Aufwand bei der Erstellung von Automatisierungslösungen lohnt sich bislang überwiegend nur bei Produktionen mit hohen Stückzahlen und vergleichbar geringer Variabilität. Bei Kleinteilserien und Produktionen mit hoher Variabilität hingegen ist eine Automatisierung aufgrund des hohen Initialaufwands durch lange und kostenintensive Entwicklungszeiten oftmals nicht lohnenswert. Damit eine Automatisierung der Fertigung gleichermaßen für kleinere Unternehmen mit Produktionen in eben solchem Ausmaß lukrativ und anwendbar wird, muss einerseits der Schritt zur Automatisierung finanziell tragbar für das Unternehmen sein, andererseits darf es bei der Umsetzung der Automatisierung nicht zu langen Ausfallzeiten der Ressourcen kommen, die für einen regulären Geschäftsbetrieb benötigt werden.

Das vorgestellte Konzept sieht dazu eine Fertigungsplanung mit maximal computergestützer Programmierung vor, die offline – also ohne Belegung einer realen Roboterzelle – auf Basis von räumlichen Modellen arbeitet. Deshalb kommt es gerade bei der

Fertigungsplanung variabler Bauteile nicht zu einem zwischenzeitlichen Stillstand in der produktiven Fertigungszelle. Um den Mehrwert wie auch die Übertragbarkeit des Ansatzes auf unterschiedliche Domänen zu evaluieren, werden zwei Fallstudien mit unterschiedlichen Montagebeispielen betrachtet. Jede Fallstudie birgt dabei jeweils eigene und teils domänenspezifische Herausforderungen, die im Rahmen des allgemeinen Ansatzes bewältigt werden.

3.1 Fallstudie 1: Montage von Strukturen mit LEGO

> *„Ein Junge und seine Großmutter haben in Bielefeld einen Zeitungsdieb mit einer aus Legosteinen gebastelten Alarmanlage gestellt“,*

so berichtete der Nachrichtensender n-tv [85]. Kinder wie auch Erwachsene kennen die bunten Kunststoffklötzchen, erfunden vom weltweit größten Spielzeughersteller, der LEGO System A/S. Die Steine folgen einerseits einer einfachen Struktur, andererseits können sie nahezu unbegrenzt zu beliebig komplexen Bauwerken kombiniert werden. Während kleine Kinder mit Legosteinen motorische und haptische Fähigkeiten erlernen, regt das Spielzeug bei älteren Kindern nicht nur ihre Phantasie und Kreativität an, sondern ermöglicht durch seine einfachen und doch mächtigen Steckmechanismen das erste Erkunden und Ausprobieren von Ingenieursdisziplinen wie der Architektur, Konstruktion und Mechanik.

Disziplinen wie diese spielen auch in der Industrie eine bedeutende Rolle. Bei der Automatisierung von Montageprozessen beispielsweise wirken Ingenieure verschiedener Fachrichtungen mit: Von der Planung des Bauteils durch den Konstrukteur, über die Festlegung von Fügeprozessen wie Kleben oder Schweißen durch den Prozessexperten, bis hin zum Automatisierungsexperten, der die einzelnen Fertigungsschritte in Roboterprogramme überführt. Sieht man das Zusammensetzen von Legosteinen als Fügeprozess, so stellt auch das Bauen von Legostrukturen ein Beispiel der klassischen Montage dar. Dank der komplexen Kombinierbarkeit ist LEGO ein geeigneter Stellvertreter, um den in dieser Arbeit vorgestellten Ansatz zur automatischen Programmierung auf den Anwendungsfall der Montage zu evaluieren. Für die Umsetzung wird dabei auf die Spielzeugvariante LEGO® DUPLO® gesetzt, die dank einer größeren Rasterung in einer realen Umsetzung mit Robotern eine einfachere Handhabung der Steine verspricht. Im Weiteren ist vereinfachend nur noch von LEGO die Rede. Nicht zuletzt ist LEGO auch deshalb eine geeignete Evaluationsdomäne, da sich systematische Schwierigkeiten der automatischen Programmierung sowie deren Lösungsansätze anhand von Beispielen mit Legosteinen leicht allgemeinverständlich motivieren und beschreiben lassen. Bereits in frühen Phasen dieser Arbeit haben sich Anwendungsbeispiele mit Legosteinen als geeignetes Medium zur Diskussion über allgemeine Planungsprobleme erwiesen.

3.1.1 Grundlagen

Der Grundstein von LEGO ist ein Quader mit rasterförmig angeordneten Noppen auf der Oberseite und entsprechenden passgenauen Aushöhlungen auf der Unterseite. Verschiedene Steine können so innerhalb des Rastermaßes und mit einer rechtwinkligen

Verdrehung aufeinander gesteckt werden. Die Seitenlänge eines Legosteins beträgt ein Vielfaches des Rastermaßes von 16 mm, wohingegen die Einheitshöhe eines Legosteins ohne Noppen 19,2 mm beträgt. Davon abweichend gibt es Plattenelemente mit halber Höhe, also 9,6 mm, sowie Basisplatten mit einer Höhe von 1 mm.

Neben der unterschiedlichen Form sind auch verschiedene Farben für jedes Legostein-Format möglich. Darüber hinaus gibt es noch weitere Legoteile wie Figuren, Tiere, Elemente für Häuser, Schienen und Züge sowie zahlreiche andere Sondersteine. Im Rahmen dieser Fallstudie werden jedoch überwiegend quaderförmige Legosteine verwendet, u. a. Legosteine mit dem Rastermaß 2×2 und 4×2 mit Einheitshöhe, sowie Legosteine mit dem Rastermaß 1×2 und doppelter Einheitshöhe. Für den Zusammenbau der Konstruktion wird eine Basisplatte mit dem Rastermaß 24×24 verwendet.

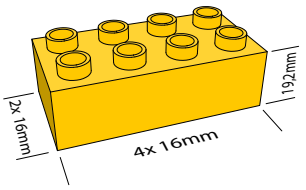


Abbildung 3.1. Legostein mit Seitenlängen von 4×2 Rastern zu je 16 mm.

Das Zusammenstecken zweier Legosteine erfordert keine besonders hohe Präzision, selbst mit einer gewissen Ungenauigkeit lassen sich die Steine noch in das Rastermaß zusammendrücken. Sind zwei Legosteine zusammengesteckt, hat ihre Position zueinander dagegen kein bis nur wenig Spiel – die Steine sitzen in der definierten Rasterung fest zusammen. Werden aber höhere Konstruktionen errichtet, so wird mit zunehmender Höhe ein gewisses Spiel bemerkbar. So kann ein hoher Legostein-Turm seitlich leicht gebogen sein. Ebenso ist bemerkbar, dass Konstruktionen mit seitlichen Ausladungen bei zunehmender Höhe und zunehmendem Gewicht instabil werden können. Insbesondere bei komplexeren Konstruktionen kommt somit der Statik eine bedeutende Rolle zu.

3.1.2 Stand der Technik

Für LEGO besteht grundsätzlich kein Bedarf an einer industriellen Montage mit Robotern. Die Planung der hierbei überwiegend auszuführenden Pick-and-Place-Aufgaben ähnelt allerdings in Teilen der der industriellen Montage. Daher wird im Folgenden der Stand der Technik bei der Automatisierung von Fertigungsprozessen beleuchtet.

Der Einsatz von Robotern ist in der produzierenden Industrie heutzutage weit verbreitet. Rund 2,4 Millionen Roboterinstallationen weltweit existierten im Jahr 2018, mit steigender Tendenz [62]. Dabei werden als größte Nutzer der Robotik die Branchen der Automobilindustrie, der Elektro- und Elektronikindustrie sowie der Metallindustrie und des Maschinenbaus genannt. Für diese Branchen spielt auch die Fertigung bei der industriellen Automatisierung eine wichtige Rolle. In großen Fertigungsstraßen arbeiten oft mehrere Roboter parallel an einem Werkstück, das eine Reihe an hintereinanderliegenden Montagezellen passiert, und führen an diesem einen speziellen Fertigungsschritt aus. Vom Setzen einer Schweißnaht über das Einbringen von Nieten bis hin zu Pick-and-Place-Aufgaben zum Anbringen neuer Teile an das Produkt sind die Aufgabenbereiche der Roboter sehr vielfältig. Die von den Robotern auszuführenden Prozesse sind in der Regel für den jeweiligen Anwendungsfall manuell programmiert und werden pro

Bauteil und in möglichst hohen Taktzyklen im Besonderen zur Produktion mit großen Stückzahlen ausgeführt.

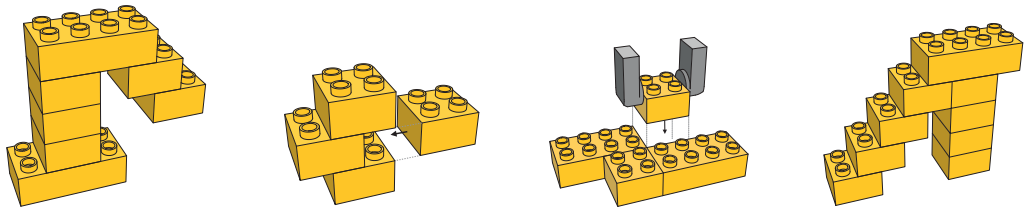
Ein anderer Ansatz, der aktuell besonders in der Kleinteilmontage zunehmend an Bedeutung gewinnt, nutzt die Programmierung von Robotern durch Vormachen (Programming by Demonstration) [8, 19]. Hierbei wird ein meist mittels Drehmomentsensoren sensitiver Roboter (z. B. KUKA LBR iiwa [72], Franka Emika Roboter [38], Rethink Robotics Sawyer [100]) vom Inbetriebnehmer direkt am Tool geführt; die demonstrierte Bewegung und der dabei ausgeführte Prozess werden mitprotokolliert. Anschließend kann der Roboter den aufgezeichneten Prozess in der Fertigung wiederholt anwenden. Im Vergleich zur konventionellen, manuellen Programmierung von Roboterprozessen ist das Einlernen durch Vormachen für bestimmte Montageschritte auf einfache und intuitive Weise möglich.

Die Firma Gigaset Communications GmbH geht bei der Produktion des derzeit einzigen deutschen Smartphones einen zukunftsweisenden Weg: Um mit den im internationalen Vergleich geringen Produktionskosten konkurrieren zu können, arbeiten im Montagewerk Bocholt Mitarbeiter und Montageroboter bei der Herstellung von Smartphones Hand in Hand [16]. Anders als in herkömmlichen automatisierten Fertigungen durchläuft das Smartphone die Montageschritte hier nicht etwa auf einem Förderband, sondern ein Mitarbeiter begleitet die gesamte Montage eines Smartphones. Dabei besucht er eine Vielzahl an Stationen, an denen das Gerät robotergestützt zusammengesetzt wird. Manche Roboter führen fest vorprogrammierte Aufgaben aus, andere erlauben z. B. die sensorgeführte Positionierung eines Schraubwerkzeugs durch den Mitarbeiter, das anschließend den Schraubvorgang automatisch ausführt, bis ein vorgegebenes Drehmoment erreicht wird.

Bei der Montage von Autos oder Smartphones ist die Stückzahl an herzustellenden Produkten entsprechend groß. Der Aufwand und die Investitionen, die in die Automatisierung des Montageprozesses mit Robotern zu investieren sind, zahlt sich sehr schnell aus: der Produktionsdurchsatz kann bei gleichzeitiger Senkung der Kosten pro Stück deutlich erhöht werden. Bei Produktionen mit kleinen Stückzahlen sieht es allerdings anders aus. Auch mit den unterstützenden Ansätzen zur Roboterprogrammierung steht der Aufwand für die Umstellung auf einen automatisierten Betrieb meist nicht im Verhältnis zur Zeit- und Kostenersparnis durch die anschließend automatisiert durchgeführte Fertigung. Die Automatisierung von Produktionen mit nur kleinen Stückzahlen ist daher oft nicht lohnend. Hier ist eine manuelle Fertigung noch üblich. Anhand eines Planungsproblems mit LEGO untersucht die Fallstudie 1 die Anwendbarkeit des vorgestellten Planungsansatzes $PaRTs$ auf ein klassisches Anwendungsbeispiel der Montage mit der Losgröße 1.

3.1.3 Problemstellung

LEGO ist in erster Linie für Kinder gemacht; die Planungskomplexität ist großteils einfacher Natur und stellt keine sonderlich große Herausforderung an den Baumeister dar. Und tatsächlich lassen sich die meisten Planungsaufgaben lösen, indem von unten nach oben gebaut wird, Ebene für Ebene, Stein für Stein. Dennoch können Situationen



(a) Überhang **(b)** Ausschluss **(c)** Kollision (zw. Stein und Greifer) **(d)** Statik

Abbildung 3.2. Besondere Konstellationen, die beim Zusammensetzen des Bauteils berücksichtigt werden müssen und somit besondere Anforderungen an eine Planungsstrategie setzen.

auftreten, die ein derart einfaches Vorgehen verhindern und deren Lösung nicht trivial ist. LEGO birgt einige Aspekte, die in bestimmten Fällen, bei bestimmten Bauteilen oder bei der Verwendung bestimmter Endeffektoren kein klassisches von unten nach oben Bauen erlauben und eine automatische Planung sowie die anschließende Fertigung mit Robotern teilweise in hohem Maße erschweren. Im vorgestellten Ansatz wird ein generisches Vorgehen verfolgt, das domänenunabhängig anwendbar ist und im Kern keine spezifischen Grundannahmen enthält, wie beispielsweise durch räumliche Anordnung vorgegebene Reihenfolgen. Solche Grundannahmen, die etwa dem Wissen eines Domänen- oder Automatisierungsexperten entspringen, sollen in einer generischen Strategie oder unter generischer Einbeziehung modularer Mechanismen berücksichtigt werden und lösbar sein. Im Fall von LEGO treten einige Sonderfälle bedingt durch seine Eigenschaften auf, die im Rahmen der allgemeinen Planungsstrategie zu berücksichtigen und zu lösen sind:

Überhang: Abbildung 3.2.a zeigt eine Konstruktion aus Legosteinen, die eine Art Vorsprung mit einer daran angebrachten Unterkonstruktion besitzt. Ein striktes von unten nach oben Bauen ergäbe in diesem Beispiel keinen vollständigen Plan, da die tiefer liegenden Steine der hängenden Unterkonstruktion erst gegen Ende angebracht werden können. Stattdessen ist eine Planungsstrategie erforderlich, die unabhängig von den Lageebenen der Legosteine ist.

Ausschluss: In Abbildung 3.2.b ist eine Situation dargestellt, in der ein Legostein in seine Zielposition nicht mehr eingesetzt werden kann. Die Noppen, die in dieser Situation ein Daraufsetzen von oben und gleichzeitig ein Ansetzen von unten erzwingen, führen zu einem unauflösbaren Widerspruch im erforderlichen Prozess. Sinnbildlich ist eine Fertigstellung des Bauteils ohne vorheriges Zurückbauen nicht mehr möglich. Bei der Planung ist in einem solchen Fall eine andere Reihenfolge der zu setzenden Steine zu wählen, die keinen Ausschluss erzeugt. Da anstelle des einen Steins auch größere zusammenhängende Baugruppen vom Ausschluss betroffen sein können, muss die Planung die Ursache eines Ausschlusses frühzeitig detektieren.

Kollision: Nicht nur die verwendeten Legosteine sind für die Planung der Einzelschritte ausschlaggebend, sondern auch das Werkzeug, mit dem der Stein gegriffen und gesetzt wird. Abbildung 3.2.c illustriert ein Beispiel, bei dem ein kollisionsfreies Absetzen des Steins durch den Greifer nicht möglich ist. Werden mit einer anderen Strategie zunächst

die kleinen Steine gesetzt, so können anschließend die großen Steine kollisionsfrei mit dem Greifer abgesetzt werden, da diese einen außenliegenden Greifpunkt besitzen. Bei der Strategie kann auch ein kollisionsfreier Anfahrtsweg von Roboter und Greifer eine Rolle spielen.

Statik: Werden Legosteine sukzessive zu einer aufsteigenden Treppe zusammengesetzt, so wird diese ab einer bestimmten Höhe bedingt durch ihr Übergewicht umkippen. Die in Abbildung 3.2.d skizzierte Treppe ist erst dann wieder stabil, wenn sich der oberste Stein der Treppe an den seitlichen Stützturm anlehnen kann. Stellt man sich eine manuelle Montage der Treppe vor, so müsste das Setzen von Steinen mit einer zweiten Hand unterstützt und die Treppe gehalten werden, bis die Gesamtkonstruktion einen stabilen Zustand erreicht. Bei der Planung muss dies mit geeigneten Reihenfolgen der zu setzenden Steine und ggf. unter Zuhilfenahme weiterer Aktuatoren zur Unterstützung der instabilen Zwischenzustände berücksichtigt werden.

Variationskomplexität: Der ausführbare Plan, der die Montage des Bauteils umsetzt, soll bevorzugt eine möglichst kurze Ausführungsdauer und eine optimale Roboterauslastung vorweisen. Um einen solchen optimalen Plan zu finden, können alle möglichen Abläufe der Fertigung eruiert und verglichen werden. Wie beispielsweise beim Schachspiel besteht allerdings auch beim Bauen von LEGO die Eigenschaft, dass die Anzahl an Möglichkeiten mit jedem weiteren Schritt exponentiell ansteigt. Stellt man sich einen einfachen Legoturm aus zehn übereinanderliegenden Steinen vor, wobei jeder Legostein auf acht unterschiedliche Weisen gesetzt werden kann (z. B. über zwei Greifer und jeweils vier möglichen Greifpunkte), so existieren insgesamt bereits 8^{10} – das ist mehr als eine Milliarde – Möglichkeiten, wie der Turm gebaut werden kann. Auch heute noch geraten Computer bei der Untersuchung derartig großer Zustandsräume mit ihrer Hardware an ihre Grenzen – eine vollständige Exploration ist bei etwas komplexeren Problemstellungen bereits nicht mehr möglich. Die Hauptherausforderung an eine erfolgreiche Planung ist es, die Suche dadurch zu optimieren, dass lediglich relevante Schritte exploriert und diejenigen Pfade außer Acht gelassen werden, die nicht zum gewünschten Ergebnis führen. Eine besondere Herausforderung bilden dabei Schritte, aus deren lokaler Sicht noch keinerlei Aussage über Erfolg oder Misserfolg getroffen werden kann.

3.1.4 Szenario: Errichten einer Brücke mit LEGO

Die Legokonstruktion, für die anhand des vorgestellten Planungsansatzes $PaRTs$ eine automatisierte Fertigung geplant und mit Robotern durchgeführt werden soll, ist in Abbildung 3.3 dargestellt: Eine Bogenbrücke bestehend aus insgesamt 64 Legosteinen in fünf unterschiedlichen Größen. Vier Pfeiler stützen die Fahrbahn auf einem Bogen ab, der eine Spannweite von ca. $29\,\mathrm{cm}$ besitzt.

Das Szenario wurde so gewählt, dass alle genannten Problemstellungen der Domäne LEGO darin auftreten. Wird die Brücke ausgehend von einer Seite aus errichtet, so können ab dem obersten Stein des Bogens die verbleibenden Steine der anderen Bogenseite von oben nach unten angefügt werden und somit als **Überhang** betrachtet werden. Gleiches ist aus Symmetriegründen der Brücke auch in die andere Richtung

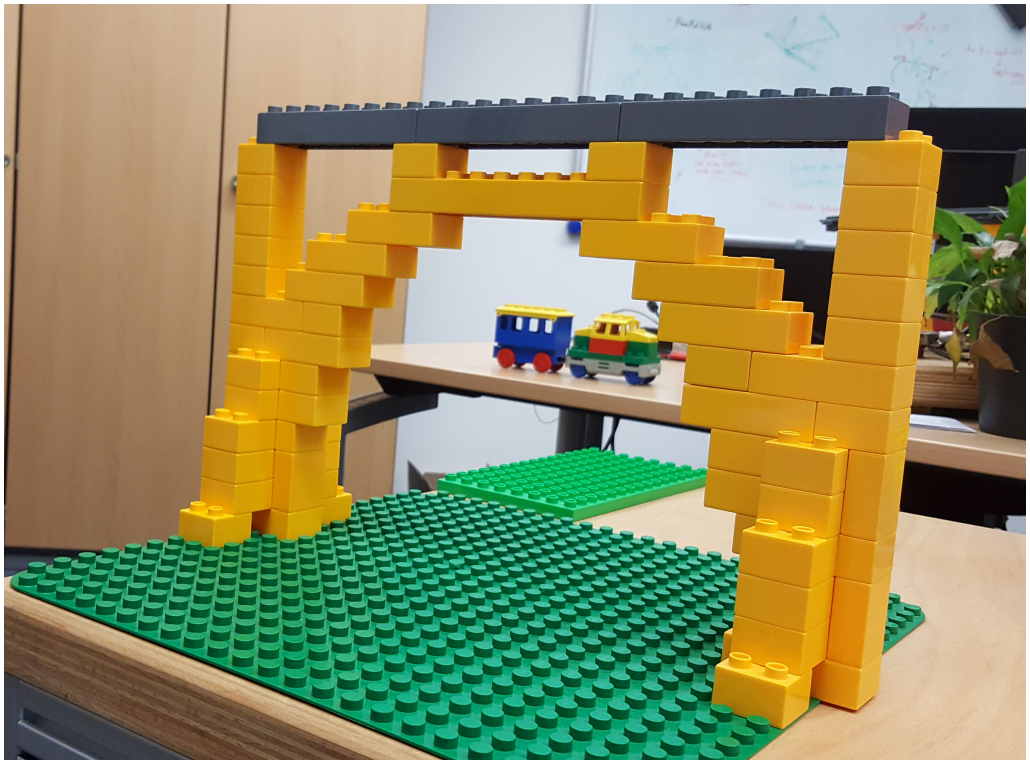


Abbildung 3.3. Brücke aus LEGO.

denkbar. In diesem Beispiel würde ein Überhang gleichzeitig auch einen **Ausschluss** ergeben, da zumindest der letzte Stein zwischen fast fertigem Bogen und Bodenplatte nicht mehr eingefügt werden kann. Weitere Fälle von Ausschlüssen können an mehreren Stellen in den Endstücken des Bogens (sog. Kämpfer) auftreten. Ebenfalls überwiegend in den Kämpfern besteht eine hohe Anzahl an möglichen Reihenfolgen der zu setzenden Legosteine, die zu **Kollisionen** zwischen Greifer und Legosteinen führen würden. Aber auch im oberen Bereich der Brücke bewirken bestimmte Reihenfolgen der zu setzenden Steine, dass ein zur Verfügung stehender Greifer nicht mehr kollisionsfrei agieren kann. Die Brücke als Ganzes ist eine selbsttragende Konstruktion, die darüber hinaus noch eine zusätzliche Last wie einen darüberfahrenden Zug tragen kann. Entnimmt man aber nur einen beliebigen Legostein des Bogens, so ist die selbsttragende **Statik** nicht mehr gegeben und die Brücke würde in sich zusammenfallen. Für den Bauvorgang der Brücke bedeutet dies, dass beide Brückenseiten ab einer gewissen Höhe instabil werden und ohne zusätzliche Unterstützung nicht stehen bleiben würden. Erst durch die „Hochzeit“ der beiden Bogenhälften mit dem obersten Verbindungsstück wird das gegenseitige Übergewicht aufgehoben. Die Statik ist auch ein Grund dafür, dass für eine erfolgreiche Montage nicht etwa ein Roboter ausreicht sondern mehrere Roboter eingesetzt werden müssen, die die Konstruktion während des Zusammenbaus geeignet unterstützen. Somit sind die Freiheitsgrade bei der Planung nicht nur durch die Reihenfolge der zu setzenden Steine sondern auch durch die Kombinationsmöglichkeiten der zur Verfügung stehenden Roboteraktionen bestimmt. Daraus ergibt sich eine enorm hohe **Variationskomplexi-**

tät für die Planung der LEGO-Brücke. Für eine effiziente und erfolgreiche Planung ist es notwendig, dem immens großen Zustandsraum an Möglichkeiten mit einer geeigneten Strategie zur zielgerichteten Suche nach gültigen Lösungen zu begegnen.

3.2 Fallstudie 2: Fertigung von Bauteilen aus carbonfaserverstärktem Kunststoff

Für die Produktion von Leichtbaukomponenten werden im Automobil- und Flugzeugbau Faserverbundmaterialien eingesetzt, die z. B. Carbon- oder Glasfasern mit anderen Materialien kombinieren. Aufgrund ihres geringen spezifischen Gewichts und ihrer mechanischen Belastbarkeit spielen dabei carbonfaserverstärkte Kunststoffe (CFK) eine besondere Rolle. Doch gerade im Flugzeugbau, der von der leichten Bauweise besonders profitiert, sind die zu produzierenden Stückzahlen gleicher Komponenten vergleichsweise nur gering. Eine produktspezifische Entwicklung von automatisierten Fertigungsanlagen ist unter diesen Umständen meist nicht rentabel. Dies ist mit ein Grund dafür, warum die Produktion von Leichtbaukomponenten auch heute noch in weiten Teilen von Handarbeit geprägt ist [3, 4]. Abhilfe schaffen können hier flexible Fertigungsanlagen, die solch komplexe Fertigungsprozesse für variierende Bauteile unterstützen. Um diese jedoch effizient einsetzen zu können, braucht es geeignete Konzepte zur Programmierung der Steuerungssoftware, sodass der Aufwand zur Umrüstung beherrschbar bleibt. Hier zeigt sich das Potential der automatischen Programmierung, über die sich der Aufwand zur Automatisierung in einer flexiblen Fertigungsanlage stark reduzieren lässt. Sie ermöglicht damit eine profitable Umsetzung der robotergestützten Fertigung von Kleinserienproduktionen im Umfeld von CFK.

3.2.1 Grundlagen

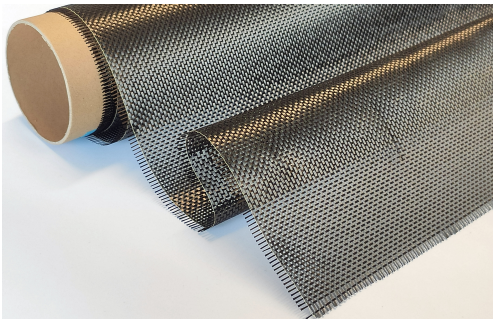
Die Fertigung eines Bauteils aus CFK ist besonders dann sinnvoll, wenn es sehr hohe Festigkeits- und Stabilitätseigenschaften aufweisen muss und gleichzeitig ein sehr geringes Gewicht erwünscht ist. Als Grundstoff dienen CFK einzelne Fasern aus Kohlenstoff, die in Anbetracht ihres geringen Gewichts eine sehr hohe Festigkeit entlang ihrer Faserrichtung aufweisen. Mehrere solcher Fasern werden üblicherweise zu Bündeln (sog. **Rovings**) zusammengefasst, die anschließend zur Herstellung von CFK-Bauteilen verwendet werden. Der Orientierung der Rovings kommt bei der Fertigung eine besondere Rolle zu, denn sie bestimmt die Belastbarkeit des fertigen Bauteils. Oftmals werden dafür Rovings in mehreren unterschiedlichen Orientierungen eingearbeitet.

Roving

Im Rahmen der Fertigung von CFK-Bauteilen mit hohen Belastungsanforderungen werden in der Industrie unterschiedliche Varianten verfolgt. Bei der **Wickel- und Tapelegetechnik** wird ein kontinuierlicher, oft mehrere Zentimeter breiter Strang an Kohlefasern auf die Form aufgebracht [91]. Bei dieser Methode ist die Faserorientierung eindeutig definiert, wodurch eine zuverlässige Stabilität des Bauteils erzielt werden kann. Eine andere Methode ist die dieser Fallstudie zugrundeliegende Fertigung mit textilen **CFK-Zuschnitten**. Hierbei werden zunächst Rovings zu textilen Geweben verarbeitet (siehe Abbildung 3.4a) und anschließend zu Zuschnitten mit einer gegebenen

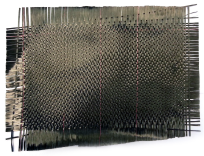
Tapelegen

Zuschnitt

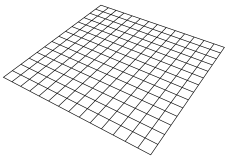


(a) Textil aus Carbonfaser-Gewebe

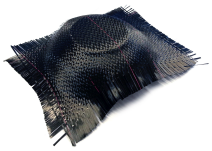
(b) Beispiel eines Lagenaufbaus aus CFK



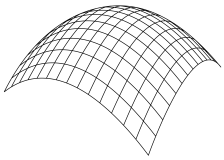
(c) Flach ausgelegtes CFK-Textil



(d) Symbolbild: Fasern eines CFK- Textils



(e) Auf 3D-Form drapiertes CFK-Textil



(f) Mögl. Verscherung der Fasern in 3D

Abbildung 3.4. Einzelne Carbonfaserstränge werden zu Textilmatten verwebt (a), aus denen Zuschnitte ausgeschnitten werden. Ein CFK-Bauteil besteht aus mehreren Lagen von Zuschnitten (b). Die Verscherungseigenschaft der Faserstränge erlaubt eine gewisse Angleichung des CFK-Textils an dreidimensionale Körper (c) bis (f).

Kontur und Faserorientierung zugeschnitten. Anschließend werden mehrere solcher Zuschnitte an bestimmte Positionen in einer Form und in mehreren übereinanderliegenden **Lagen** abgelegt. Dabei können die verwebten Rovings des CFK-Textils – anders als beispielsweise bei üblichen Textilien aus Stoff – derart gegeneinander verscheren, dass bis zu einem bestimmten Grad eine Angleichung des Textils an dreidimensional gekrümmte Oberflächen möglich ist, ohne dabei Falten zu werfen. Abbildung 3.4c zeigt ein Bild von einem kleinen, flach ausgelegtem CFK-Textil, dessen Faserstränge in Abbildung 3.4d symbolisch dargestellt sind. Die Faserstränge weisen an den mit anderen Strängen kreuzenden Stellen einen Winkel von jeweils 90° auf. Wird das Textil auf eine dreidimensional gekrümmte Form aufgebracht, so ist dank der Verscherungseigenschaft, die eine Winkeländerung zwischen den kreuzenden Fasersträngen erlaubt (exemplarisch dargestellt in Abbildung 3.4f), ein passgenaues Angleichen des Textils an die Form möglich (siehe Abbildung 3.4e). Das passgenaue Applizieren von CFK-Zuschnitten in oder auf eine Form wird **Drapieren** genannt.

Lage

Drapieren

Im Falle von textilen Zuschnitten erfolgt die Konstruktion und Auslegung eines CFK-Bauteils in spezialisierten CAD-Programmen durch einen Experten, der die Statik und Belastungstoleranzen des Bauteils berücksichtigt. Das Ergebnis ist eine Beschreibung des Bauteils, das sogenannte **Plybook**, welches neben Materialdaten den Aufbau des Bauteils, d. h. die Lage und Form textiler Zuschnitte im verformten Zustand, aber auch die Form der Zuschnitte im ebenen Zustand enthält. So werden z. B. große rechteckige Zuschnitte verwendet, um die Struktur eines Flugzeugrumpfs zu bauen, während klei-

Plybook

nere, speziell geformte Zuschnitte dazu dienen, Verstärkungen etwa an den Rändern von Fensteröffnungen anzubringen.

RTM

Bei der Herstellung im **RTM-Verfahren** (Resin Transfer Moulding) [10, 58] werden textile Zuschnitte in Form eines vorkonfektionierten **Preforms** in eine zweiteilige Werkzeugform abgelegt und darin in die beabsichtigte Zielform gepresst. Unter Druck wird nun Harz in das Bauteil injiziert, und die Fasern werden unter Verdrängung jeglicher Luft damit getränkt. Nach der Aushärtung des Harzes wird das Bauteil aus dem Presswerkzeug entnommen.

Preform

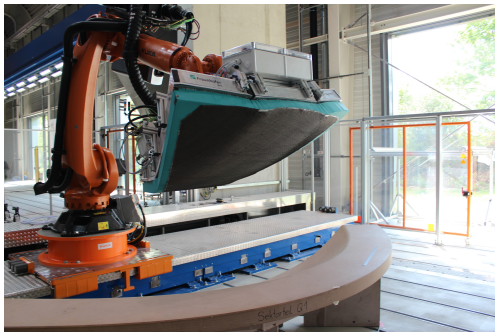
VARI

Bei der Herstellung im **VARI-Verfahren** (Vacuum Assisted Resin Injection) [12, 35] dagegen wird das Bauteil zunächst – in der Industrie aktuell überwiegend in einem manuellen Verfahren – aus trockenen Carbonfasergeweben geformt. Dazu werden die textilen Zuschnitte schichtweise in die Form drapiert und mehrere derartige Lagen übereinander aufgebaut (vgl. Abbildung 3.4b). Nachdem alle Zuschnitte drapiert sind, wird eine luftdichte Folie aufgebracht und unter Vakuum Kunstharz in das Bauteil injiziert. Anschließend wird das Bauteil in einem Ofen ausgehärtet.

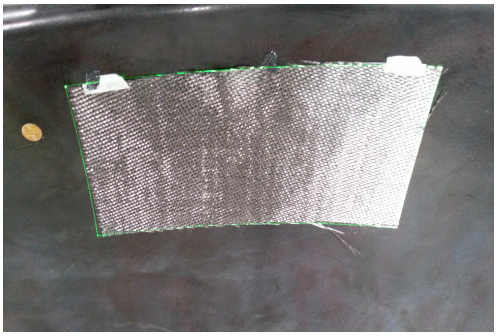
3.2.2 Stand der Technik

Im Bereich der textilen CFK-Verarbeitung produziert der Automobilhersteller BMW in seinen Werken Landshut und Leipzig mittlerweile automatisiert und in Serie Karosserieteile aus CFK, unter anderem für Modellreihen der BMW i Fahrzeuge [14, 25]. Dabei werden sogenannte Stacks – Stapel von Zuschnitten – über das RTM-Pressverfahren in Form gebracht. Diese Verarbeitung ist aufgrund des Textilverhaltens von CFK-Zuschnitten jedoch nicht für alle Bauteilgrößen und Geometrien, im besonderen für komplexere zweifach gekrümmte Bauteile, gleichermaßen geeignet. Hier kann stattdessen auf die mit textilen Zuschnitten arbeitende Drapierung unter Anwendung des VARI-Verfahrens zurückgegriffen werden. Für das automatisierte Erstellen eines CFK-Bauteils wird hierbei in der Forschung oft eine Pick-and-Place-Vorgehensweise verfolgt, bei der textile Zuschnitte aus Carbonfasern über ein Lagersystem bereitgestellt und von einem oder mehreren Robotern mit speziell entwickelten Endeffektoren auf eine gekrümmte Bauteilform appliziert werden [4, 44]. Abbildung 3.5 zeigt einen am Fraunhofer IWU-RMV entwickelten Greifer zur Drapierung großformatiger Zuschnitte in doppelt gekrümmte Formen. Die Aufnahmefläche bildet ein spezieller Schaumstoff, der einen in mehreren Kammern des Greifers erzeugten Luftsog an die Außenfläche weitergibt und so das Textil aufnimmt. Durch die Krümmung des Greifers ist bei der Aufnahme eines Zuschnitts – beispielsweise von einem Tisch oder einem Magazin – eine Rollbewegung des Greifers mit jeweils zum richtigen Zeitpunkt stattfindenden Aktivierungen der Vakuumkammern notwendig. Der Greifer ist mechanisch so gestaltet, dass er bei der Drapierung in eine doppelt gekrümmte Form zwei Seitenteile anklappen und so die gewünschte Verformung in den Zuschnitt einbringen kann.

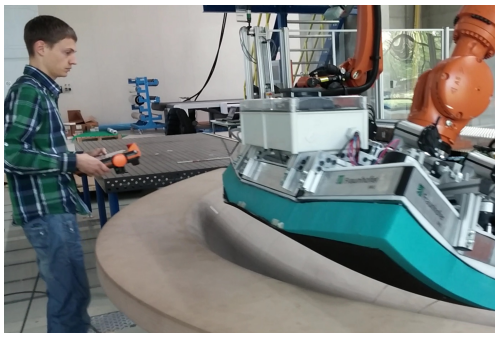
Die manuelle Programmierung von Roboter und Greifer für die Drapierung von Zuschnitten ist oft aufwendig und nimmt meist mehrere Stunden pro einzelnem Zuschnitt in Anspruch [86]. Der Inbetriebnehmer bewegt dazu Roboter und Greifer manuell mit dem Bedienpanel des Roboters, um einen Zuschnitt aufzunehmen, über die Greiferak-



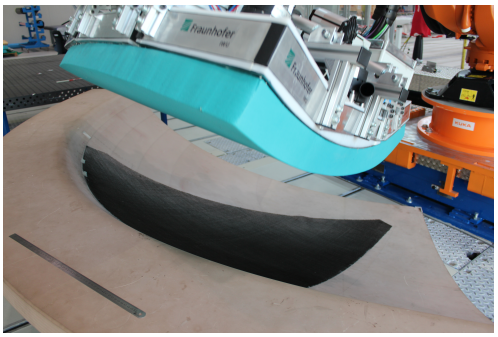
(a) Spezialgreifer für CFK-Textilien



(b) Laserprojektion (grün) in der Form



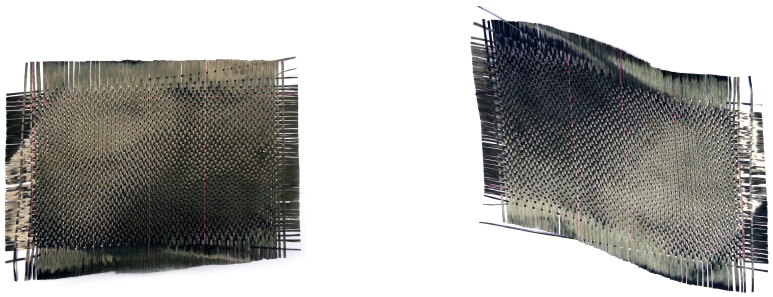
(c) Manuelles Teachen des Programms



(d) Der in die Form drapierte Zuschnitt

Abbildung 3.5. Die Zuschnitte werden von einzelnen Robotern bzw. Roboterteams mit deren Endeffektoren aufgenommen, verformt und anschließend in eine Form drapiert.

tuatorik die Verformung einzubringen und ihn in der Form abzulegen. Anschließend wird die Position des drapierten Zuschnitts mit einer hochpräzisen Laserprojektion der Zielkontur verglichen und daraus die Ablagequalität des Zuschnitts bewertet. Oftmals verdecken bei der Ablage Greifer und Roboter die Laserprojektion, sodass ein präzises Anfahren der Ablageposition nicht ohne weiteres möglich ist. Ebenso kann die gewählte Aufnahmeposition des Zuschnitts auf dem Greifer die Ablagequalität beeinflussen, wenn die Greifergeometrie an der Ablageposition nicht zur Geometrie der Form passt. Da bei der Verformung des Greifers der Zuschnitt an bestimmten Stellen auf der Greiffläche gleitet, ist die in den Zuschnitt eingebrachte Verformung irreversibel. D. h. bei einer Zurückformung des Greifers würde der Zuschnitt nicht mehr identisch wie zu Beginn auf dem Greifer liegen und ggf. sogar Falten werfen. Aus diesem Grund muss der gesamte Vorgang des Roboter-Teachens oft mehrmals von vorn wiederholt werden, bevor eine ausreichende Ablagegenauigkeit erreicht ist. Entspricht diese den Vorgaben, werden die vom Inbetriebnehmer eingestellten Parameter für Roboter und Greifer in ein Roboterprogramm überführt, das ein wiederholtes Drapieren des entsprechenden Zuschnitts erlaubt. Dieses Vorgehen ist für jeden Zuschnitt des Bauteils in analoger Weise einzeln zu wiederholen. Für ein vollständiges Bauteil sind jedoch viele Zuschnitte erforderlich, bei Flugzeugrümpfen unter Umständen mehrere hundert, wobei Form und Größe der verschiedenen Zuschnitte eines Bauteils stark schwanken können.



(a) CFK-Textil in der Grundform **(b)** Seitlich verzogenes CFK-Textil

Abbildung 3.6. Die Verscherungseigenschaft der Fasern von CFK-Textilien ermöglicht eine Verformung von Zuschnitten mit vielen Freiheitsgraden.

3.2.3 Problemstellung

In Fallstudie 2 wird die Fertigung von CFK-Bauteilen mit formlabilen, textilen CFK-Zuschnitten im VARI-Verfahren betrachtet. Dabei liegt der Fokus auf dem Drapieren der Zuschnitte in die Form zum Herstellen des Lagenaufbaus. Das in Abbildung 3.4 exemplarisch dargestellte Verscherungsverhalten von CFK-Textilien ermöglicht zwar die Fertigung von dreidimensionalen Produktformen, doch stellt das formgenaue Drapieren eine große Herausforderung dar. Denn die Verscherung ermöglicht ebenso ein Verziehen des Zuschnitts (siehe Abbildung 3.6), sodass gerade größere Zuschnitte das exakte Treffen ihrer Zielkontur in der Form erschweren oder sogar leicht von den intendierten Faserverläufen abweichen. Im Bereich der Luft- und Raumfahrt sind zudem oftmals hohe Anforderungen an die Präzision bei der Fertigung gegeben. Doch das formlabile Textilverhalten macht ein exaktes Drapieren mit einer geforderten Genauigkeit von unter einem Millimeter zu einer aufwendigen Aufgabe, sodass diese Art der CFK-Verarbeitung meist manuell erfolgt.

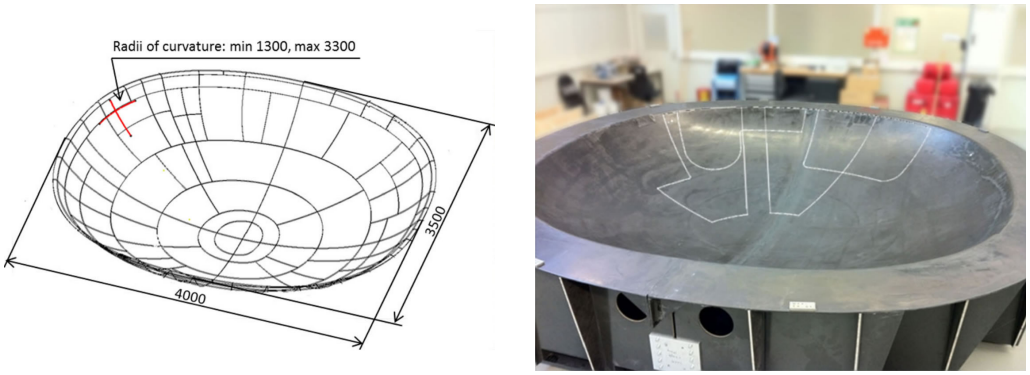
Um CFK-Zuschnitte bei geometrisch komplexen Bauteilen mit Robotern automatisiert zu drapieren, werden hochgradig spezialisierte und komplexe Endeffektoren eingesetzt, die oftmals eigens für das zu fertigende Bauteil entwickelt werden. Diese verfügen über Aktuatorikelemente (z. B. Saugmodule), um den Zuschnitt aufzunehmen, und eine Mechanik, die die Verformung geeignet in den Zuschnitt einbringt. Die möglichen Einstellungen der Greiferparameter können dabei sehr komplex sein: Beispielsweise kann über die Bestimmung, welche Vakuumsektoren aktiviert werden, ein störender Luftstrom an den vom Zuschnitt nicht bedeckten Vakuumsektoren reduziert werden. Über die Einstellung des Unterdrucks in den Vakuumsektoren kann Einfluss darauf genommen werden, an welchen Stellen der Zuschnitt bei der Verformung besonders am Greifer haftet und an welchen Stellen ein Gleiten möglich ist. Auch die Verformung des Greifers selbst ist für jeden Zuschnitt geeignet zu wählen, sodass dieser passgenau in die Form appliziert werden kann.

Diese Vielzahl an Einstellmöglichkeiten macht die Programmierung einer robotergestützten Fertigung sehr aufwendig. Dazu kommen die bereits erwähnten Herausforderungen beim Teaching-basierten Programmieren, wonach der Greifer die in der Form angezeigte

Zielkontur beim Ermitteln der Ablageposition verdeckt (siehe Abschnitt 3.2.2). Alle diese Schwierigkeiten und die gleichzeitig nur geringen Stückzahlen desselben Produkttyps in der Luft- und Raumfahrt stellen eine große Herausforderung für die Automatisierung derartiger Fertigungsprozesse dar.

Die automatische Offline-Planung solcher Prozesse auf Basis geometrischer Analysen und Simulationen der Drapierprozesse scheint hier besonders vielversprechend. Zudem könnte davon profitiert werden, dass die durch das Teaching-Vorgehen verursachten langen Inbetriebnahmezeiten entfallen, in denen die Roboterzelle nicht für die Fertigung z. B. anderer Produkte einsetzbar ist. Gleiches gilt für die Beschäftigungszeit des benötigten Fachpersonals, also die der Inbetriebnehmer und Ingenieure. Doch es ergeben sich eine Reihe an Problemstellungen, die für eine möglichst vollautomatische Ableitung geeigneter Drapierprozesse berücksichtigt und gelöst werden müssen:

1. **Auswertung des Plybooks:** In diesem Dokument, das der Ingenieur in einem CAD-Programm erstellt, sind wichtige Informationen für die Herstellung des CFK-Bauteils enthalten. Problemstellung 1 adressiert die Art und Weise, wie die Daten des Plybooks zu interpretieren sind, sowie die Frage, wie diese für den Planungsansatz aufbereitet und verfügbar gemacht werden können.
2. **Bereitstellen von Prozessparametern:** Damit die aufwendige, manuelle Programmierung durch den Inbetriebnehmer durch eine vom Computer errechnete Programmerstellung ersetzt werden kann, muss der Computer in die Lage versetzt werden, auf all das zur Fertigung notwendige Wissen von Inbetriebnehmer und Ingenieur zurückgreifen und dies anwenden zu können. Problemstellung 2 formuliert die Frage, wie Inbetriebnehmer und Ingenieur ihr Wissen zur Planung beisteuern können, damit die korrekte Roboter- und Greiferansteuerung (technische Sicht) und die Wahl der richtigen Prozessparameter bei der Verarbeitung von CFK-Zuschnitten (Domänensicht) automatisiert erfolgen kann.
3. **Austauschbarkeit des Endeffektors:** Die verschiedenen Zuschnitte eines CFK-Bauteils variieren oft in Form und Größe. Für eine präzise Drapierung aller Zuschnitte kann es sinnvoll sein, unterschiedliche, auf die jeweiligen Eigenschaften der Zuschnitte abgestimmte Endeffektoren einzusetzen. Problemstellung 3 fordert eine Untersuchung, inwieweit der konkret einzusetzende Endeffektor für die Planung austauschbar bzw. durch weitere Endeffektoren ergänzbar ist, und mit welchem Aufwand ein solcher Wechsel (in Kombination mit Problemstellung 2) verbunden ist.
4. **Übergang zur realen Fertigung:** Anhand der Beschreibung der Roboterzelle, den zur Verfügung stehenden Fähigkeiten und zusätzlichen Informationen (so wie denen aus dem Plybook) soll automatisiert ein geeignetes Roboterprogramm zur Fertigung eines gegebenen CFK-Bauteils bestimmt werden. Eine zentrale Frage hierbei ist, inwieweit ein offline geplantes Roboterprogramm unmittelbar zur Fertigung in einer realen Roboterzelle geeignet ist, bzw. welche Mechanismen notwendig sind, um ein Programm „realtauglich“ zu machen. Problemstellung 4 benennt die Herausforderungen, die bei der Online-Ausführung von offline geplanten Roboterprogrammen zu lösen sind, um ein CFK-Bauteil real zu fertigen.



(a) Techn. Darstellung der Druckkalotte [44]

(b) Form zur Herstellung der Druckkalotte [44]

Abbildung 3.7. Eine Druckkalotte (a) bildet im Heck eines Flugzeug das Abschlusselement, das den Innendruck des Flugzeuginnenraums aufrechterhält. Eine spezielle Form (b) wird zur Herstellung der Druckkalotte aus CFK verwendet.

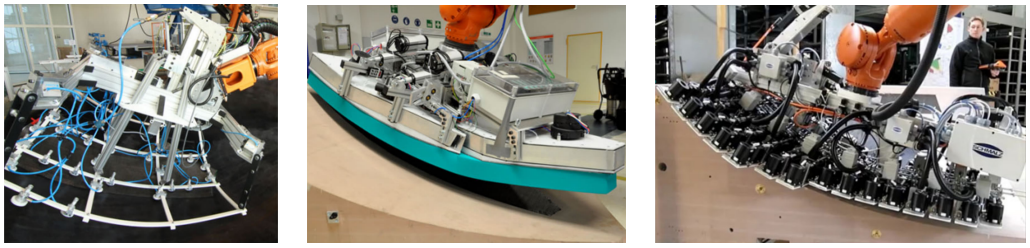
5. **Präzision:** Dem vom Computer automatisiert geplanten Programm zur Ansteuerung der Aktuatorik liegen präzise (geometrische) Daten zugrunde, die eine Ausführung der Fertigungsprozesse in einer idealen und toleranzfreien Welt beschreiben. In einer realen Welt kommen allerdings Toleranzen und Ungenauigkeiten hinzu, die das Ergebnis der Fertigung maßgeblich beeinflussen können. Problemstellung 5 adressiert die Frage, ob bei einer Offline-Planung die vorgegebene Genauigkeit bei der Herstellung von CFK-Bauteilen erlangt werden kann.

3.2.4 Szenario: Fertigung eines Lagenaufbaus für doppelt gekrümmte Bauteile

Besonders die Luft- und Raumfahrt profitiert von den Vorteilen, die Kohlefaser-Verbundwerkstoffe dank ihrer hohen Belastbarkeit und ihres niedrigen spezifischen Gewichts bieten. So wird heutzutage auch bei der Fertigung von Flugzeugen zunehmend auf Carbonteile gesetzt, die die schwereren Stahlelemente ersetzen. Während das Flugzeug Airbus A300 im Jahr 1970 lediglich $7\,\%$ an verbauten CFK-Elementen enthielt, besteht heute z. B. das Modell Airbus A350 XWB und auch das Flugzeug Boeing 787 zu mehr als $50\,\%$ aus CFK [49]. Die einzelnen Bauteile werden meist so konzipiert, dass das Flugzeug eine stabile aber auch effiziente Flugweise erhält. Viele der Bauteile haben daher eine räumlich komplexe, aerodynamische Form. So auch die **Druckkalotte** des Airbus A350: Das zweifach gekrümmte Abschlusselement im Heck des Flugzeugs sorgt für die Aufrechterhaltung des Luftdrucks im Passagierbereich, während beim Flug in großer Höhe der Außendruck abfällt. Abbildung 3.7a zeigt die asymmetrische Form der Druckkalotte, die einen Durchmesser von $3{,}5$ bis 4 Metern besitzt. Die Herstellung im VARI-Verfahren geschieht über die Drapierung von CFK-Zuschnitten in eine speziell dafür angefertigte Form (siehe Abbildung 3.7b).

Druckkalotte

Im Jahr 2012 startete ein Forschungsprojekt mit mehreren Beteiligten aus Industrie und Forschung, darunter u. a. die Premium Aerotec GmbH, das Zentrum für Leichtbauproduktion des Deutschen Zentrums für Luft- und Raumfahrt e. V. (DLR-ZLP) und das Institut



(a) Netzgreifer [44] **(b)** Schaumstoffgreifer [44] **(c)** Flächengreifer [44]

Abbildung 3.8. Drei Greifer mit unterschiedlichen Prinzipien zur Drapierung von CFK-Textilien wurden im Projekt AZIMUT evaluiert.

für Software & Systems Engineering (ISSE) der Universität Augsburg. Das Projekt mit dem Namen **AZIMUT** („Automatisierung zukunftsweisender industrieller Methoden und Technologien für CFK Rümpfe“) hatte in einer Fallstudie zum Ziel, die Fertigungskette einer Druckkalotte des Airbus A350 aus CFK durchgängig zu automatisieren [44, 81]. Für die Erstellung des Preforms – das ist der mit den Zuschnitten zusammengesetzte Lagenaufbau in der Form – wurden drei unterschiedliche Spezial-Endeffektoren entwickelt, die im Besonderen für die Zuschnittsdrapierung in eine zweifach gekrümmte Form geeignet sind. Der vom DLR entwickelte **Netzgreifer** (siehe Abbildung 3.8a) bringt die Verformung über ein biegbares Gitter an Glasfaserstäben, an denen einzelne Saugmodule angebracht sind, schonend in den Zuschnitt ein. Über mehrere Vakuumkammern verfügt der am Fraunhofer IWU-RMV entwickelte **Schaumstoffgreifer** (siehe Abbildung 3.8b), der den Zuschnitt über einen Luftsog am Schaumstoff anheftet und ihn über zwei anklappbare Seitenflügel in die dreidimensionale Zielkontur verformt. Aufgrund der abgerundeten Form des Greifers ist eine rollende Aufnahme des Zuschnitts erforderlich. In Abbildung 3.8c ist der von der Firma J. Schmalz GmbH entwickelte **Flächengreifer** abgebildet, der dank eines verformbaren Rückgrats und 15 daran angebrachten, wölbbaren Rippenbögen seine zahlreichen Saugflächen präzise an eine gegebene Zielform angleichen kann.

AZIMUT

Netzgreifer

Schaumstoffgreifer

Flächengreifer

Mit einer ersten, rudimentären Variante des in dieser Arbeit vorgestellten Planungsansatzes $PaRTs$ wurde eine Offline-Programmierplattform (**CFK-OPP** [78, 86]) entwickelt, die eine semi-automatische Planung des Fertigungsprozesses mit jedem der drei Greifer ermöglicht. Geometrische Berechnungen erfolgen automatisch; Prozessparameter, die nicht automatisiert abgeleitet werden können, werden interaktiv von den Domänenexperten abgefragt. Drei unterschiedliche Zuschnitte des CFK-Bauteils wurden ausgesucht, um anhand derer den Planungsansatz zu evaluieren. Für jeden der drei Zuschnitte und jeden Greifer wurde jeweils ein Roboterprogramm geplant und in einer realen Roboterzelle zur Ausführung gebracht. Dabei wurde zu jeder Ausführung der Zeitaufwand für die interaktive Planung sowie die Präzision bei der Ablage des Zuschnitts in der Kalottenform erfasst.

CFK-OPP

Die Fallstudie zur CFK-Bauteilerstellung in dieser Arbeit beschäftigt sich ebenso mit der Planung von Fertigungsprozessen zur Drapierung textiler CFK-Zuschnitte. Es wird der vorgestellte Planungsansatz $PaRTs$ eingesetzt, um das im AZIMUT-Projekt postulierte

Vorhaben der Preformerstellung mit einem generischen und noch weiter automatisierten Ansatz umzusetzen. Es werden dabei dieselben Rahmenbedingungen aufgegriffen, die für die CFK-OPP im Rahmen von AZIMUT galten. Dadurch sind die erzielten Ergebnisse mit denen aus AZIMUT vergleichbar. Sind schlussendlich die mit $PaRTs$ erzeugten Roboterprogramme mit denen der CFK-OPP identisch, so gelten die Evaluationsergebnisse aus AZIMUT auch für die Lösung über den Planungsansatz $PaRTs$.

Während im Rahmen von AZIMUT das Drapieren der Zuschnitte in einzelnen, isoliert betrachteten Szenarien ausgeführt wurde und die Spezialgreifer nicht miteinander kombiniert wurden, soll in der Fallstudie 2 mit $PaRTs$ die gesamte Prozesskette aller drei Zuschnitte in einem Gesamtszenario betrachtet werden. Ebenso wird der Paralleleinsatz zweier unterschiedlicher Greifer beim Erstellen des Lagenaufbaus mit den drei Zuschnitten untersucht.

Zusammenfassung. Automatische Planung kann besonders dann von Bedeutung werden, wenn eine manuelle Programmierung der Roboterzelle aufgrund zu hoher Variabilität der Produkte nicht mehr möglich ist. Der vorgestellte Planungsansatz $PaRTs$ gliedert sich in verschiedene Phasen, die auf verschiedenen Abstraktionsebenen und unter Zuhilfenahme von Expertenwissen aus unterschiedlichen Bereichen zur Bestimmung eines finalen Programms beitragen. Auch die Verwendung mehrerer Roboter und deren mögliche Interaktionen im Rahmen der Fertigung ist im Planungsansatz berücksichtigt.

4

Allgemeiner Planungsansatz

Die stetig steigende Nachfrage nach individualisierten und gleichzeitig kurzlebigeren Produkten auf dem Konsumentenmarkt zeigt, welche Bedeutung einer variablen und auf kleine Stückzahlen ausgerichteten Produktion zukommt. Um die industrielle Produktion unter diesem Maßstab weiterhin erfolgreich bewältigen zu können, ist ebenso ein Paradigmenwechsel im Hinblick auf die Programmierung der fabrizierenden Roboteranlagen notwendig. Wird bei der klassischen Serienproduktion die Programmierung der Roboteranlage meist durch Automatisierungsexperten manuell und explizit für das zu fertigende Produkt angefertigt, ist diese Vorgehensweise bei kleinen Losgrößen nicht mehr rentabel. Es sind verstärkt Mechanismen gefragt, die eine automatische Programmierung der Roboterzelle erlauben und somit einen schnellen Wechsel des zu fertigenden Produkttyps in der laufenden Produktion ermöglichen.

Der in dieser Arbeit vorgestellte Planungsansatz $PaRTs$ ist so konzipiert, dass er auf Basis von Fähigkeiten der Roboterzelle und zusätzlichen Informationen über das Bauteil und die Domäne automatisch Fertigungsabläufe bestimmen kann. Die Planung läuft dabei in mehreren verschieden Phasen auf unterschiedlichen Abstraktionsebenen ab, um ein finales Fertigungsprogramm zu bestimmen. Die Phasen sowie deren jeweilige Einbeziehung von Expertenwissen werden in Abschnitt 4.1 genauer erläutert. In der Produktion ist es oftmals nützlich oder gar notwendig, dass mehrere Roboter in einem Fertigungsschritt zusammenarbeiten. Abschnitt 4.2 beschreibt verschiedene Formen der Kooperation zwischen mehreren Robotern und stellt dar, wie diese in der Konzeption des Planungsansatzes berücksichtigt sind.

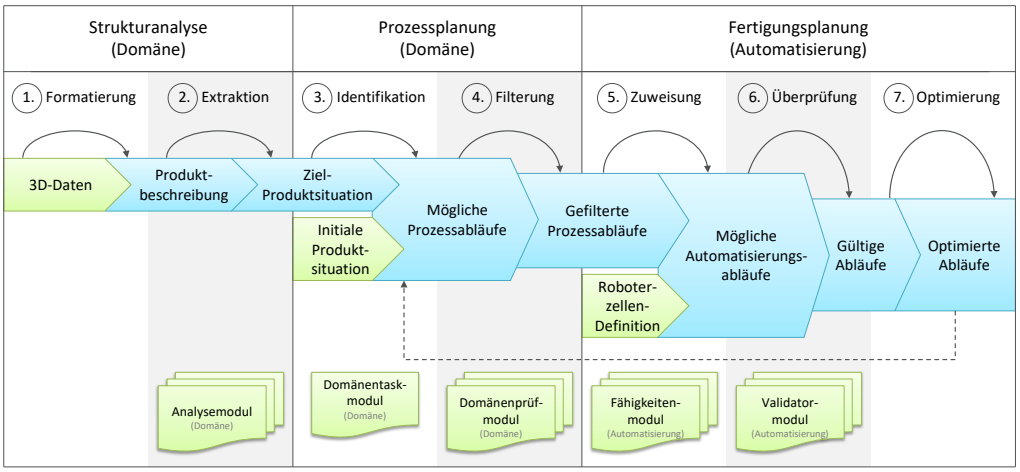


Abbildung 4.1. Überblick über den mehrphasigen Planungsansatz $PaRTs$.

4.1 Mehrphasige Planung

Bei der automatischen Planung spielen viele Faktoren eine Rolle, wie etwa die Eingabedaten, die das zu fertigende Produkt beschreiben, die Beschreibung der Roboterzelle, in der die Montage stattfindet, sowie die Definition der Domäne und der darin beschriebenen Prozesse zur Umsetzung einzelner Fertigungsschritte. Aber auch die Gestaltung der Planung selbst, die dabei verwendeten Datenstrukturen und entstehenden Artefakte, sowie die an unterschiedlichen Stellen verwendete Einbeziehung von Expertenwissen, sind prägende Merkmale des vorgestellten Planungsansatzes $PaRTs$.

Abbildung 4.1 stellt einen schematischen Überblick über den Planungsansatz dar. In insgesamt drei Hauptphasen wird aus einem Modell des zu fertigenden Produkts ein Programm für die Roboterzelle ermittelt, das die tatsächliche Montage ausführt.

Einheitliche Produktbeschreibung

In Phase 1, der Strukturanalyse (beschrieben in Kapitel 5), wird das Modell des zu fertigenden Produkts in ein für die Planung geeignetes Format überführt. Dieses speziell für den Planungsansatz entwickelte Format wird **einheitliche Produktbeschreibung** genannt. Schritt (1), die Formatierung, dient dabei dazu, das oft im CAD-Format oder in einem proprietären Datenformat vorliegende Produktmodell in einem ersten Schritt zunächst in das Format der einheitlichen Produktbeschreibung zu übersetzen, das die Gesamtkonstruktion räumlich darstellt. In einem zweiten Schritt versucht die Extraktion (2) über eine Analyse der einheitlichen Produktbeschreibung explizit bestehende wie auch implizit vorhandene Eigenschaften und Zusammenhänge zwischen den verschiedenen Bauteilen zu identifizieren und überführt diese in das für die Planung verständliche Format, die Ziel-Produktsituation. Am Beispiel LEGO liegt eine Beschreibung der fertigen Konstruktion meist in Form von Bildern (z. B. auf der Verpackung) vor. Daraus können im Formatierungsschritt die verbauten Legosteine abgelesen und deren räumliche Position bestimmt werden. Bei der Extraktion wird anhand der räumlichen Information über die Steine ermittelt, welche Steckverbindungen zwischen einzelnen Steinen in der fertigen Konstruktion vorhanden sein müssen.

In Phase 2, der Prozessplanung (beschrieben in Kapitel 6), findet eine erste Planung auf abstrakter Ebene der Domäne statt, ohne dabei eine konkrete Umsetzung mit Robotern zu betrachten. Die Identifikation (3) bestimmt dabei mögliche abstrakte Prozesse zur Umsetzung der expliziten sowie impliziten Produkteigenschaften und konkateniert diese zu möglichen Prozessabläufen. Neben der gewünschten Ziel-Produktsituation wird dabei auch die initial vorherrschende Produktsituation benötigt, um aus deren Differenz notwendige Montageschritte abzuleiten. Für die Planung steht eine Vielzahl an möglichen Prozessabläufen von der initialen zur Ziel-Situation zur Verfügung, von denen ein gültiger Ablauf zu bestimmen ist. Über die Filterung (4) werden dabei ungültige Prozessabläufe identifiziert und von der weiteren Planung ausgeschlossen. Unsinnige Produktvarianten der zahlreichen Permutationsmöglichkeiten werden dadurch bereits frühzeitig erkannt und entfernt, wodurch die Komplexität der Prozessplanung reduziert wird. Bei LEGO kann die initiale Produktsituation z. B. eine grüne Grundplatte sein, auf die die zu montierende Konstruktion aufgesteckt wird. Die einzelnen Legosteine des Bauteils werden erst während der Fertigung über Bereitsteller oder Magazine geeignet zugeführt und sind demnach nicht Bestandteil der initialen Produktistuation. Die Prozessplanung identifiziert mögliche Reihenfolgen, in denen die Steine zur Gesamtkonstruktion zusammengesetzt werden können. Reihenfolgen, in denen versäumt wurde, unten einzubauende Steine vorher zu setzen, werden dabei als ungültig verworfen.

Phase 3, die Fertigungsplanung (beschrieben in Kapitel 7), widmet sich der Umsetzung mit konkreten Robotern und versucht, zu vorher gewählten Prozessabläufen konkrete Automatisierungslösungen zu bestimmen. In einem ersten Schritt – der Zuweisung (5) – wird anhand einer gegebenen Roboterzelle ermittelt, wie die einzelnen Schritte eines Prozessablaufs mit den Möglichkeiten der Roboterzelle umgesetzt werden können. Die daraus resultierenden möglichen Automatisierungsabläufe werden in der anschließenden Überprüfung (6) auf verschiedene Kriterien hin untersucht, um ungültige Abläufe mit beispielsweise kollidierenden Robotern oder instabilen Zwischenprodukten herauszufiltern. Die übrig bleibenden, gültigen Automatisierungsabläufe haben oftmals – besonders wenn mehrere Roboter zur Montage eingeplant wurden – noch ein gewisses Optimierungspotential, beispielsweise um die Gesamtausführungszeit zu reduzieren. Dieses Potential wird im letzten Schritt, der Optimierung (7), genutzt, um ein möglichst gutes Ergebnis der Planung zu erhalten. Am Beispiel von LEGO werden abstrakte Prozessschritte durch Greif-, Transport- und Absetzoperationen in Automatisierungslösungen überführt. Dabei können Kollisionen auftreten oder instabile Teilkonstruktionen entstehen (z. B. Teilbrücke ohne Stützung), die anschließend ausgefiltert werden.

Im Allgemeinen entstehen bei der Planung extrem viele Möglichkeiten, um aus einer Produktbeschreibung ein konkretes Montageprogramm zu ermitteln. Bei der Identifikation (3) entstehen üblicherweise bereits immens viele Möglichkeiten, wie das Produkt abstrakt zusammengesetzt werden kann. Jeder einzelne dieser möglichen Prozessabläufe besteht aus einer Sequenz von Prozessschritten, für die es in der Zuweisung (5) je nach Roboterzelle selbst wiederum eine große Menge an potentiellen Automatisierungslösungen gibt. Kombiniert man nun noch alle möglichen Lösungen einzelner Prozessschritte zu allen möglichen Prozessabläufen, erhält man einen immens großen Raum an Lösungen, worin die Suche nach einer optimalen Lösung – vielleicht sogar

überhaupt nach einer existenten Lösung – der Suche nach einer Nadel im Heuhaufen gleicht. Eine Untersuchung aller möglichen Abläufe – sowohl bei der Prozess- als auch bei der Fertigungsplanung – ist daher insbesondere für komplexere Planungsprobleme im Allgemeinen nicht umsetzbar. Stattdessen verfolgt der Planungsansatz eine verwobene, inkrementelle Strategie, die eine zweischichtige Planung mit einer Art Makro- und Mikroschritt-Aufteilung zwischen Domäne und Automatisierung verfolgt. Bei der Prozessplanung werden nicht etwa alle möglichen Prozessabläufe im voraus berechnet, sondern es wird ein allererster Prozessschritt bestimmt, der aus Sicht der initialen Produktsituation eine Annäherung an die angestrebte Ziel-Produktsituation bringt. Für diesen einen Prozessschritt wird vorausgreifend im Rahmen der anschließenden Fertigungsplanung bei der Optimierung (7) eine Automatisierungslösung (Ablauf an Automatisierungsschritten) gesucht, die zum einen gültig und zum anderen möglichst optimal ist. Ist eine optimale Automatisierungslösung für den Prozessschritt ermittelt, wird zu der Identifikation (3) zurückgesprungen und aufbauend auf dem bisherigen Ergebnis die Prozessplanung mit der Selektion des nächst zu untersuchenden Prozessschritts fortgeführt. Wird in einem Schritt hingegen keine Automatisierungslösung für einen Prozessschritt gefunden, wird auf Ebene der Prozessplanung nach einer Alternative gesucht. Dieses Vorgehen wiederholt sich, bis der gefundene Ablauf die komplette Herstellung der angestrebten Ziel-Produktsituation beschreibt.

4.1.1 Nomenklatur bei der Montageplanung

Zur Formalisierung von Prozessschritten und Fertigungsabläufen existieren unterschiedliche Notationen. Eine davon ist die Formalisierte Prozessbeschreibung nach VDI/VDE 3682 [121], wonach einzelne Prozesse eines Ablaufs durch sogenannte Prozessoperatoren beschrieben werden. Ein Prozessoperator weist dabei eine Menge an Produkten aus, auf die eine bestimmte Operation angewandt werden soll, sowie die ausführenden Aktuatoren. Ebenso sind die bei der Operation veränderten oder neu entstehenden Produkte angegeben. Über eine Verkettung der jeweils entstehenden Zwischenprodukte über Prozessoperatoren kann letztlich der gesamte Prozessablauf modelliert werden. Bei dieser Modellierung von Montageprozessen ist allerdings eindeutig festgelegt, wie das Zwischenprodukt zu jedem Zeitpunkt der Montage aussieht. Eine Veränderung der Reihenfolge oder eine Parallelausführung der Operationen ist hier nicht mehr möglich. Als Eingabe für den Planungsansatz $PaRTs$ ist die Modellierung des Produkts über Prozessoperatoren daher ungeeignet. Denn gerade die Reihenfolgen der Operatoren und die zugewiesenen Aktuatoren sollen für die Planung möglichst als Freiheitsgrade zur Verfügung stehen. Daher wird für die in dieser Arbeit betrachtete Prozess- und Fertigungsplanung eine eigene Nomenklatur eingeführt, die im Besonderen auf die Planung von Montageprozessen abgestimmt ist.

Bauteil

Einzelteil

Ein **Bauteil** beschreibt generell eine Komponente, die zum Zusammenbau mit anderen Bauteilen genutzt werden kann. Ein Bauteil kann dabei zum einen ein **Einzelteil** sein, das nach DIN 6789 [23] einen Gegenstand darstellt, der nicht zerstörungsfrei zerlegt werden kann. Für den Planungsansatz beschreibt das Einzelteil somit einen grundlegenden Rohstoff, der als atomares Element der Planung zur Verfügung steht. Zum anderen kann ein Bauteil ein bereits zusammengesetztes Konstrukt aus anderen Bautei-

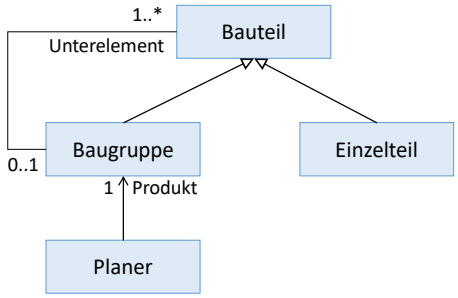


Abbildung 4.2. Übersicht über die verwendete Nomenklatur in der Domäne der Montage.

Baugruppe

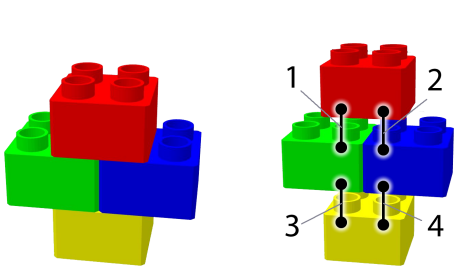
len darstellen und wird dann **Baugruppe** genannt. Nach DIN EN ISO 10209 [24] ist eine Baugruppe ein in sich geschlossener, aus mehreren Bauteilen bestehender Gegenstand. Die Baugruppe kann daher aus Einzelteilen aber auch aus anderen untergeordneten Baugruppen bestehen und beschreibt damit ggf. eine hierarchisch gegliederte Definition eines Bauteils. Neben den Unter-Bauteilen selbst enthält eine Baugruppe zusätzlich noch räumliche Informationen, die die geometrische Ausrichtung der Bauteile zueinander beschreibt. Eine Baugruppe, die einen bestimmten Fertigstellungsgrad erreicht hat und damit einem gewünschten Erzeugnis entspricht, wird auch **Produkt** genannt. Im Rahmen der Planung ist oft von Zwischenprodukten die Rede, wenn eine bestimmte, gesondert betrachtete Teilmontage erfolgt ist.

Produkt

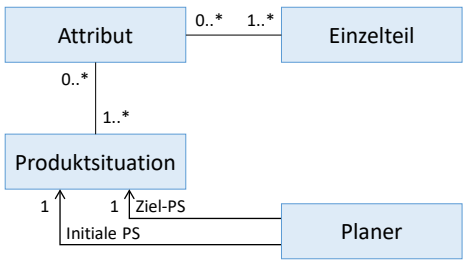
Abbildung 4.2 stellt die für den montagezentrierten Ansatz verwendete Nomenklatur und die Beziehungen der beschriebenen Objekte dar. Die Beschreibung eines Produkts über die logische Gliederung in hierarchisch geschachtelte Baugruppen findet überwiegend in der einheitlichen Produktbeschreibung Anwendung. Die eigentliche Planung mit Robotern stützt sich hingegen auf eine reine Betrachtung der Einzelteile, um Verarbeitungsschritte für sie zu finden. Dabei stellt jedes Einzelteil für die Planung eine unverwechselbare Instanz dar. Sollte es mehrere Einzelteile desselben Typs geben, z. B. mehrere gleichfarbene Legosteine im selben Rastermaß, so ist dennoch jedem seine feste Identität und Rolle zugeordnet, die über den gesamten Planungsverlauf beibehalten wird. Man kann sich dies als eine Art unsichtbaren Notitzzettel an jedem Einzelteil vorstellen, auf dem steht, welche konkrete Rolle das Einzelteil im späteren, fertig montierten Produkt einnehmen wird.

4.1.2 Datenmodell der Planung: Attribute und Produktsituationen

Bei der suchbasierten Planung benötigt der Planer ein Datenmodell zur Beschreibung des Suchraums. In einem zustandsbasierten Suchraum muss jeder erreichbare Zustand mit all seinen für die Planung relevanten Faktoren eindeutig beschrieben werden können. Dafür reicht allerdings das Modell der einheitlichen Produktbeschreibung nicht aus. Statt einer räumlichen Einordnung der Bauteile steht bei der Planung die schrittweise Montage der Einzelteile miteinander im Fokus. Dazu bedarf es einer Beschreibung, die eine existierende Beziehung – sei sie geometrisch oder semantisch – zwischen Einzelteilen ausdrückt.



(a) Attribute einer LEGO-Struktur



(b) Definition der Produktsituation

Abbildung 4.3. Vier Attribute beschreiben eine Diamant-Konstruktion als Produktsituation (a). Definition der Produktsituation über Attribute als Beschreibung von Eigenschaften von Einzelteilen (b).

Produktsituation

Ziel-Produktsituation

Initiale Produktsituation

Attribut

Im vorgestellten Planungsansatz erfolgt die Beschreibung eines Zustands über eine sogenannte **Produktsituation**. Sie repräsentiert im Allgemeinen einen bestimmten Herstellungszustand eines Produkts während der Montage. Die zwei wichtigsten Vertreter, die der Planer als Eingabeparameter benötigt, sind die **Ziel-Produktsituation**, die das zu fertigende Produkt in seiner Gesamtheit repräsentiert, sowie die **initiale Produktsituation**, die den Ausgangszustand vor Beginn der Montage darstellt. Jede Produktsituation beschreibt für jedes Einzelteil, das im fertigen Produkt verbaut sein wird, wie und mit welchen anderen Einzelteilen es aktuell verbaut ist. Eine Produktsituation ist somit eine Zusammensetzung an Eigenschaften, die zum jeweiligen Zeitpunkt der Produktsituation im Bauteil gelten. Diese Eigenschaften (auch **Attribute** genannt) können einerseits semantische oder geometrische Informationen über ein Einzelteil enthalten, andererseits können sie auch einen Zusammenhang zwischen mehreren Einzelteilen darstellen. Abbildung 4.3b stellt die Beziehung zwischen Produktsituationen, Attributen und Einzelteilen schematisch dar. Eine formale Beschreibung von Einzelteilen und von Attributen ist in Definition 4.1 gegeben:

Definition 4.1. *Einzelteil, Attribut*
*Sei $\mathcal{O}$ die Menge aller Objekte des Produkts und der Roboterzelle. Die Menge $\mathcal{E} \subseteq \mathcal{O}$ sei gegeben durch all diejenigen Objekte $o \in \mathcal{O}$, die ein **Einzelteil** e des zu fertigenden Produkts darstellen. Die Menge $\mathcal{O} \setminus \mathcal{E}$ beschreibt somit die Objekte der Roboterzelle.*

*Ein **Attribut** a ist gegeben durch ein Prädikat, das ausdrückt, ob eine Verbindung oder Eigenschaft eines oder mehrerer Objekte zu einem gegebenen Zeitpunkt der Montage gilt. Bezieht sich das Attribut ausschließlich auf Einzelteile des Produkts (für alle Einzelteile e des Attributs gilt $e \in \mathcal{E}$) wollen wir es **Produktattribut** nennen; bezieht es sich hingegen auf mindestens ein Objekt der Roboterzelle (z. B. Roboter), so ein **Aktuatorattribut**.*

Produktattribut

Aktuatorattribut

Sei $\mathcal{A}$ die Menge aller Attribute, $\mathcal{A}\restriction Produkt \subseteq \mathcal{A}$ die Menge aller Produktattribute und $\mathcal{A}\restriction Aktuator = \mathcal{A} \setminus \mathcal{A}\restriction Produkt$ die Menge aller Aktuatorattribute.

*Auf der Menge $\wp\mathcal{A}$ der Teilmengen von Attributen gebe es ein Prädikat $kons : \wp\mathcal{A} \to \mathbb{B}$, das ausdrückt, dass eine Menge von Attributen **konsistent** ist.*

Konsistenz

Eine Menge von Attributen ist dann als konsistent anzusehen, wenn sie in der realen Welt in dieser Konstellation existieren können. Stelle man sich zwei Legosteine vor,

zu denen zwei Attribute definiert sind. Beschreiben diese jeweils einen zueinander widersprüchlichen geometrischen Offset, so ist jede Attributmenge, die diese beiden Attribute enthält, inkonsistent. Ebenso ist eine Attributmenge auch dann nicht konsistent, wenn durch sie eine Konstruktion beschrieben wird, bei der beispielsweise verschiedene Einzelteile ineinander ragen – also kollidieren.

Abbildung 4.3a skizziert am Beispiel LEGO eine aus vier Steinen zusammengesetzte diamantenförmige Konstruktion. Sie enthält insgesamt vier Attribute, die jeweils einen Zusammenhang zwischen zwei Legosteinen ausdrücken – sie sind also Produktattribute. In Kombination beschreiben die vier Attribute damit eine Produktsituation, die den dargestellten Diamant repräsentiert. Im Beispiel sind alle vier Attribute ähnlich, sie beziehen sich jeweils auf zwei Steine, setzen diese in einen räumlichen Zusammenhang und beschreiben semantisch eine Steckverbindung. Für andere Planungsprobleme oder in anderen Domänen sind weitere Attribut-Typen denkbar, die etwa andere Füge-Arten zwischen Einzelteilen (z. B. „geklebt“) oder bestimmte Modifikationen an einem einzelnen Teil (z. B. „lackiert“) ausdrücken. Im Allgemeinen lassen sich Montageprobleme mit den folgenden drei üblichen Typen von Produktsichtattributen beschreiben:

Beziehung

Beziehung: Sie beschreibt einen geometrischen und semantischen Zusammenhang zwischen zwei oder mehreren Einzelteilen. Zum einen definiert eine Beziehung – wie im Beispiel des Diamanten aus LEGO angedeutet – einen räumlichen Zusammenhang, über den dreidimensional definiert ist, wie die Einzelteile zueinander positioniert sind. Zum anderen enthält ein Beziehungs-Attribut weiterhin die semantische Information, um welche Art von Verbindung es sich handelt. Die Attribute im LEGO-Beispiel beschreiben, dass es sich bei der Beziehung um eine gesteckt-Verbindung als spezielle Form der Fügung zwischen zwei Steinen handelt. Hieraus kann später ein entsprechender Montageprozess abgeleitet werden, der mit einer geeigneten Zueinanderbewegung der beiden Steine diese aufeinandersteckt und damit die Fügung an der geometrisch vorgegebenen Stelle bewirkt.

Eigenschaft

Eigenschaft: Sie bezieht sich auf ein Einzelteil und definiert eine bestimmte semantische Information, beispielsweise in welchem Zustand sich das Einzelteil befindet, etwa nachdem ein bestimmter Prozess darauf angewandt wurde. Dies mag beispielsweise die erfolgte Aufbringung von Lackfarbe auf einem Einzelteil sein. Am Beispiel der CFK-Fallstudie kann der Grad der Verformung, die in einen Zuschnitt eingebracht wurde, durch ein Eigenschafts-Attribut ausgedrückt werden.

Fixpunkt

Fixpunkt: Diese Art von Attribut beschreibt eine geometrische Fixierung eines Einzelteils. Werden Einzelteile Zusammengesetzt, so ist nur ihre relative Position wichtig, nicht aber ihre Position in der Welt beziehungsweise in der Roboterzelle. Damit ist eine Montage „in der Luft“ möglich, bei der beispielsweise zwei Roboter die Einzelteile in einer zueinander abgestimmten Bewegung zusammenfügen. Oft wird in der Montage allerdings eine Art Trägerplatte oder Grundbauteil verwendet, die entweder statisch in der Roboterzelle steht oder auf einem Fließband die Produktion durchläuft. Darauf erfolgt im Weiteren der Zusammenbau des Produkts. Ein solches Trägerobjekt kann mittels eines Fixpunkt-Attributs als statisches Objekt markiert werden, sodass es bei der Planung nicht etwa von einem

Roboter gegriffen und umplaziert werden kann. Damit ist bei der Automobilmontage beispielsweise erwirkbar, dass stets das Rad zur Karosserie gebracht wird und nicht umgekehrt.

Im Gegensatz zur einheitlichen Bauteilbeschreibung, die eine strukturierte Definition über hierarchische Baugruppen ermöglicht, verwendet die Modellierung mit Produktsituationen gewollt keine rekursive Untergliederung über Baugruppen. Damit soll verhindert werden, dass die Planung durch eine etwa feste Baugruppenstruktur auf bestimmte Fertigungsreihenfolgen eingeschränkt wird und etwa Unterbaugruppen zuerst fertigzustellen wären, bevor ein Zusammenbau beider Bauteile erfolgen kann. Stattdessen beziehen sich alle Attribute unmittelbar auf die vorhandenen Einzelteile, wonach die Produktsituation mit einer Baugruppe vergleichbar ist, die eine hierarchische Verschachtelungstiefe von eins hat.

Für eine Fertigungsplanung mit Robotern muss neben dem Zustand des Bauteils, der über geltende Attribute in einer Produktsituation beschrieben ist, ebenso der Zustand der Roboterzelle ausdrückbar sein. Dafür ist es zum einen notwendig, dass die räumliche Position und die zum jeweiligen Zeitpunkt geltende Konfiguration von technischen Ressourcen – beispielsweise die Achsstellungen des Roboters – im Planungsverlauf berücksichtigt wird. Zum anderen müssen auch technische Ressourcen über Attribute mit Einzelteilen in Beziehung gebracht werden können, beispielsweise wenn ein Objekt von einem Greifer gegriffen wurde. Beschreibt man neben dem Bauteilzustand (Produktsituation) nun auch den Zustand der Roboterzelle durch Einbeziehung von Attributen über technische Ressourcen, so wird die Menge an Attributen **Planungssituation** genannt. Abbildung 4.4 zeigt die Produktsituation als Teil der Planungssituation und beschreibt die jeweils zugehörigen Arten an Attributen. In einer Planungssituation werden insgesamt drei Arten zur Beschreibung des Roboterzellen-Zustands verwendet:

Planungssituation

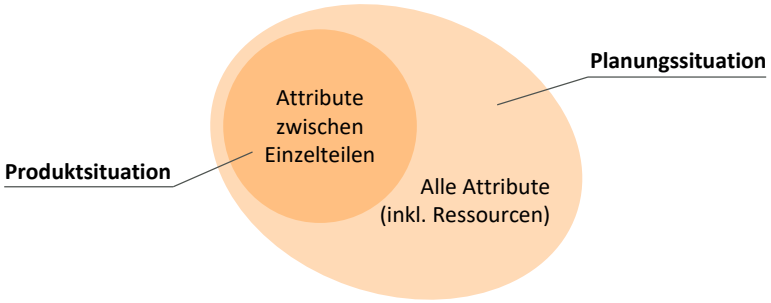


Abbildung 4.4. Gruppierung der Attribute zur Produkt- und Planungssituation.

Belegung

Belegung: Analog zur Beziehung, die zwischen zwei Einzelteilen gilt, wird über die Belegung ein Zusammenhang zwischen dem Endeffektor einer technischen Ressource und einem Einzelteil beschrieben (z. B. Objekt von Roboter gegriffen).

Konfiguration

Konfiguration: Sie beschreibt eine eindeutige Einstellung einer technischen Ressource, z. B. die Achsstellungen eines Roboters oder den Öffnungswinkel eines Greifers.

Position

Position: Eine technische Ressource kann damit räumlich positioniert werden; sie definiert somit beispielsweise den Arbeitsraum eines Roboterarms in der Roboterzelle.

Definition 4.2 beschreibt die Planungssituation und die Produktsituation formal:

Definition 4.2. *Situation*
*Eine **Situation** (auch **Planungssituation** genannt) $s \subseteq \mathcal{A}$ ist gegeben durch eine konsistente Menge von Attributen, die zu einem gegebenen Zeitpunkt der Montage gelten, d. h. $kons(s)$. Sei $\mathcal{S}$ die Menge aller Situationen.*

*Eine **Produktsituation** $p \subseteq \mathcal{A}\restriction Produkt$ ist gegeben durch eine Menge von Produktattributen, die zu einem gegebenen Zeitpunkt der Montage gelten. Für eine Planungssituation s sei $s\restriction Produkt$ die zugehörige Produktsituation. Sei $\mathcal{S}\restriction Produkt = \{s\restriction Produkt \mid s \in \mathcal{S}\}$ die Menge aller Produktsituationen.*

All diejenigen Attribute einer Situation $s \in \mathcal{S}$, die für ein gegebenes Objekt $o \in \mathcal{O}$ gelten, sind durch $attr(o, s)$ beschrieben.

Soll festgestellt werden, ob mit einer Planungssituation ein bestimmter Zustand des Bauteils erreicht ist, so wie er in einer Produktsituation – beispielsweise der Ziel-Produktsituation – beschrieben ist, so kann die Planungssituation mit der Produktsituation verglichen werden. Dazu werden die von beiden ausgewiesenen Produktattribute abgeglichen, folglich diejenigen, die sich lediglich auf Einzelteile beziehen. Die Planungssituation wird sozusagen auf eine Produktsituation reduziert. Stimmen beide Mengen an Produktattributen überein, so **erfüllt** die gegebene Planungssituation die (Ziel-)Produktsituation. Die formale Beschreibung dazu ist in Definition 4.3 gegeben:

Erfüllung

Definition 4.3. *Erfüllung*
*Eine Planungssituation $s \in \mathcal{S}$ **erfüllt** eine Produktsituation $p \in \mathcal{S}\restriction Produkt$, wenn s und p dieselben Produktattribute beinhalten; i. Z. $s \Vdash p$ gdw. $s\restriction Produkt = p$.*

4.1.3 Abstraktion zwischen Domäne und Automatisierung

Ziel der Planung ist es, einen Ablauf an Aktionen (sogenannten **Tasks**) zu finden, die jeweils einen auf ein oder mehrere Einzelteile anzuwendenden **Prozess** beinhalten und insgesamt die Fertigung des Produkts beschreiben. Um die Komplexität dieser Planung beherrschbar zu gestalten, untergliedert sich der vorgestellte Planungsansatz unter anderem in die zwei Phasen der Prozess- und Fertigungsplanung. Die Unterteilung in eine zunächst abstrakte, domänenorientierte Planung und eine anschließende Umsetzung unter Betrachtung konkreter Automatisierungslösungen erfordert ebenso eine Separierung der Planungskonzepte und der dabei entstehenden Artefakte.

Task

Prozess

Im Rahmen der Prozessplanung wird zunächst ermittelt, welche abstrakten Schritte notwendig sind, um ein gegebenes Produkt herzustellen, und in welcher Reihenfolge sie auszuführen sind. Stellt man sich einen Menschen vor, der ein Haus aus LEGO errichtet, so wäre die Prozessplannung gleichbedeutend mit dem Betrachten des Bilds auf der Verpackung und dem Überlegen einer mentalen Strategie, in welcher Reihenfolge die einzelnen Steine zu setzen sind. Die in der Domäne möglichen abstrakten Aktionen – hier das Setzen von Steinen – werden **Domänenprozesse** genannt. Wird ein Domänenprozess konkret auf ein oder mehrere Einzelteile angewandt, so spricht man, wie in Abbildung 4.5 dargestellt, von einem **Domänentask**. Im Rahmen der zustandsbasierten Planung entspricht der Domänentask einer Aktion/Transition zwischen Zuständen.

Domänenprozess

Domänentask

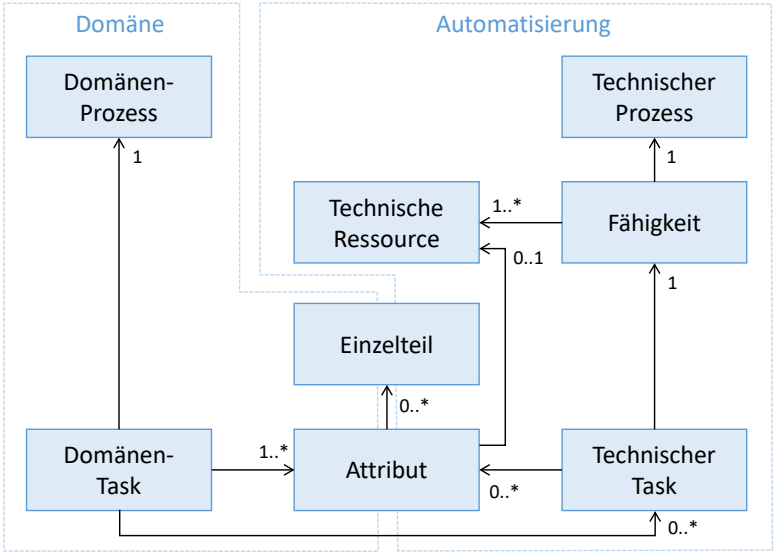


Abbildung 4.5. Überblick über die verschiedenen Konzepte, zugeordnet zu den Bereichen Domäne und Automatisierung.

Jeder Domänentask benennt Attribute, die bei einer fiktiven Ausführung des Tasks ab- beziehungsweise aufgebaut werden müssen. Ist eine konkrete Produktsituation gegeben, die vor der Anwendung des Domänentasks gilt, so lässt sich anhand seiner Attribut-Änderung eine resultierende Produktsituation berechnen.

Technische Ressource

Technischer Task

Bis hierhin ist das Vorgehen völlig losgelöst von der Verwendung jeglicher Aktuatorik (im Folgenden **technische Ressource** oder synonymisch Roboter genannt). Bei der Fertigungsplanung, deren Aufgabe es nun ist herauszufinden, wie abstrakte Domänentasks in der Realität ausgeführt werden können, rücken jedoch die technischen Ressourcen in den Fokus. Ziel ist es, für jeden Domänentask eine Menge an **technischen Tasks** zu finden, die das abstrakt beschriebene Verhalten unter Zuhilfenahme von technischen Ressourcen konkret umsetzt. Im Legohaus-Beispiel ist es der Mensch, der den nächsten Stein in der Schachtel sucht, ihn mit seinen Fingern greift und knapp über die abzusetzende Stelle bringt, bevor er ihn mit leichtem Druck auf den darunterliegenden Steinen fixiert. In Analogie zum Domänentask ist es auf Ebene der Automatisierung der technische Task, der eine konkrete Aktion ausführt und dabei eine Menge an Attributen auf- oder abbaut, wodurch er bestimmte Einzelteile beeinflusst oder manipuliert. Auch hier beschreibt der Task eine spezifische Anwendung eines Prozesses – in diesem Fall des **technischen Prozesses** – auf konkrete Einzelteile. Im Gegensatz zur Domäne werden im Bereich der Automatisierung Tasks über technische Ressourcen umgesetzt, deren Vermögen, einen bestimmten Prozess ausführen zu können, über eine **Fähigkeit** ausgedrückt wird. Mit einer Fähigkeit wiederum kann ein Task abgeleitet werden, der einen bestimmten Prozess mit gegebenen Robotern umsetzt. Im Gegensatz zu Domänentasks können technische Tasks zudem Attribute auf- beziehungsweise abbauen, die sich auf technische Ressourcen beziehen, beispielsweise ein „Gegriffen“-Attribut

Technischer Prozess

Fähigkeit

zwischen Greifer und Einzelteil. Der Zustand der Roboterzelle und des Produkts ist somit nach wie auch vor der Ausführung eines technischen Tasks in einer Planungssituation beschreibbar. Die formale Beschreibung von Tasks ist in Definition 4.4 gegeben:

Definition 4.4. *Task*
*Sei t ein **Task** und $\mathcal{T}$ die Menge aller Tasks.*

Für einen Task $t \in \mathcal{T}$ sei $ExplEffekt^+(t) \subseteq \mathcal{A}$ die Menge aller Attribute, die von t explizit als aufzubauend erwähnt werden, und $ExplEffekt^-(t) \subseteq \mathcal{A}$ die Menge aller Attribute, die von t explizit als abzubauend erwähnt werden. Für jeden Task muss $ExplEffekt^+(t) \cap ExplEffekt^-(t) = \emptyset$ gelten. Das Paar

$$ExplEffekt(t) = (ExplEffekt^+(t), ExplEffekt^-(t))$$

beschreibt damit die von t definierten Attributänderungen vollständig.

*Die Menge der **Domänentasks** $\mathcal{T}\restriction Produkt$ ist gegeben durch die Menge derjenigen Tasks $t \in \mathcal{T}$, die ausschließlich Produktattribute aufbauen oder abbauen, d. h. $ExplEffekt^+(t) \subseteq \mathcal{A}\restriction Produkt$ und $ExplEffekt^-(t) \subseteq \mathcal{A}\restriction Produkt$.*

*Ein Task $t \in \mathcal{T} \setminus \mathcal{T}\restriction Produkt$ heißt **technischer Task**.*

Sei $\mathcal{P}$ die Menge aller Prozesse, die eine abstrakte technische Aktion (technischer Prozess) oder eine Aktion der Domäne (Domänenprozess) repräsentieren. Für einen Task $t \in \mathcal{T}$ beschreibt $process(t) = p$ mit $p \in \mathcal{P}$ den Prozess, der von Task t ausgeführt wird.

4.1.4 Einbeziehung von Expertenwissen

Bei der Inbetriebnahme von automatisierten Fertigungsanlagen im industriellen Umfeld sind verschiedene Menschen mit unterschiedlichen Expertisen involviert. So gibt es z. B. den Ingenieur, der das zu fertigende Produkt konstruiert, einen Domänenexperten, der sich mit den Fertigungsprozessen der Domäne auskennt, sowie einen Automatisierungsexperten, der die Ausführung der Prozesse auf die Roboteranlage bringt. Der vorgestellte Planungsansatz $PaRTs$ zielt zwar auf eine weitgehend automatische Planung ab, er befreit allerdings nicht von der Notwendigkeit all dieser Expertisen. Daher wird das Wissen der Experten auf eine geordnete und weitestgehend generische Weise erfasst und für Berechnungen und Abwägungen im Planungsverlauf verfügbar gemacht. Außerdem wird so ermöglicht, dass die Experten – anders als oftmals bei der konventionellen Produktion – unabhängig voneinander ihre Expertise beitragen können.

In der schematischen Übersicht über den Planungsansatz $PaRTs$ in Abbildung 4.1 (Kapitel 4) sind die verschiedenen Phasen und deren Unterschritte illustriert. Die jeweils am weitesten links stehenden Pfeile (grün hinterlegt) werden der Planung in den verschiedenen Schritten als Artefakte zugeführt. Sie sind von Experten explizit für das konkret vorliegende Planungsproblem zu definieren. So bestimmt der Ingenieur das zu fertigende Produkt (3D-Daten), das der Planung in Schritt (1) zugrunde liegt. Der Domänenexperte steuert in Schritt (3) die initiale Produktsituation bei, auf die die Planung fußt. Darüber ist beschrieben, wo sich die Einzelteile des fertigen Produkts zu Beginn der Planung befinden. Befinden sich die Einzelteile zu Beginn noch nicht in der

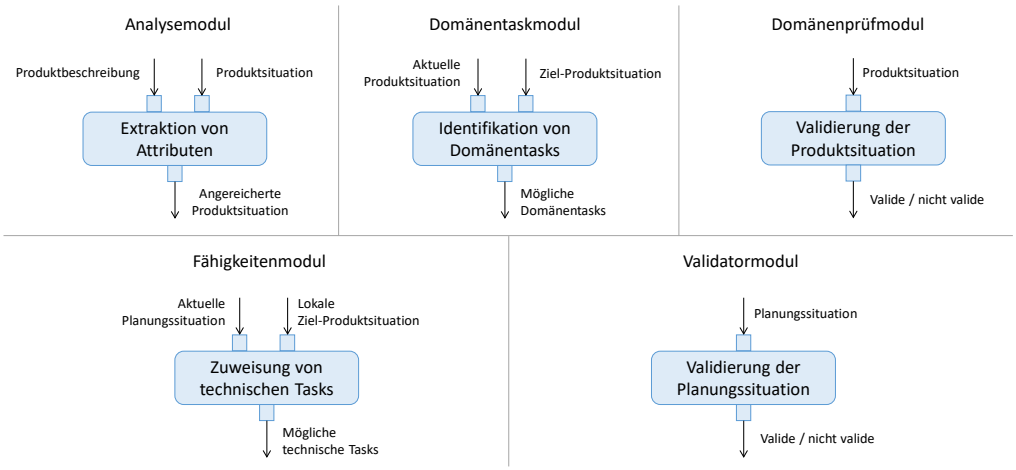


Abbildung 4.6. Module und ihr Zusammenhang zu Artefakten als Möglichkeit zur Einbeziehung von Expertenwissen.

Fertigungszelle, sondern werden sie beispielsweise über ein Magazin im Laufe der Fertigung bereitgestellt, so ist die initiale Produktsituation oftmals auch leer. Das bedeutet, dass sie zu Beginn keinerlei Attribute enthält und vom finalen Produkt folglich keinerlei Teilmontagen der Einzelteile vorhanden sind. Wird die Montage auf einem Trägerobjekt (bei LEGO kann dies eine Basisplatte sein) ausgeführt, dann enthält die initiale Produktsituation in diesem Beispiel zumindest ein Fixpunkt-Attribut für das Trägerobjekt. In Schritt (4) ist eine **Definition der Roboterzelle** durch den Automatisierungsexperten notwendig, um konkrete Prozessausführungen durch Roboter im dreidimensionalen Raum bestimmbar zu machen. Diese Definition enthält die zur Verfügung stehenden technischen Ressourcen (Roboter, Greifer, etc.) und deren räumliche Anordnung sowie sämtliche Konfigurationsdaten der Aktuatorik.

Definition der Roboterzelle

Diese drei Artefakte, die 3D-Daten, die initiale Produktsituation sowie die Definition der Roboterzelle, sind von den Experten auf das konkrete Produkt und die gegebene Fertigungsanlage zugeschnitten beizusteuern. Für die Planung reichen diese Angaben allerdings nicht aus. Weiterhin sind unter anderem abgeleitete Informationen über das Bauteil notwendig, die in Kombination mit möglichen Prozessausführungen in gültige Roboterprogramme überführt werden können. Der vorgestellte Planungsansatz erlangt solche zusätzlichen Informationen während der Planung aus zur Verfügung gestellten Modulen, die generisches Expertenwissen enthalten und entweder domänen-, prozess- oder automatisierungsspezifische Berechnungen und Analysen bereitstellen. In Abbildung 4.1 (Kapitel 4) sind diese Module als grün hinterlegte Kästchen zu den jeweils zugeordneten Schritten eingezeichnet.

In Schritt (2) der Planung, der Extraktion, wird aus dem räumlichen Modell des fertigen Bauteils eine Ziel-Produktsituation abgeleitet, die um Prozesswissen angereichert ist und der Planung als Eingabeparameter dient. Die Anreicherung erfolgt dabei durch von Domänenexperten bereitgestellte **Analysemodule**, die aus der vorliegenden einheitlichen Produktbeschreibung weitere für die Planung notwendige Informationen ermittelt.

Analysemodul

Am Beispiel LEGO untersucht etwa ein spezifisches Modul die räumliche Beziehung zwischen jeweils zwei Steinen und erkennt, ob zwischen diesen eine gesteckte Fügeverbindung vorliegt. Diese wird vom Analysemodul in Form eines Attributs zu einer Ziel-Produktsituation beigetragen. Gleichermaßen können weitere Module analysieren, ob Klebeverbindungen, Farblackierungen oder sonstige Verarbeitungen der Einzelteile im fertigen Bauteil vorhanden sein müssen. Ein Modul kann sich dabei intern auf geometrische Untersuchungen zur Identifikation von Attributen beschränken oder sich auch auf komplexere Berechnungen wie statische Analysen – beispielsweise zur Ermittlung der notwendigen Festigkeit einer Klebeverbindung und des daher zu verwendenden Klebstoffs – beziehen. Das Vorgehen der Strukturanalyse über Analysemodule wird in Abschnitt 5.2.1 erläutert.

Anhand der mit Attributen angereicherten Ziel-Produktsituation werden in Schritt (3) des Planungsansatzes mögliche Prozessabläufe identifiziert. Dafür werden von Domänenexperten sogenannte **Domänentaskmodule** zur Verfügung gestellt, die auf Basis dessen, welche Teilmontagen des Produkts bereits erfolgt sind, die nächsten möglichen Schritte in Form von abstrakten Domänentasks ermitteln. Als Eingabeparameter erwartet das Modul die aktuelle Produktsituation sowie die angestrebte Ziel-Produktsituation und erstellt daraus eine Menge an möglichen Nachfolge-Domänentasks. Die Funktion und Verwendung von Domänentaskmodulen wird in Abschnitt 6.1.1 erläutert.

Domänentaskmodul

Nicht alle Prozessabläufe, die durch die identifizierten Domänentasks beschrieben werden, sind unbedingt valide. Abhängig von der Domäne können manche der Abläufe zu ungültigen Produktsituationen führen. Daher werden diese von **Domänenprüfmodulen** untersucht und ggf. aussortiert. Ein Domänenprüfmodul wird von Domänenexperten zur Verfügung gestellt und fungiert als Prüfer für ein bestimmtes Kriterium der Domäne. Anhand einer ihm übergebenen Produktsituation kann das Modul entscheiden, ob es im Hinblick auf das von ihm untersuchte Kriterium valide ist oder nicht.

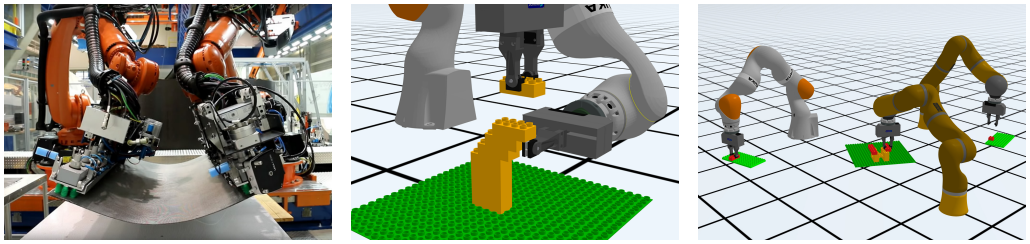
Domänenprüfmodul

Schritt (4) als Teil der Fertigungsplanung sucht für jeden Domänentask eine konkrete Automatisierungslösung in Form einer Sequenz von technischen Tasks. Von Automatisierungsexperten werden dazu **Fähigkeitenmodule** bereitgestellt, die basierend auf einem bisher gefundenen Automatisierungsprogramm einen nächsten mit Robotern umsetzbaren Schritt ermitteln, der sich dem im Domänentask hinterlegten Fertigungsziel annähert. Als Eingabe erwartet ein Fähigkeitenmodul eine aktuell geltende Planungssituation, die den konkreten Zustand von Bauteil und Roboterzelle beschreibt, sowie die angestrebte Ziel-Produktsituation des Domänentasks, und liefert dafür eine Menge an möglichen nächsten technischen Tasks. Fähigkeitenmodule und deren Anwendung werden in Abschnitt 7.1.3 erklärt.

Fähigkeitenmodul

Oftmals resultieren bei der Planung identifizierte technische Tasks in ungültige Planungssituationen, beispielsweise wenn eine Kollision zwischen Robotern oder zwischen Roboter und Bauteil auftritt oder wenn das entstandene Bauteil statisch instabil ist. Damit nicht jedes Fähigkeitenmodul alle solchen (teils domänenspezifischen) Validierungen selbst berücksichtigen muss, werden unabhängig davon nach jedem Schritt der Planung **Validatormodule** zur wiederkehrenden Überprüfung des Ergebnisses – also der resultierenden Planungssituation – herangezogen. Ein Validatormodul kann dabei sehr allgemeine Eigenschaften wie Kollisionsfreiheit überprüfen; aber auch domänen-

Validatormodul



(a) Explizite Kooperation [7] **(b)** Implizite Kooperation **(c)** Parallelarbeit

Abbildung 4.7. Formen der Kooperation, die im Planungsansatz berücksichtigt oder direkt einbezogen sind. Explizite Kooperation durch zwei Roboter, die einen CFK-Zuschnitt gemeinsam tragen (a). Implizite Kooperation durch einen Roboter, der das Absetzen eines Legosteins durch einen zweiten Roboter ermöglicht (b). Unabhängiges paralleles Arbeiten mehrerer Roboter im gleichen Arbeitsraum (c).

spezifische Eigenschaften, wie die Stabilität von Legostrukturen, oder Erfordernisse der Roboterzelle können über Validatormodule kontrolliert werden. Solche Module werden sowohl von Domänen- wie auch Automatisierungsexperten beigetragen.

4.2 Planung für Roboter-Teams

Werden bei der Fertigung eines Produkts mehrere Roboter eingesetzt, so steigt die Komplexität der Planung aufgrund der Vielzahl an kombinatorischen Möglichkeiten sehr schnell an. Die Anzahl an verfügbaren Fähigkeiten führt zu einem exponentiellen Anstieg an Planungsmöglichkeiten. Ist die Planung aber einmal erfolgreich erfolgt, kann die Ausführung der Fertigung davon profitieren, dass mehrere Roboter – ggf. parallel – am Bauteil arbeiten. Denn dadurch kann etwa Produktionszeit eingespart und damit der Durchsatz erhöht werden, oder aber es eröffnet sich die Möglichkeit, auch größere und schwerere Bauteile in einem Produkt zu verarbeiten.

Die Verwendung mehrerer Roboter ermöglicht auf der einen Seite neue Herstellungsmethoden, auf der anderen Seite muss man sich über ein sicheres Zusammenarbeiten der Roboter Gedanken machen. Insgesamt drei Formen der Kooperation zwischen Robotern können identifiziert werden und sind an verschiedenen Stellen im vorgestellten Planungsansatz $PaRTs$ berücksichtigt oder einbezogen.

Explizite Kooperation

Explizite Kooperation: Arbeiten zwei oder mehr Roboter explizit an ein und demselben Prozess zusammen, beispielsweise am gemeinsamen Transport eines schweren oder großen Teils (siehe Abbildung 4.7a), so erfordert die Kooperation oftmals eine exakte Synchronisation unter Echtzeitgarantien. Diese Art von Kooperation ist im Planungsansatz innerhalb eines Fähigkeitenmoduls verwendbar, das von einem Automatisierungsexperten entsprechend erstellt wird. Das Fähigkeitenmodul kann bei der Planung konventionell einbezogen werden, ohne dass der Planer die Kooperation eigens berücksichtigen muss. Auch die Planungskomplexität bleibt von der Einbeziehung zusätzlicher expliziter Kooperationen unberührt. Der Planer muss lediglich sicherstellen, dass bei Verwendung mehrerer Roboter keiner davon gleichzeitig in mehr als einem parallelen Task involviert ist.

Implizite Kooperation

Implizite Kooperation: Manchmal kann bei der Fertigung eine Situation eintreten, in der ein Roboter ein bestimmtes Setting in der Roboterzelle oder am Produkt benötigt, das durch eine zuvor oder gleichzeitig ausgeführte Aktion eines anderen Roboters hergestellt werden muss. Beispielsweise ist beim Aufbauen einer aufsteigenden Treppe aus LEGO (siehe Abbildung 4.7b) ab einer bestimmten Höhe die Konstruktion derart instabil, dass ein Daraufsetzen des nächsten Steins allein schon durch die Aufdrückkraft ein Zusammenbrechen der darunterliegenden Konstruktion bewirkt. Hilfreich wäre hier ein zweiter Roboter, der vor dem Absetzen des nächsten Steins die Konstruktion von unten geeignet unterstützt und hält. Die Kooperation der Roboter ergibt sich dabei aus der Kombination von an sich unabhängigen Fähigkeiten. Diese Variante der Kooperation muss also nicht unbedingt von Experten definiert und der Planung zugeführt werden. Stattdessen kann es die Aufgabe des Planers selbst sein, geeignete Verkettungen entsprechender technischer Tasks zu finden, die solche impliziten, für die Fertigung notwendigen Kooperationen ergeben. Der Experte kann verschiedene Fähigkeitenmodule dadurch unabhängig voneinander entwerfen, ohne explizite Abhängigkeiten dabei berücksichtigen zu müssen. Allerdings stellt das Finden impliziter Kooperationen einen erheblichen Mehraufwand für die Planung dar und erhöht aufgrund der kombinatorischen Möglichkeiten die Planungskomplexität. Da die spezielle Kombination der Fähigkeiten, die einen notwendigen Prozess erst ermöglicht, im Vorhinein gänzlich unbekannt ist, gleicht die Planung nahezu einer blinden Suche im Suchraum der möglichen Fähigkeiten. So kann es vorkommen, dass ein Validatormodul stehts zu einer Ablehnung des gefundenen Ablaufs führt, bis eben exakt die eine Fähigkeit zur Vorbereitung einer Situation gewählt wurde, in der eine spezielle zweite Fähigkeit zum Erfolg führt. Möchte man die Komplexität der Suche reduzieren, kann der Experte die für die Fertigung notwendigen Kooperationen alternativ über die Angabe expliziter Kooperationen ausdrücken.

Parallelarbeit

Parallelarbeit: Roboter, die parallel und unabhängig voneinander arbeiten, befinden sich in einer speziellen, weniger offensichtlichen Form der Kooperation (siehe Abbildung 4.7c). Die parallele Ausführung eröffnet dem Planer großes Potential, das Ergebnis im Hinblick auf die Gesamtperformanz hin zu optimieren – also die Ausführungszeit des Plans durch eine maximale Parallelisierung zu minimieren. Gleichzeitig besteht durch die Parallelarbeit aber zunehmend auch die Gefahr, dass es bei den Bewegungen der gleichzeitig in der Roboterzelle arbeitenden Roboter zu einer Kollision kommt. Zur Parallelisierung von Fertigungsabläufen verwendet der Planungsansatz einen abhängigkeitsbasierten Ansatz, der einerseits eine parallele Manipulation desselben Einzelteils verhindert und zum anderen eine Kollisionsfreiheit vor und nach Ausführung eines jeden technischen Tasks sicherstellt. Ein weiterer Ansatz basierend auf räumlichen Bewegungskörpern untersucht asynchrone Bewegungen auf potentielle Kollisionen und wendet verschiedene Strategien zu deren Vermeidung an. Die parallele Ausführung birgt aber noch weitere Herausforderungen, die die Planung zu bewältigen hat. Insbesondere die Verwebung der Tasks gleichzeitig arbeitender Roboter führt – im Vergleich zu einer rein sequentiellen Abfolge von Tasks nur eines Roboters – dazu, dass konkrete Ursachen beispielsweise von Deadlocks (erfolglose Planungszustände) schwieriger zu identifizieren sind. Ebenso sind notwendige Kombinationen von technischen Tasks, die

sich über mehrere, teilweise bezugslose oder gar unnötige Schritte erstrecken können, schwerer zu finden, wie es auch bei der impliziten Kooperation der Fall ist.

Der Planungsansatz $PaRTs$ nutzt weitestgehend die Freiheitsgrade und Möglichkeiten, die notwendig sind, um ein optimales Ergebnis zu finden, sofern es denn existiert. Gleichzeitig werden Konzepte angewandt, die die Komplexität des Planungsproblems beherrschbar machen und somit eine effiziente und auch auf größere Planungsprobleme skalierbare Lösung ermöglichen.

Zusammenfassung. Für eine automatische Planung muss das zu fertigende Produkt in einem maschinenlesbaren Beschreibungsformat vorliegen. Dieses Kapitel stellt dazu eine einheitliche Produktbeschreibung vor, die Informationen über Bestandteile und den Aufbau des zu fertigenden Produkts ausdrückt. Anschließend wird darauf eingegangen, wie eine vorliegende Produktbeschreibung über die automatisierte Analyse eingelesen und ausgewertet wird. Auf Basis von beigesteuertem Expertenwissen werden hierbei aus der Beschreibung zusätzliche Prozessinformationen zur Fertigung sowie relevante Parameter für die nachgelagerte Planung extrahiert.

5 Strukturanalyse

Wann immer ein Produkt von einem Werker von Hand hergestellt wird, ist eine detaillierte Beschreibung des Produkts – etwa in Form einer Anleitung oder einer technischen Zeichnung – unerlässlich. Die Wahl der konkreten Bearbeitungsmethode oder des Werkzeugs liegt dabei oftmals im Ermessen des Werkers auf Basis seiner langjährigen Erfahrung. Wird die Fertigung nun nicht mehr manuell sondern automatisiert und mit Hilfe von Maschinen realisiert, so muss zusätzlich zur Beschreibung des Produkts auch das Erfahrungswissen des Werkers, das nicht explizit in der Anleitung erfasst ist, geeignet in der entstehenden Automatisierungslösung abgebildet werden.

Um eine automatische Planung von Fertigungsschritten zur Herstellung eines Produkts zu ermöglichen, ist zudem weiteres Wissen über das Produkt, über seine Struktur, die Verarbeitung sowie über Besonderheiten der jeweiligen Domäne notwendig. Daher ist es bereits bei der Konstruktion des Produkts wichtig, dass sämtliche Daten und Informationen, die in die Erstellung der Produktkonstruktion durch den Ingenieur eingeflossen sind, erfasst werden und in die spätere Planung einfließen können. Dabei ist nicht zuletzt der Detailgrad ausschlaggebend dafür, ob daraus automatisiert Prozesse für die Fertigung abgeleitet werden können. Im Falle von LEGO liegen den Bausätzen grafische Anleitungen bei, die schrittweise die benötigten Teile auflisten und entsprechende Fügeprozesse mit räumlichen Darstellungen erläutern. Bei der Produktion von Bauteilen aus CFK-Verbundwerkstoffen hingegen liegen die vom Ingenieur zur Verfügung gestellten Daten oftmals in sogenannten Plybooks im CAD-Format vor, die sowohl Konturen und

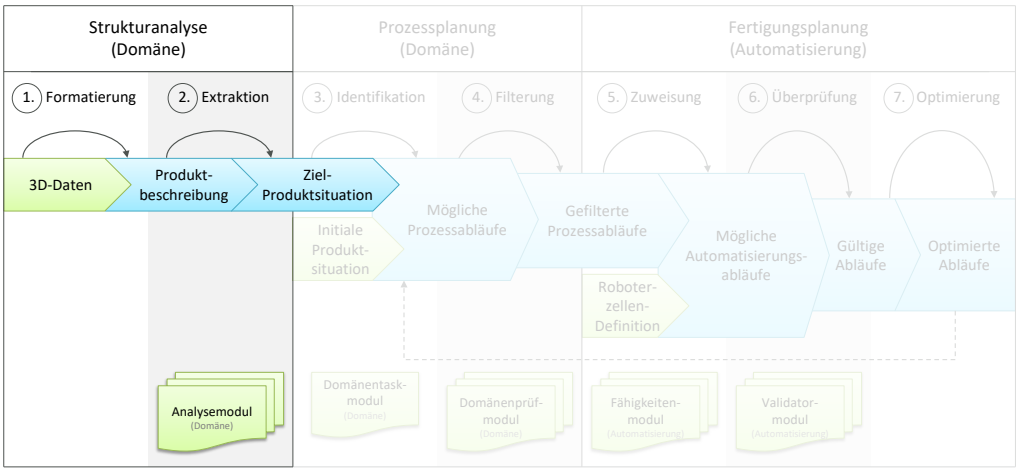


Abbildung 5.1. Einordnung der Strukturanalyse in den Gesamtansatz $PaRTs$.

Faserorientierungen der einzelnen Zuschnitte als auch deren finale Ablageposition in der Drapierform beinhalten.

Ziel der Strukturanalyse (Phase 1 des Planungsansatzes) ist es, der nachfolgenden Planung eine Beschreibung des Endprodukts in Form einer Ziel-Produktsituation bereitzustellen. Dafür wird im ersten Schritt – der Formatierung (1) – die oftmals domänenspezifische Beschreibung des zu fertigenden Produkts in das maschinenlesbare Format der einheitlichen Produktbeschreibung überführt, das sämtliche zur Fertigungsplanung relevanten Informationen ausdrücken kann (siehe Abschnitt 5.1). Aus dieser einheitlichen Produktbeschreibung werden in Schritt 2 (Extraktion) semantische Informationen und Zusammenhänge des Produkts abgeleitet (siehe Abschnitt 5.2), die im Späteren als Grundlage für die automatische Planung dienen. Abbildung 5.1 ordnet die beiden Schritte der Strukturanalyse und die jeweiligen Artefakte in den Gesamtansatz ein.

5.1 Einheitliche Produktbeschreibung

Grundlage für den automatischen Planungsansatz ist ein einheitliches und maschinenlesbares Format der Produktbeschreibung. Dieses muss zum einen Informationen zu Einzelteilen enthalten, aus denen das Bauteil gefertigt wird, zum anderen müssen Angaben über die Struktur des Produkts gemacht und insbesondere die Anordnung der Einzelteile innerhalb von Baugruppen definiert werden.

Bauteil-beschreibung

Einzelteil-beschreibung

Baugruppen-beschreibung

Abbildung 5.2 zeigt den Aufbau der einheitlichen Produktbeschreibung für die automatische Planung. Enthalten sind **Bauteilbeschreibungen**, die im Wesentlichen den Typ eines zusammenhängenden Objekts repräsentieren. Einerseits kann dies die Beschreibung eines einfachen Einzelteils (**Einzelteilbeschreibung**) sein, das neben einem detaillierten grafischen Modell für eine eventuelle Visualisierung auch ein im Detailgrad reduziertes Kollisionsmodell zur Berücksichtigung bei der späteren Planung enthält. Eine Bauteilbeschreibung kann andererseits auch eine Beschreibung einer zusammengesetzten Baugruppe sein – die sogenannte **Baugruppenbeschreibung**. Diese enthält

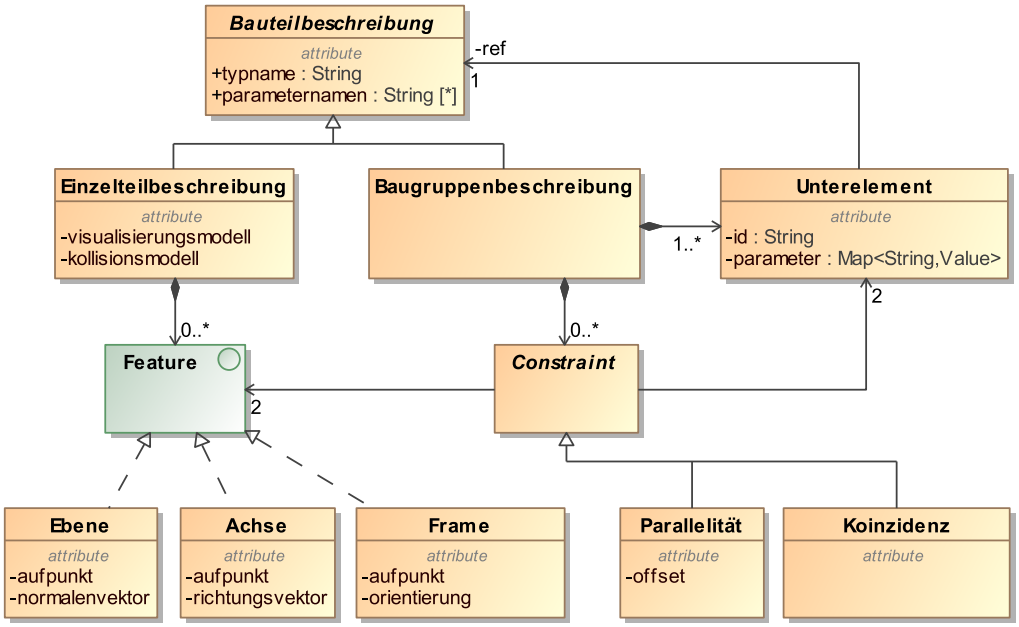


Abbildung 5.2. Aufbau der einheitlichen Produktbeschreibung.

Unterelemente, die selbst wiederum von einem bestimmten Bauteiltyp sind und damit eine bestimmte Bauteilbeschreibung referenzieren.

Eine Bauteilbeschreibung kann für das beschriebene Objekt bestimmte Platzhalter für Eigenschaften definieren, die bei konkreten Bauteilen dieses Typs variieren dürfen. Dafür sind Parameter vorgesehen, über die beispielsweise die Farbe oder die Größe des Bauteils eingestellt werden können. Die Einzelteilbeschreibung wird durch solche Parameter direkt beschrieben. Eine Baugruppenbeschreibung kann eigene Parameter lediglich an seine eigenen Unterelemente weiterreichen. Beispielsweise kann damit an einer Stelle die Farbe aller Einzelteile in einer Baugruppe definiert werden. Für jedes Unterelement einer Baugruppenbeschreibung werden selbst wiederum eigene konkrete Parameterwerte vergeben. Dabei können Unterelemente vom selben Typ beispielsweise in unterschiedlichen Farben zu einer solchen Baugruppe gehören.

Die Position von Unterelementen zueinander innerhalb einer Baugruppe ist ebenfalls Bestandteil der Baugruppenbeschreibung und wird über die Angabe räumlicher Constraints definiert. Damit können räumliche Zusammenhänge wie etwa parallel zueinander ausgerichtete Achsen oder deckungsgleiche Ebenen (Flächen) der einzelnen Unterelemente festgelegt werden.

Baugruppenbeschreibungen können gezielt dazu verwendet werden, Typen von Zwischenbauteilen zu definieren und damit andere Baugruppenbeschreibungen – insbesondere bei größeren Bauteilen – beliebig hierarchisch zu dekomponieren und zu strukturieren. Dank der Abstrahierung über Unterelemente und deren Parametrisierbarkeit können Baugruppen in unterschiedlichen Zwischenprodukten in unterschiedlicher Parametrisierung mehrfach verwendet und auch projektübergreifend als wiederverwendbare

```
<!-- Beschreibung eines Einzelteils -->
<Einzelteilbeschreibung id="schrankkorpus" params="höhe">
  <Model src="schrankkorpus_{$höhe}.stp" />
</Einzelteilbeschreibung>

<Einzelteilbeschreibung id="schranktür" params="farbe,höhe">
  <Model src="schranktür_{$höhe}_{$farbe}.stp" />
</Einzelteilbeschreibung>

<Einzelteilbeschreibung id="fußnoppe">
  <Model src="fußnoppe.stp" />
</Einzelteilbeschreibung>

<!-- Beschreibung einer Baugruppe -->
<Baugruppenbeschreibung id="schrank" params="farbe,höhe">
  <Unterelement id="schrankkorpus" ref="schrankkorpus" params="höhe:{$höhe}" />
  <Unterelement id="schranktür1" ref="schranktür"
      params="farbe:{$farbe},höhe:{$höhe}" />
  <Unterelement id="schranktür2" ref="schranktür"
      params="farbe:{$farbe},höhe:{$höhe}" />
  <Unterelement id="fußnoppe1" ref="fußnoppe" />
  <Unterelement id="fußnoppe2" ref="fußnoppe" />
  <Unterelement id="fußnoppe3" ref="fußnoppe" />
  <Unterelement id="fußnoppe4" ref="fußnoppe" />
</Baugruppenbeschreibung>
```

Listing 5.1. Beispiel für eine Produktbeschreibung im XML-Format. Beschrieben ist die hierarchische Zusammensetzung eines Schranks mit zwei Flügeltüren.

Komponenten eingesetzt werden. Dies kommt aktuellen industriellen Trends entgegen, wonach aus ökonomischen Gründen die Variation an unterschiedlichen Produkten möglichst gering gehalten wird und oftmals standardisierte Kleinteile verbaut werden. In einem Fertigteil werden zudem, wo möglich, Teilkomponenten des gleichen Typs verbaut, um einerseits den Bedarf an unterschiedlichen Bauteilen einzuschränken und andererseits den Fertigungsprozess zu vereinfachen.

Liegen die Eingabedaten für die Fertigung eines Produkts nicht bereits in dieser einheitlichen Form der Produktbeschreibung sondern etwa in einem proprietären Modell vor, ist zunächst eine Überführung in die Zielbeschreibung nötig. Dies kann manuell für jedes einzelne Produkt über einen entsprechenden Experten angefertigt werden, der sowohl das ursprüngliche Modell versteht als auch Kenntnisse über die Modellierung mit der Produktbeschreibung verfügt. Insbesondere falls zahlreiche Prozessplanungen unterschiedlicher Produkte im gleichem Ursprungsformat stattfinden, ist der Einsatz von Compilern vorgesehen, die automatisiert vom ursprünglichen in das gewünschte Format übersetzen können.

Listing 5.1 zeigt ein Beispiel der einheitlichen Produktbeschreibung im XML-Format für den Aufbau eines Schranks mit zwei Flügeltüren. Die drei Einzelteilkomponenten, die bei der Planung als Ausgangsmaterialien zur Verfügung stehen, sind vom Typ Schrankkorpus, Schranktür und Fußnoppe. Bei der Beschreibung der Einzelteile ist jeweils die Verlinkung zu einer STEP-Datei angegeben, die eine 3D-Beschreibung des Objekts zur späteren Visualisierung und ggf. zur Planung beinhaltet. Die Einzelteilbeschreibung für

den Schrankkorpus weist einen Parameter „höhe“ aus, der eine individuelle Höhenangabe einzelner Schränke erlaubt. Der bei einer konkreten Referenzierung übergebene Parameter kann über die Dollar-Notation in geschweiften Klammern bezogen werden und wird im Beispiel zur Spezifikation des Dateinamens zum Modell herangezogen, um die entsprechende Datei zu laden. Der Einfachheit halber wird hier angenommen, dass ein STEP-Modell für jede verwendete Höhe existiert. Die Schranktür kann neben der Höhe auch über eine beliebige Farbe individualisiert werden.

Die Baugruppenbeschreibung des Schranks referenziert den Korpus, zwei Flügeltüren, sowie vier Fußnoppen zur Anbringung auf der Unterseite des Korpus, auf denen der Schrank später stehen wird. Bei der Referenzierung des Schrankkorpus sowie der Schranktüren werden für die benötigten Parameter der Unterelemente die jeweils eigenen Parameterwerte weitergegeben. Somit enthält ein Schrank in einer gewünschten Höhe und Farbe einen Korpus in der richtigen Höhe, sowie zwei Türen mit richtiger Höhe und in der angegebenen Farbe.

5.1.1 Definition geometrischer Beziehungen über Constraints

Die Beschreibung einer komplexen zusammengesetzten Baugruppe definiert nicht nur ihre Einzelkomponenten, aus denen sie besteht, sondern auch deren räumliche Beziehung untereinander. In der hierarchischen Baugruppenbeschreibung werden solche geometrischen Zusammenhänge über **Constraints** beschrieben, die die sechs Freiheitsgrade der räumlichen Drehverschiebung (Translation und Rotation) entweder einschränken oder gänzlich festlegen. Ein Constraint bezieht sich dabei mit der räumlichen Eigenschaft, die er ausdrückt, auf zwei Einzelteile, und zwar im Detail auf jeweils eines ihrer sogenannten **Features** (siehe Abbildung 5.2). Ein Feature repräsentiert ein markantes Oberflächenmerkmal eines Einzelteils wie etwa eine ebene Fläche, eine Kante oder eine zentrische Achse. Dabei muss ein Feature in seiner Position und Ausrichtung zu seinem Einzelteil eindeutig beschrieben sein. So ist eine Ebene durch einen Aufpunkt und einen Normalenvektor beschrieben, eine Achse durch einen Aufpunkt und einen Richtungsvektor. Auch eine Ecke kann als Feature verwendet werden, das entweder in Form eines Punkts ohne Orientierungseigenschaft oder als vollqualifiziertes Koordinatensystem (Frame) mit Orientierung definiert wird. Features eines Einzelteils vererben sich dabei auf Baugruppenbeschreibungen, in denen das Einzelteil enthalten ist. Eine komplexe Baugruppenbeschreibung besitzt damit alle Features der in sich enthalten Unterkomponenten. Dadurch können Constraints auch Baugruppen in räumliche Beziehung setzen, deren Features durch deren hierarchisch enthaltenen Einzelteile definiert werden.

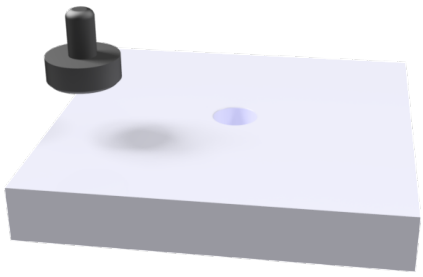
Constraint

Feature

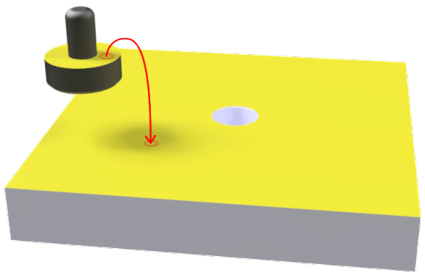
Innerhalb einer Baugruppenbeschreibung werden die darin enthaltenen Bauteile in eine räumliche Beziehung gesetzt, indem Constraints zwischen Features der Bauteile zur Baugruppenbeschreibung hinzugefügt werden. Tabelle 5.1 beschreibt eine Auswahl an Möglichkeiten von Constraints zwischen unterschiedlichen Typen an Features, die für die Beschreibung in den Domänen der beiden Fallstudien gewählt wurden. Eine Koinzidenz repräsentiert ein Übereinanderliegen von Features, was bei Achsen, Ebenen und Frames möglich ist. Eine Parallelität kann zwischen entweder zwei Achsen oder

Tabelle 5.1. Verschiedene Möglichkeiten an Constraints.

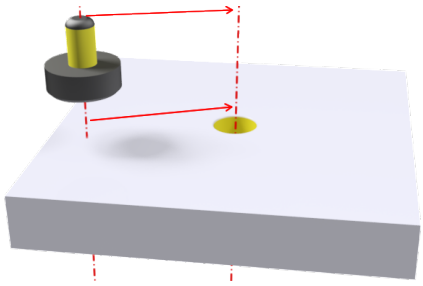
Constraint	Features		Einschränkung der Drehverschiebung
	Baut. 1	Baut. 2	
Koinzidenz	Frame	Frame	Festlegung aller translatorischen und rotatorischen Achsen. Freiheitsgrade: keine.
	Achse	Achse	Festlegung zweier translatorischer Achsen und deren Rotationen. Die Ausrichtung der beiden Achsen (mitläufig oder entgegengerichtet) ist dabei festgelegt. Freiheitsgrade: Verschiebung entlang der gemeinsamen Koinzidenzachse, Rotation um diese Achse.
	Ebene	Ebene	Festlegung einer translatorischen Achse und der Rotationen um die beiden anderen Achsen. Die Ausrichtung der Ebenen-Normalen zueinander ist dabei festgelegt. Freiheitsgrade: Verschiebung in zwei Richtungen parallel zu den Ebenen, Rotation um die Normalen.
Parallelität	Achse	Achse	Festlegung zweier translatorischer Achsen und deren Rotationen. Die Ausrichtung der beiden Achsen (mitläufig oder entgegengerichtet) ist dabei festgelegt. Ein Abstand (räumliche Verschiebung) zwischen den Achsen wird berücksichtigt. Freiheitsgrade: Verschiebung entlang der Parallel-Achsen, Rotation um diese Achsen.
	Ebene	Ebene	Festlegung einer translatorischen Achse und der Rotationen um die beiden anderen Achsen. Die Ausrichtung der Ebenen-Normalen zueinander ist dabei festgelegt. Ein Abstand zwischen den Flächen wird berücksichtigt. Freiheitsgrade: Verschiebung in zwei Richtungen parallel zu den Ebenen, Rotation um die Normalen.



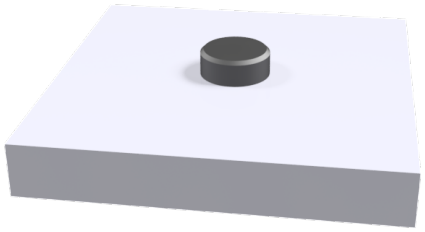
(a) Schrankkorpus und Fußnoppe



(b) Definition einer Koinzidenz zwischen zwei Ebenen



(c) Definition einer Koinzidenz zwischen zwei Zylinder-Mittelachsen



(d) Ergebnis der Ausrichtung nach Anwendung beider Constraints

Abbildung 5.3. Beispiel einer geometrischen Ausrichtung von Bauteilen anhand von Constraints: Positionierung einer Fußnoppe in eine dafür vorgesehene Bohrung auf der Schrankkorpus-Unterseite.

zwei Ebenen definiert werden und beschreibt eine gegenseitige Ausrichtung mit einem definierbaren räumlichen Abstand.

Während eine Koinzidenz zwischen zwei Frames die räumliche Beziehung eindeutig beschreibt, schränken Constraints zwischen Achsen und Ebenen nur einen Teil der sechs räumlichen Freiheitsgrade ein. Für eine vollständige Beschreibung der geometrischen Beziehung zweier Bauteile kann es daher nötig sein, dass mehrere Constraints zwischen deren Features definiert werden. Erst aus einer vollständigen geometrischen Beschreibung kann eine eindeutige Drehverschiebung zwischen den beteiligten Bauteilen abgeleitet werden. Eine ungeschickte Wahl mehrerer Constraints zwischen zwei Bauteilen kann allerdings dazu führen, dass die geometrische Beziehung überbestimmt oder sogar widersprüchlich ist. In einem solchen Fall kann die Drehverschiebung nicht aufgelöst werden und führt zu einer fehlerhaften Produktbeschreibung.

Abbildung 5.3 illustriert anhand des Schrank-Beispiels die Verwendung von Features und Constraints. Eine Fußnoppe soll so positioniert sein, dass sie bündig in einer dafür vorgesehenen Bohrung auf der Schrankkorpus-Unterseite eingepasst ist und dabei mit der Noppenkopf-Innenseite auf der Oberfläche des Korpus aufliegt. Die Korpus-Unterseite des Schranks ist in Abbildung 5.3 exemplarisch über ein rechteckiges Brett dargestellt, das mittig eine Bohrung für die Anbringung der Fußnoppe aufweist. Die Fußnoppe besteht aus einem flachen Kopf sowie einem zylindrischen Hals (Stift), der in

```
<Einzelteilbeschreibung id="schrankkorpus" params="hoehe">
  ...
  <Feature id="unterseite" typ="ebene" aufpunkt="..." normalenvektor="..." />
  <Feature id="bohrung1" typ="achse" aufpunkt="..." richtungsvektor="..." />
  <Feature id="bohrung2" typ="achse" aufpunkt="..." richtungsvektor="..." />
  <Feature id="bohrung3" typ="achse" aufpunkt="..." richtungsvektor="..." />
  <Feature id="bohrung4" typ="achse" aufpunkt="..." richtungsvektor="..." />
</Einzelteilbeschreibung>

<Einzelteilbeschreibung id="fußnoppe">
  ...
  <Feature id="auflagefläche" typ="ebene" aufpunkt="..." normalenvektor="..." />
  <Feature id="mittelachse" typ="achse" aufpunkt="..." richtungsvektor="..." />
</Einzelteilbeschreibung>

<Baugruppenbeschreibung id="schrank">
  ...
  <Constraint typ="koinzidenz" feature1="schrankkorpus.unterseite"
      feature2="fußnoppe1.auflagefläche" flip="true" />
  <Constraint typ="koinzidenz" feature1="schrankkorpus.bohrung1"
      feature2="fußnoppe1.mittelachse" />
</Baugruppenbeschreibung>
```

Listing 5.2. Erweiterung der Einzelteilbeschreibungen um Features und Constraints, beispielhaft dargestellt zur Anbringung einer Fußnoppe an den Schrankkorpus.

der Bohrung des Schrankkorpus versenkt wird. Markante Merkmale der beiden Objekte sind zum einen Flächen auf der Korpus-Unterseite und der Noppenkopf-Innenseite. Zum anderen zeichnen sich die Mittelachsen in der Bohrung und inmitten des Noppenhalses als Features aus. Zur Positionierung der beiden Objekte zueinander wird eine Ebenenkoinzidenz zwischen der Noppenkopf-Unterseite und der Korpusoberfläche hergestellt (siehe Abbildung 5.3b), die einhergehend mit einer Achsenkoinzidenz zwischen den Mittelachsen des Noppenstifts und der Bohrung (siehe Abbildung 5.3c) die relative Position der Fußnoppe zum Korpus festlegt. Einzig die Rotation zueinander um die gemeinsame Mittelachse ist hierbei nicht bestimmt. Wegen der Symmetrie des Noppenkopfes kann diese allerdings vernachlässigt und im weiteren Verlauf durch den Planer festgelegt werden, um daraus eine eindeutige Drehverschiebung extrahieren zu können.

Listing 5.2 erweitert die XML-Definition zur Beschreibung eines Schranks in Listing 5.1 um diese beiden Constraints der Baugruppenbeschreibung exemplarisch für eine Fußnoppe und ergänzt die Einzelteilbeschreibungen um die dafür notwendigen Features. Die Beschreibung des Schrankkorpus weist ein Feature aus, das die Unterseite des Korpus repräsentiert, sowie vier Features für die Mittelachsen der Bohrungen, die zur Anbringung der Fußnoppen vorgesehen sind. Die Fußnoppe definiert eine Mittelachse sowie eine Ebene auf der Unterseite ihres Kopfes. In der Baugruppenbeschreibung des Schranks werden die beiden Ebenen und die beiden Achsen über Koinzidenzen zusammengefügt, wobei die Normalenvektoren der beiden Ebenen entgegengerichtet sind (über den Parameter *flip* spezifizierbar).

5.1.2 Einbeziehung planungsrelevanter Prozessparameter

Im Rahmen der automatischen Planung werden gültige Abläufe geeigneter Prozessschritte gesucht, die das gegebene Produkt möglichst effizient herstellen. Dabei ist die Anzahl an Möglichkeiten der variablen Ausprägungen, verschiedenen Anordnungen und konkreten Parametrisierungen der einzelnen Prozessschritte oftmals immens. Das Finden einer optimalen Lösung, die entweder eine möglichst kurze Zeittaktung besitzt, die beste Produktqualität erzielt, die effizienteste Ressourcenauslastung ausweist oder eine hieraus resultierende Erfolgskennzahl erfüllt, stellt oftmals eine nicht triviale Herausforderung in Form eines großen Komplexitätsproblems dar. Es ist möglich, dass mit der Erfahrung eines Experten der Domäne einzelne Prozessparameter bereits im Vorfeld der Planung festgelegt werden können und der Planungsaufwand damit deutlich reduziert werden kann. Auch die Qualität des Planungsergebnisses kann dadurch ggf. entscheidend beeinflusst werden. Die Verbesserung der Planung kommt besonders dann zum Tragen, wenn die Einstellungen der Prozessparameter nicht offensichtlich sind und sie einen großen Explorationsaufwand bedeuten. Es kann daher zielführend sein, der Planung solche Prozessparameter vorzugeben. Oft sind solche Parameter bereits im ursprünglichen, vom Ingenieur entwickelten Modell des zu fertigenden Produkts implizit enthalten. So ist bei einem Bauteil aus CFK beispielsweise anhand der räumlichen Position des Zuschnitts ableitbar, ob, und wenn ja, welche Reihenfolgen bei der Drapierung der Zuschnitte einzuhalten sind. Ebenso kann ein Experte zu den spezifischen Zuschnittskonturen bestimmte Drapiertechniken vorgeben.

Oftmals kann ein Domänen- oder Prozessexperte aufgrund seiner langjährigen Erfahrung Wissen über den Planungsprozess beisteuern, das aus den zugrundeliegenden Daten unter Umständen nicht ohne Weiteres automatisiert ableitbar wäre. Für derartige Informationen sind zusätzliche Angaben von Prozesseigenschaften in der Bauteilbeschreibung vorgesehen, die anschließend bei der automatischen Planung berücksichtigt werden können. Im Gesamtansatz betrachtet (siehe Abbildung 5.1) ermöglicht die Angabe eines solchen Wissens nicht nur die Vereinfachung des Analyseschritts, in dem die Vorgaben aus der Bauteilbeschreibung analysiert und gedeutet werden, sondern hat auch signifikanten Einfluss auf die Performanz der nachgelagerten Planungsphasen und die Qualität der dabei entstehenden Ergebnisse.

Abbildung 5.4 erweitert das Modell der einheitlichen Produktbeschreibung aus Abbildung 5.2. Üblicherweise werden Constraints verwendet, um Bauteile untereinander in räumliche Verbindungen zu setzen. Die Aufgabe des Planers ist es anschließend, in einem ersten Schritt die semantische Bedeutung zwischen Bauteilen zu erkennen (sind sie beispielsweise zusammengeklebt oder zusammengesteckt) und in einem zweiten Schritt mögliche Prozesse zur Umsetzung der semantischen Beziehung zu bestimmen (beispielsweise Zusammenkleben mit Heißkleber). Dabei kann es wünschenswert sein, dass beide Schritte – statt automatisch vom Planer identifiziert werden zu müssen – auch explizit vom Experten in der einheitlichen Produktbeschreibung spezifizierbar sind. Für den ersten Schritt werden anstelle von Constraints in einer Baugruppenbeschreibung dazu unmittelbar Attribute definiert, die damit erweitertes Wissen über die Planung

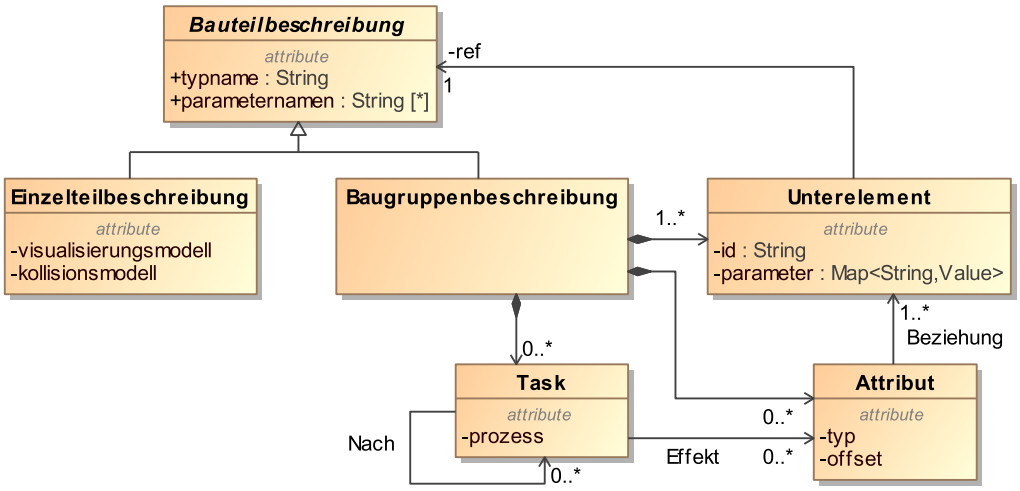


Abbildung 5.4. Prozesse und Attribute als Beschreibungsmerkmale planungsrelevanter Prozessparameter in der Bauteilbeschreibung.

zu einem oder mehreren Unterelementen der Baugruppe beitragen. Der Planer kann daraus geeignete Prozesse ermitteln, um die gegebenen Attribute umzusetzen.

Für den zweiten Schritt ist es möglich, auch Prozesse manuell vorzugeben, wodurch die Planungskomplexität weiter reduziert werden kann. Dazu sieht die einheitliche Produktbeschreibung das Element „Task“ vor, über das eine auszuführende Operation im Rahmen der Fertigung einer Baugruppe angegeben werden kann. Verglichen mit der Nomenklatur aus Abschnitt 4.1.3 entspricht der Task einem Domänentask, der unabhängig von Robotern eine abstrakte Aufgabe beschreibt. Ein Task bezieht sich damit ebenso auf eine Menge an Attributen, die im Zuge seiner Ausführung als Effekte resultieren. Diejenigen Unterelemente der Baugruppe, die durch den Task beeinflusst werden, sind über dessen Effekt-Attribute ableitbar. Werden mehrere Tasks innerhalb einer Baugruppenbeschreibung spezifiziert, so kann über eine Nachfolgebeziehung jeweils eine zeitliche Reihenfolge untereinander angegeben werden, um damit eine partielle Ordnung bei der Ausführung zu erzielen. Tasks benennen ihren Prozess, der einen abstrakten Bearbeitungsschritt der Domäne beschreibt. Dabei handelt es sich zunächst lediglich um ein Identifizierungsmerkal, für das im späteren Verlauf der Planung anhand der zur Verfügung stehenden Fähigkeiten der Roboter ein ausführbarer Task zur Realisierung des Prozesses gesucht wird.

In Listing 5.3 ist die XML-Notation der einheitlichen Produktbeschreibung am Beispiel der Schrankmontage weiter ausgeführt. Anstelle der Constraints, die zuvor die Fußnoppen am Schrankkorpus positioniert haben, sind nun Attribute angegeben, die neben der Positionierung (Offset) auch die Art der Verbindung (Fügung) zwischen jeweils einer Fußnoppe und dem Schrankkorpus beschreiben. Damit entfällt zum einen die automatische Bestimmung der Positionierung über die Auflösung von Constraints, zum anderen muss das Wissen über die Füge-Verbindungen nicht mehr implizit aus dem Modell extrahiert werden.

```
<Baugruppenbeschreibung id="schrank" params="farbe,höhe">

  <Unterelement id="schrankkorpus" ref="schrankkorpus" params="höhe:{$höhe}" />
  ...

  <Unterelement id="fußnoppe1" ref="fußnoppe" />
  <Unterelement id="fußnoppe2" ref="fußnoppe" />
  ...

  <!-- Constraints entfallen -->

  <!-- Attribute der Einzelteile -->
  <Attribut id="korpus_noppe1" typ="fügung" offset="...">
    <Beziehung unterelement="schrankkorpus" />
    <Beziehung unterelement="fußnoppe1" />
  </Attribut>
  ...
  <Attribut id="korpus_noppe4" typ="fügung" offset="...">
    <Beziehung unterelement="schrankkorpus" />
    <Beziehung unterelement="fußnoppe4" />
  </Attribut>

  <!-- Tasks zur Herstellung der Attribute -->
  <Task id="verbinde_korpus_noppe1" prozess="fügen">
    <Effekt attribut="korpus_noppe1" />
  </Task>
  ...
  <Task id="verbinde_korpus_noppe4" prozess="fügen">
    <Effekt attribut="korpus_noppe4" />
    <Nach task="verbinde_korpus_noppe1" />
  </Task>

</Baugruppenbeschreibung>
```

Listing 5.3. Erweiterung der Einzelteilbeschreibungen um Attribute und Tasks, beispielhaft zur geordneten Anbringung von zwei Fußnoppen an den Schrankkorpus.

Für die Montage der Fußnoppen sind zudem Tasks angegeben, die dem Planer die auszuführenden Operationen sowie ihre Effekte, also die dabei hergestellten Attribute, benennen. Der Task zur Montage der vierten Fußnoppe spezifiziert zudem über das „Nach"-Element einen zeitlichen Ablauf, wonach Fußnoppe 4 erst gefügt werden kann, wenn Fußnoppe 1 bereits montiert ist.

5.2 Modulare Strukturanalyse

Liegt das Modell des zu fertigenden Produkts in Form der einheitlichen Produktbeschreibung vor, so kann daraus in einem nächsten Schritt die Ziel-Produktsituation abgeleitet werden, die als Grundlage für die weiteren Planungsphasen dient. Dazu werden aus der Produktbeschreibung zunächst sämtliche Constraints, die die verschiedenen Bauteile untereinander in geometrische Beziehungen setzen, aufgelöst und daraus konsistente Positionen der Bauteile ermittelt. Damit ist eine grundlegende räumliche Struktur des Gesamtbauteils gegeben.

Um diese räumliche Struktur fertigen zu können, ist in einem weiteren Schritt ein gültiger Ablauf an geeigneten Tasks zu bestimmen. Doch um von der räumlichen Struktur auf konkret anzuwendende Tasks rückschließen zu können, ist zusätzliches Wissen über die Domäne und notwendige Verarbeitungsschritte notwendig. Zum Teil sind solche Informationen bereits explizit in der Produktbeschreibung in Form von planungsrelevanten Prozessparametern (Attribute oder Tasks) gegeben. Oftmals stecken sie allerdings versteckt (implizit) in den Angaben der Produktbeschreibung und sind lediglich mit geschultem Expertenblick und langjähriger Erfahrung ersichtlich und ableitbar.

Der Planungsansatz sieht für die automatische Ableitung solcher Zusatzinformationen aus der Produktbeschreibung ein und modulares Konzept vor, über das Domänen- sowie Prozessexperten unabhängig voneinander ihr Wissen abstrakt formulieren und zu einer automatisiert anwendbaren Wissens-Datenbasis beitragen. Für die modulare Strukturanalyse wird das notwendige Wissen in Analysemodule zusammengefasst. Angewandt auf verschiedene Produktbeschreibungen der betreffenden Domäne, dienen Analysemodule zur automatischen Extraktion und Ableitung impliziter und erweiterter Informationen.

5.2.1 Extraktion planungsrelevanter Prozessparameter: Analysemodule

In der Domäne LEGO haben zwei in der Produktbeschreibung nebeneinanderliegende Steine zwar eine geometrische Beziehung, semantisch stehen sie aber nicht in Korrelation. Bei zwei übereinanderliegenden Steinen hingegen liegt die Vermutung nahe, dass sie im fertigen Produkt durch eine formschlüssige Fügung verbunden sind. Diese Information ist – sofern nicht als planungsrelevanter Prozessparameter angegeben – üblicherweise nicht explizit in der Produktbeschreibung enthalten. Doch erst durch das Wissen über solche Attribute können später passende Prozesse zur Umsetzung zugeordnet werden.

Die Aufgabe von Analysemodulen (siehe Abbildung 4.6, Abschnitt 4.1.4) besteht darin, solche semantischen Informationen in Form von Attributen aus dem geometrischen Modell der Produktbeschreibung zu gewinnen. Das Modell wird dazu von jedem Analysemodul hinsichtlich expliziter Eigenschaften untersucht und diese in implizite Zusatzangaben überführt. Die Schlussfolgerungen, die basierend auf notwendigerweise geltenden Annahmen getroffen werden können, sind dabei für jedes Modul von Domänen- und Prozessexperten zu definieren.

Ein Analysemodul erwartet als Eingabeparameter zum einen die Produktbeschreibung des zu fertigenden Bauteils, zum anderen kann bereits eine vorgegebene Produktsituation beigesteuert werden, sollten Attribute bereits bekannt sein. Das Analysemodul kann nun die räumliche Struktur des Produkts sowie die ggf. bereits vorhandenen Attribute untersuchen und sich darauf basierend Rückschlüsse auf semantische Eigenschaften ziehen. Dabei fokussiert ein Analysemodul üblicherweise auf einen bestimmten Aspekt der Domäne. Bei LEGO ist dies beispielsweise die Erkennung, ob zwei Steine aufgrund ihrer Größe und ihrer Position zueinander aufeinander gesteckt sein müssen. Andere Analysemodule könnten statische Analysen durchführen, um zu ermitteln, mit welcher Festigkeit eine Klebeverbindung zwischen Bauteilen hergestellt werden muss. Erkennt ein Analysemodul eine solche Eigenschaft, so überführt es diese in ein entsprechendes Attribut, wobei für eine Domäne verschiedene Arten an Attributen existieren können. Für die Planung mit LEGO existiert beispielsweise ein Attribut „LegoSteckVerbindung“, das einerseits den geometrischen Zusammenhang beider Steine enthält und andererseits eine semantische Information über den späteren Prozess festlegt. Hat das Analysemodul die Analyse beendet, so werden alle neu erkannten Attribute zu der übergebenen Produktsituation hinzugefügt und als angereicherte Produktsituation zurückgegeben.

5.2.2 Fixpunktbestimmung bei der Auswertung

Für ein Planungsproblem und dessen Domäne existieren oft mehrere Analysemodule, die in der vorliegenden Produktbeschreibung verschiedene Typen an Attributen identifizieren können. Analysemodule haben die Möglichkeit, die zuvor bereits bekannten Attribute bei der Analyse mit einzubeziehen und verfeinern somit ggf. das Wissen über implizite Informationen. Bei Konstruktionen aus LEGO ist z. B. mit der Kenntnis über sämtliche „Gefügt-Attribute“ eine nachgelagerte Analyse der Statik und Ergänzung durch notwendige Stabilitätseigenschaften der Verbindungen denkbar. Für die Analyse bedeutet das allerdings, dass ein Analysemodul unterschiedliche Ergebnisse liefern kann, je nachdem, welche Attribute in der übergebenen Produktbeschreibung enthalten sind. Die Anwendungsreihenfolge der Analysemodule und die Rückspeisung der bisherigen Analyseergebnisse spielt daher eine entscheidende Rolle.

Die Strukturanalyse startet zunächst mit einer leeren Produktsituation, d. h. einer leeren Attributmenge. Sind planungsrelevante Prozessparameter bereits in der Produktbeschreibung in Form von Attributen angegeben, werden diese in die Attributmenge aufgenommen. Die Identifizierung neuer Attribute über die vorhandenen Analysemodule erfolgt anschließend in einem Fixpunktverfahren. Da Analysemodule eigenständige Komponenten sind, die unabhängig voneinander entwickelt werden und keine Abhän-

gigkeiten untereinander besitzen, gewährleistet eine repetitive Auswertung durch die Analysemodule unter ständiger Rückführung der jeweils neu identifizierten Attribute ein homogenes Analyseresultat. Sämtliche Analysemodule werden also ggf. auch mehrfach in einer beliebigen Reihenfolge solange erneut mit der aktuellsten Attributmenge ausgeführt, bis kein einziges Analysemodul mehr einen neuen Erkenntnisbeitrag liefert. Als Ergebnis erhält man eine Ansammlung aller bereits in der Produktbeschreibung explizit angegebener Attribute vereint mit den Attributen, die über die Analyse identifiziert wurden. Diese Attributmenge wird als Ziel-Produktsituation der nachgelagerten Planungsphase übergeben.

Zusammenfassung. Phase 2 des Planungsansatzes beschäftigt sich mit einer abstrakten „Vorab-Planung“ auf Domänenebene. In diesem Kapitel wird beschrieben, mit welchen Konzepten diese Planung losgelöst von konkreten Fähigkeiten der Roboterzelle erfolgt und auf welche Weise sie mit der anschließenden, roboterspezifischen Fertigungsplanung verzahnt ist. Seiteneffekte, die bei gleich zu lösenden Montageaufgaben in unterschiedlichen Ausgangssituationen auftreten können, werden ebenso in diesem Kapitel beleuchtet. Auf deren Erkennung und Berücksichtigung als wichtiges Puzzlestück einer konsistenten Planung wird detailliert eingegangen.

6 Prozessplanung

Ausgangspunkt der zweiten Phase des Planungsansatzes von $PaRTs$ – und zwar der Prozessplanung – ist ein Modell des zu fertigenden Produkts in Form einer Produktsituation, der Ziel-Produktsituation. Über die Ziel-Produktsituation sind sämtliche Einzelteile des finalen Bauteils bekannt, sowie alle Attribute, durch die der semantische Zusammenhang der Einzelteile sowie deren benötigte Verarbeitungen beschrieben sind. Im Rahmen der Prozessplanung sollen daraus mögliche Prozessabläufe abgeleitet werden, die eine Montage des Bauteils aus rein abstrakter Domänensicht – also noch ohne jegliche Einbeziehung von Robotern – darstellen.

Um Prozessabläufe zu finden, die von einer initialen Produktsituation in eine Ziel-Produktsituation überführen, greift der Planungsansatz auf das in der Forschung weit verbreitete Konzept der zustandsbasierten Suche zurück. Grundidee hierbei ist, dass aus einer Menge an möglichen Zuständen sowie Aktionen, die einen Übergang zwischen Zuständen erlauben, eine geeignete Abfolge von Aktionen gefunden wird, die von einem Start-Zustand in einen gewünschten Ziel-Zustand führt.

Im Rahmen der Prozessplanung stellen Produktsituationen die unterschiedlichen Zustände dar, Domänentasks (siehe Abbildung 4.5, Abschnitt 4.1.3) repräsentieren als „abstrakte“ Aktionen die möglichen Zustandsübergänge. Das Finden gültiger Prozessabläufe gliedert sich damit grundlegend in zwei getrennte Teile:

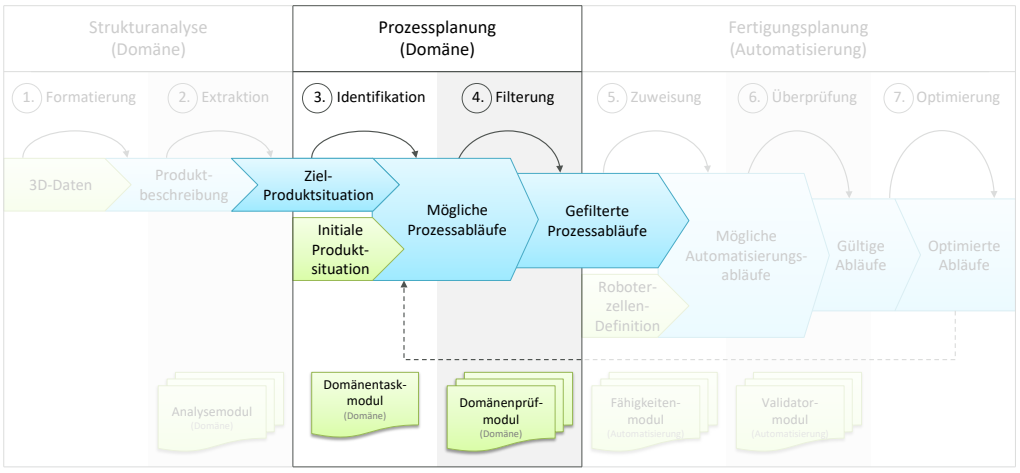


Abbildung 6.1. Einordnung der Prozessplanung in den Gesamtansatz $PaRTs$.

Planung: Die ausgehend von einer gegebenen Produktsituation möglichen Aktionen (Domänentasks) sind über Domänentaskmodule geeignet zu berechnen und zu planen. Nach der Ausführung von Domänentasks ergeben sich neue Nach-Produktsituationen, über die in fortführender Weise ein kompletter Suchraum aufgespannt werden kann. Dies wird in Abschnitt 6.1 vorgestellt.

Suche: Im geplanten Suchraum der möglichen Domänentasks ist eine konkrete Folge zu suchen, deren ausgewählte Domänentasks zum einen einen gültigen Prozessablauf der Montage bilden und zum anderen einem gewünschten Optimierungskriterium entsprechen. Spezielle Heuristiken sind dabei nötig, um die Suche im oftmals riesigen Suchraum effizient zu gestalten. Darauf geht Abschnitt 6.2 näher ein.

6.1 Grundlagen der Prozessplanung

Abbildung 6.1 ordnet die Phase der Prozessplanung in den Gesamtansatz von $PaRTs$ ein. Sie enthält den Schritt der Identifikation (3), dessen Aufgabe es ist, zu einer Ziel-Produktsituation mögliche Prozessabläufe abzuleiten. Neben der Ziel-Produktsituation wird dafür ebenso eine initiale Produktsituation benötigt, die die Situation zu Beginn der Montage und damit den Ausgangspunkt der Planung beschreibt. Somit sind sowohl Anfangs- wie auch Endzustand der Planung gegeben. Die Prozessplanung versucht einen gültigen Pfad an Domänentasks zu finden, indem sie ausgehend von der initialen Produktsituation im Zustandsraum in Richtung der Ziel-Produktsituation wandert. In der Domäne ungültige Prozessabläufe werden im Schritt der Filterung (4) aussortiert.

6.1.1 Identifikation nächster Schritte: Domänentaskmodul

Basierend auf der initialen Produktsituation kann der Zustandsraum der Suche aufgespannt werden. Über ein inkrementelles Vorgehen wird dabei auf ein vom Experten zur Verfügung gestelltes Domänentaskmodul zurückgegriffen, das ausgehend von einer gegebenen Produktsituation mögliche Domänentasks berechnet. Wie in Abbildung 4.6

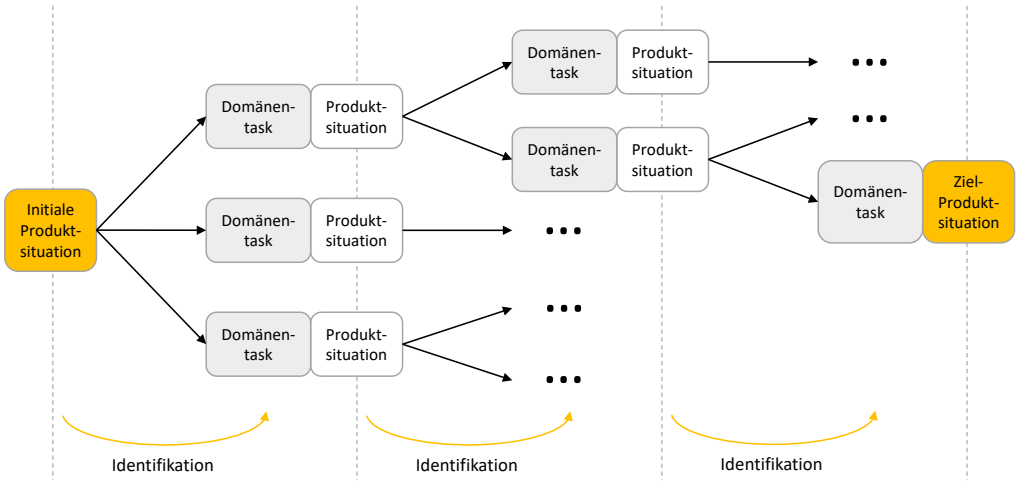


Abbildung 6.2. Beispielhafte Aufspannung des Suchraums über das Domänentaskmodul. Aus einer Produktsituation werden nachfolgende Domänentasks abgeleitet, die selbst wiederum neue Nachfolge-Produktsituationen erzeugen.

(Abschnitt 4.1.4) bereits dargestellt, erwartet ein Domänentaskmodul eine aktuelle Produktsituation sowie die angestrebte Ziel-Produktsituation als Eingabe und liefert alle daraus ableitbaren Domänentasks zurück, die in der gegebenen Produktsituation unmittelbar anwendbar sind und dem Ziel näherkommen. Dabei stellen sämtliche Berechnungsergebnisse Alternativen zueinander dar, wovon später höchstens eine im finalen Ablauf als Einzelschritt der Domäne enthalten sein wird. Aus einem Domänentask wird die nach seiner Ausführung geltende Produktsituation abgeleitet, die wiederum in gleicher Weise über das Domänentaskmodul in nachfolgende Domänentasks des nächsten Schritts überführt werden kann. Dies wiederholt sich, bis keine neue Produktsituation mehr erreicht werden kann. Abbildung 6.2 zeigt exemplarisch die Aufspannung des Suchraums durch aufeinanderfolgende Schritte der Identifikation.

Die über die Identifikation bestimmten Domänentasks referenzieren jeweils einen abstrakten Prozess, der vom Task für den Montageschritt auszuführen ist, sowie eine Menge an Attributen, die der Task bei seiner Ausführung auf- oder abbaut (siehe Abbildung 4.5, Abschnitt 4.1.3). Die vom Task beeinflussten Einzelteile sind dabei über die verknüpften Attribute ermittelbar.

Für die Bereitstellung von Domänentaskmodulen sieht der Planungsansatz zwei Optionen vor. In der ersten Variante kann ein Experte eine eigens implementierte Version des Moduls entwerfen, über die er die Logik zur Identifikation von Domänentasks manuell definiert. Auf diese Weise kann der Experte erzwingen, dass in bestimmten Produktsituationen auch (nur) bestimmte Domänentasks zur Verfügung stehen. Damit hat er eine Schnittstelle, über die er das Planungsverhalten auf der Domänenebene unmittelbar lenken kann.

Um von einer maximalen Automatisierung der Planung zu profitieren, kann über eine zweite Variante anstelle einer manuellen Implementierung alternativ eine vordefinierte,

generische Version des Domänentaskmoduls eingesetzt werden. Diese versucht, über eine Heuristik alle möglichen Domänentasks zu identifizieren, die in einer bestimmten Produktsituation möglich sind. Da bei einer Montage davon auszugehen ist, dass alle im fertigen Produkt enthaltenen Einzelteile auf irgend eine Art und Weise an einer richtigen Stelle plaziert oder montiert werden müssen, kann für jede dieser Teilmontagen zunächst ein eigener Domänentask definiert werden. Semantisch betrachtet führt diese Art von Domänentask damit den Prozess „Tue alles Notwendige, um das Einzelteil in der gegebenen Situation zu montieren“ aus. Für einen ersten Schritt der Prozessplanung liefert das automatische Domänentaskmodul alle dieser Tasks als Optionen der Planung. In den folgenden Schritten werden all diejenigen Domänentasks ausgelassen, die im aktuellen Suchpfad bereits enthalten sind. Das Domänentaskmodul ermittelt die nächstmöglichen Tasks dabei anhand der aktuellen Produktsituation. Diejenigen Einzelteile, die in der aktuellen Produktsituation bereits über Attribute referenziert sind, werden als bereits erledigt angesehen und bieten folglich keinen Domänentask an. Sind soviele Schritte (Domänentasks) absolviert wie Einzelteile im finalen Produkt enthalten sind, ist die Ziel-Produksituation erreicht.

Damit die Planung später weiß, welche Aufgaben durch den abstrakten Domänentask auszuführen sind, werden die durch den Task beeinflussten Attribute an den Task angefügt. Zunächst werden aus der Ziel-Produktion alle Attribute des Einzelteils, die vom Typ „Eigenschaft“ oder vom Typ „Fixpunkt“ sind, gefiltert und dem Domänentask zugewiesen. Dadurch werden mit dem Domänentask sämtliche Operationen am Einzelteil selbst sowie – falls vorhanden – die fixe Platzierung des Einzelteils in der Welt beschrieben. Zudem werden ebenso alle Attribute des Einzelteils vom Typ „Beziehung“ zum Task hinzugefügt, sofern diese sich ausschließlich auf Einzelteile beziehen, die bereits in einem vorigen Domänentask abgehandelt wurden. Auf diese Weise werden mit einem Domänentask all diejenigen Attribute eines Einzelteils hergestellt, die im aktuellen Stand der Montage und mit den bisher verbauten Einzelteilen umsetzbar sind.

Ob eine manuelle Implementierung des Domänentaskmoduls oder die automatisierte Variante verwendet wird, hängt von der Planungsdomäne und dem konkreten Planungsproblem ab. Die manuelle Variante benötigt im Vergleich zur automatisierten Variante einen sehr hohen initialen Aufwand für die Implementierung, kann aber gezielter bei der Auswahl gültiger und optimaler Domänentaks vorgehen und dadurch die Planungszeit möglicherweise signifikant reduzieren.

Egal ob eine eigens implementierte oder die automatisierte Variante des Domänentaskmoduls verwendet wird, nicht alle daraus generierten Pfade an Tasks sind in jedem Fall bezüglich der Domäne des Planungsproblems tatsächlich valide. Bei LEGO kann man sich eine Situation vorstellen, in der in einem Domänentask ein Stein auf einen anderen abgesetzt werden soll, dabei ein weiterer Stein, der sich dann ebenfalls unterhalb des neuen Steins befinden würde, aber noch nicht platziert ist. In Abschnitt 3.1.3, Abbildung 3.2 ist diese konkrete Problemstellung als „Ausschluss“ klassifiziert, wonach ein Stein anschließend nicht mehr (ein)gesetzt werden kann. Um die Planung frühzeitig von einem eventuell enormen und unnötigen Planungsmehraufwand zu entlasten, sollen Teiläste des Suchraums, die unweigerlich in einen Deadlock führen, möglichst früh geschlossen und nicht weiter durchsucht werden. Dafür sind im Planungsansatz Do-

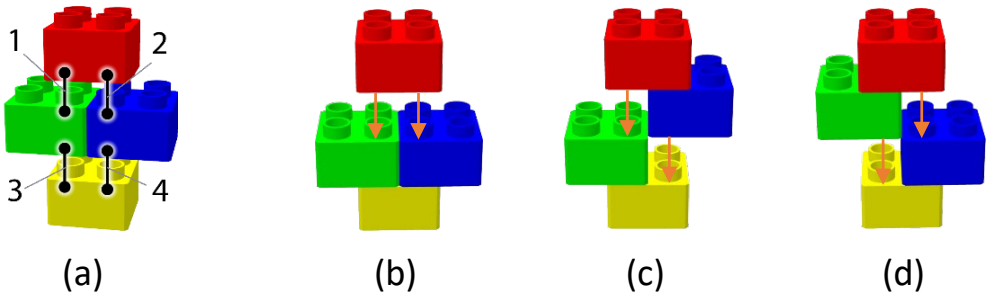


Abbildung 6.3. Attribute in einem Beispiel einer Diamantkonstruktion aus LEGO (a). Ausgehend von einer vorliegenden Produktsituation ergeben sich unterschiedliche Paare an umgesetzten Attributen beim Setzen des obersten Steins (b-d).

mänenprüfmodule (siehe Abbildung 4.6, Abschnitt 4.1.4) vorgesehen, die von Experten gezielt für die jeweilige Domäne entworfen werden und die Integrität von Produktsituationen sicherstellen können. Beliebig viele dieser Module können dem Planungsansatz hinzugefügt werden. Jedes Modul untersucht in jedem Schritt der Prozessplanung die nächsten vorgesehenen Tasks (beziehungsweise deren resultierende Produktsituationen) auf domänenspezifische Kriterien und kann diese im Bedarfsfall verwerfen.

6.1.2 Situationsbedingte Task-Ausführung: Kollateraleffekte

Für die erfolgreiche Montage eines Bauteils ist ein durchgängig geplanter Prozessablauf erforderlich. Dabei sind auf Domänenebene die einzelnen Schritte des Ablaufs Domänentasks, die jeweils ihre Auswirkungen über eine Menge an geänderten Attributen beschreiben. Ausgehend von einer Produktsituation, die vor Ausführung des Tasks galt, lässt sich daraus eine resultierende Produktsituation nach Ausführung des Schritts ableiten. Doch die resultierende Produktsituation muss – besonders wenn die Generierung der Domänentasks über ein manuell implementiertes Domänentaskmodul erfolgte – nicht in allen Fällen den tatsächlichen Änderungen des Montageschritts entsprechen. Als Beispiel illustriert Abbildung 6.3 (a) in der Domäne LEGO die Struktur eines aus vier Steinen zusammengesetzten Diamanten, beschrieben durch vier Attribute. In den drei Beispielen (b) bis (d) soll jeweils der oberste Stein abgesetzt werden, wobei sich die vorherrschende Ausgangs-Produktsituation in jeder Variante unterscheidet. Beispielsweise sind in (b) die untersten drei Steine bereits zusammengesetzt, Attribute 3 und 4 wurden also bereits zuvor aufgebaut. Die Prozessplanung könnte nun als nächstes einen Domänentask wählen, der Attribut 1 aufbaut – was einem Absetzen des roten Steins auf den grünen Stein entspricht. Doch da der blaue Stein bereits auf dem gelben, untersten Stein sitzt (Attribut 4 ist aufgebaut), wird mit dem Setzen des roten Steins ebenso Attribut 2 zwangsläufig mit aufgebaut. Diese Eigenschaft, wonach die Herstellung eines Attributs in einer gegebenen Produktsituation das gleichzeitige Aufbauen eines oder mehrerer anderen Attribute bedingt, wird **Kollateraleffekt** genannt. Beispiele (c) und (d) zeigen weitere Varianten auf, in denen dieser Effekt sichtbar wird. In beiden Szenarien ist die Ausgangssituation so gegeben, dass jeweils zwei Steine miteinander verbunden sind. Wird Beispiel (b) mit (c) verglichen, so bewirkt ein Aufbauen von Attri-

Kollateraleffekt

but 1 jeweils einen anderen Kollateraleffekt: Attribut 2 in Beispiel (b) und Attribut 4 in Beispiel (c). Es ist also festzustellen, dass das Aufbauen eines Attributs abhängig von der Ausgangs-Produktsituation unterschiedliche Kollateraleffekte bewirken kann. Analog gilt für einen Task, der eine gegebene Menge an Attributen umsetzt, dass seine Ausführung abhängig von der zugrundeliegenden Ausgangs-Produktsituation unterschiedliche Kollateraleffekte bewirken kann. Definition 6.1 beschreibt den Kollateraleffekt formal:

Definition 6.1. *Kollateraleffekt*
Für einen Task $t \in \mathcal{T}$ und eine Situation $s \in \mathcal{S}$ seien $Effekt^+(t,s)$ und $Effekt^-(t,s)$ die in der Situation s durch den Task t tatsächlich auf- bzw. abgebauten Attribute. Für alle $t \in \mathcal{T}$ und $s \in \mathcal{S}$ gelte immer $Effekt^+(t,s) \cap Effekt^-(t,s) = \emptyset$, $ExplEffekt^+(t) \subseteq Effekt^+(t,s)$ und $ExplEffekt^-(t) \subseteq Effekt^-(t,s)$,

Damit ist das Paar

$$Effekt(t,s) = (Effekt^+(t,s), Effekt^-(t,s))$$

die vollständige Beschreibung der tatsächlichen Attributänderungen von t in s. Die resultierende Situation $nach(t,s) \in \mathcal{S}$ aus der Anwendung eines Tasks t auf s ist

$$nach(t,s) = (s \setminus Effekt^-(t,s)) \cup Effekt^+(t,s)\ .$$

Weiterhin beschreibt das Paar

$$KollEffekt(t,s) = (KollEffekt^+(t,s), KollEffekt^-(t,s))$$

*mit $KollEffekt^+(t,s) = Effekt^+(t,s) \setminus ExplEffekt^+(t)$ und $KollEffekt^-(t,s) = Effekt^-(t,s) \setminus ExplEffekt^-(t)$ diejenigen Attribute, die durch den Task in der Situation **kollateral** auf- bzw. abgebaut werden.*

Ein Attribut, das von einem Task in einer gegebenen Ausgangs-Produktsituation zwar auf- bzw. abgebaut, nicht aber von diesem als expliziter Effekt ausgewiesen wird, ist demnach über den Kollateraleffekt des Tasks beschrieben.

6.1.3 Automatische Extraktion von Kollateraleffekten

Für eine automatische Planung des Montageablaufs, die auf Basis des bekannten Wissens die weiteren Schritte in Form von Tasks berechnet, ist eine möglichst treue Erfassung der realistischen Effekte der Aktionen unmittelbar ausschlaggebend für den Erfolg der Planung und die Qualität des Ergebnisses. Denn nur mit einer vollständigen Beschreibung, welche Attribute von jedem Task auf- bzw. abgebaut werden, ist die korrekte Bestimmung von Nachfolgeaktionen und damit das Finden eines konsistenten Prozessablaufs möglich. Da die tatsächlich von einem Task auf- bzw. abgebauten Attribute von der zuvor geltenden Situation abhängen, stellt sich daher unweigerlich die Frage, wie die jeweils auftretenden Kollateraleffekte und damit die tatsächlich stattfindenden Attributänderungen in den einzelnen Montageschritten ermittelt werden können. Dabei ist es nicht unbedingt zielführend, die Bestimmung dem Experten bei der Entwicklung von Modulen zu überlassen. Denn zum einen ist es zeitaufwendig und komplex, alle

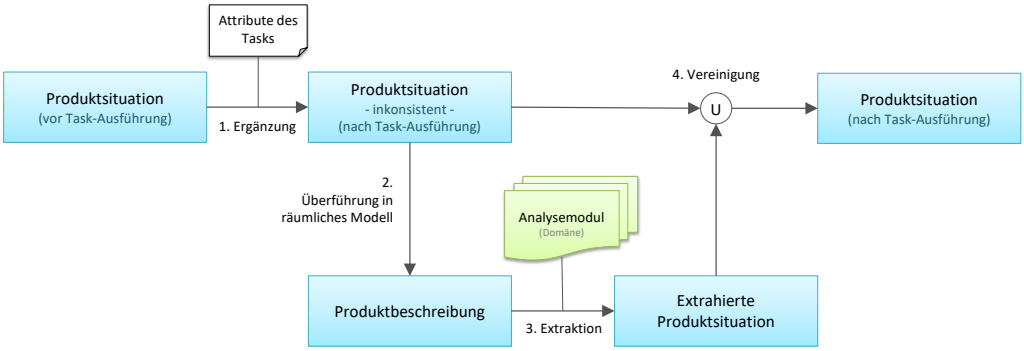


Abbildung 6.4. Verwendung des Ansatzes der Strukturanalyse zur Erkennung von Kollateraleffekten eines Tasks. Aus einer gegebener Produktsituation und einem auszuführenden Task wird eine konsistente Produktsituation ermittelt, die nach Ausführung des Tasks gilt.

Möglichkeiten der tatsächlichen Effekte eines Tasks abhängig von den verschiedenen Situationen zu berücksichtigen, was die Verwendung des Planungsansatzes deutlich erschwert. Zum anderen würde eine solche Voraussetzung einer vollständigen Beschreibung durch den Experten die Fehleranfälligkeit deutlich erhöhen. Dies wäre bereits der Fall, wenn beispielsweise auch nur ein Kollateraleffekt in einer bestimmten Situation falsch beschrieben wäre. Stattdessen ist es wünschenswert, dass der Experte lediglich den unmittelbaren Effekt von Tasks beschreibt und der Planungsansatz abhängig von der konkreten Situation, in der die Tasks verwendet werden, das Ergebnis um entsprechende Kollateraleffekte automatisch ergänzt.

Kollateraleffekt-Extraktion

Der Planungsansatz unterstützt eine automatische Bestimmung von Kollateraleffekten (**Kollateraleffekt-Extraktion**) ohne zusätzlichen Mehraufwand für den Experten auf Basis bereits bestehender Module. Dabei wird ausgenützt, dass die in Phase 1 des Planungsansatzes (Strukturanalyse, Kapitel 5) eingebundenen Analysemodule räumliche Modelle analysieren und darin implizit enthaltene Attribute extrahieren können. Abbildung 6.4 beschreibt die Herangehensweise, mittels derer die Kollateraleffekt-Bestimmung bei der Planung erfolgt. Gegeben sei eine Produktsituation, die einen derzeitigen Montagezustand der Planung beschreibt. In dieser Produktsituation soll ein gegebener Task ausgeführt und die Attribute, die er enthält, umgesetzt werden. Als Illustration kann hier Beispiel (b) aus Abbildung 6.3 dienen, wobei mit einem Domänentask das Attribut 1 zwischen rotem und grünem Stein umgesetzt werden soll.

Zunächst wird die Menge der vom Task umzusetzenden Attribute (im Beispiel also Attribut 1) in die Produktsituation aufgenommen. Diese neue Produktsituation ist gegebenenfalls noch nicht konsistent, möglicherweise fehlen noch weitere Attribute, die den Kollateraleffekt ergänzen. Trotzdem werden durch die Attribute semantische wie auch geometrische Informationen zum bestimmten Montagezustand beschrieben. In einem zweiten Schritt werden alle geometrischen Informationen der Produktsituation zusammengetragen und daraus ein räumliches Modell erzeugt, wie es durch die einheitliche Produktbeschreibung ausgedrückt werden kann. Semantische Informationen der Produktsituation werden dabei absichtlich verworfen. Am LEGO-Beispiel führt dies zu einem räumlichen Modell, in dem jeder Stein seinen vorgesehenen Platz in der finalen

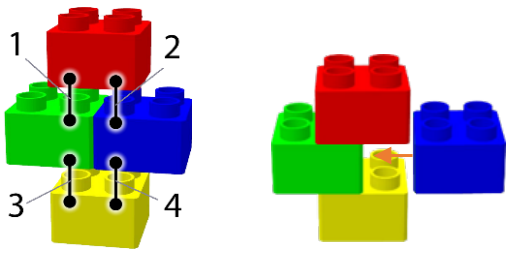


Abbildung 6.5. Deadlocksituation bei der Montageplanung einer Diamantstruktur aus LEGO aufgrund der Inkompatibilität der zur Umsetzung zweier Attribute (2 und 4) notwendigen Prozesse.

Diamantstruktur einnimmt. In Schritt 3 wird das Konzept der modularen Strukturanalyse verwendet, um das räumliche Modell im Hinblick auf implizit enthaltene Informationen zu analysieren und entsprechende Attribute zu extrahieren. Im Beispiel des LEGO-Diamanten werden dabei alle vier Attribute extrahiert, darunter auch das kollateral umgesetzte Attribut 4. Schritt 4 erstellt die Vereinigung aus der ursprünglichen und der neu erhaltenen Produktsituation. Das Resultat ist eine konsistente Produktsituation, die den Effekt des ausgeführten Tasks sowie alle dabei auftretenden Kollateraleffekte beinhaltet. Mit diesem Ergebnis als neuen Ist-Zustand der Montage kann die Planung darauf aufbauend fortgeführt werden.

6.1.4 Ausführungsäquivalenz als Kriterium für Prozess-Kompatibilität von Tasks

Jeder Task, der von der Planung als nächster Montageschritt in einer bestimmten Situation ausgewählt wurde, beschreibt eine Menge an Attributen, die bei seiner Ausführung in dieser Situation umgesetzt werden. Sind insbesondere nicht alle Attribute, die bei einer realen Ausführung auftreten würden, dabei eindeutig über den Task spezifiziert, so stellen diese einen kollateral ausgeführten, kontextabhängigen Effekt des Tasks dar. Die spezifizierten wie auch die kollateral auftretenden Attribute enthalten eine semantische Information darüber, mit welcher Art von Prozess sie hergestellt werden können. Dabei sind seitens des Tasks jedoch nur Prozessinformationen derjenigen Attribute berücksichtigt, die er als seine unmittelbaren Effekte benennt. Prozessinformationen kollateral umgesetzter Attribute hingegen bleiben bei der Auswahl des Tasks unberücksichtigt. In einem unerwünschten Fall mag es nun vorkommen, dass der von einem unberücksichtigten Attribut geforderte Prozess mit dem aktuell ausgeführten Task nicht kompatibel ist. Eine reale Ausführung des Tasks ist demnach nicht möglich, was zu einem ungültigen Planungsergebnis führt.

Abbildung 6.5 greift das Beispiel der Diamantstruktur aus LEGO wieder auf. In der gegebenen Situation sind Atribute 1 und 3 bereits umgesetzt. Als nächsten Schritt liefere der Planer einen Task zur Umsetzung von Attribut 4. Über die Extraktion von Kollateraleffekten wird erkannt, dass auch Attribut 2 mit der Ausführung des Tasks umgesetzt würde. Doch augenscheinlich ist dies in Kombination nicht möglich, da der blaue Stein durch den vorliegenden Ausschluss nicht mehr einsetzbar ist. Tatsächlich sind

die Prozesse, die für die Umsetzung beider Attribute erforderlich sind, nicht kompatibel. Beide Attribute beschreiben einen Prozess, wonach jeweils zwei Steine aufeinander zubewegt werden bis diese schlüssig aufeinander stecken. Da die C-förmige Struktur über eine bereits umgesetzte Kette an Attributen den unteren und den oberen Stein über den grünen Stein fest miteinander verbindet, sind aus Sicht des blauen Steins zwei Prozesse notwendig: Während Attribut 4 ein nach-unten-Bewegen auf den gelben Stein erfordert, erzwingt Attribut 2 ein nach-oben-Bewegen des blauen Steins an den roten Stein. Da die Prozesse zur Umsetzung der beiden Attribute notwendig aber widersprüchlich sind, ist die Herstellung des Produkts aus der gegebenen Produktsituation nicht möglich.

Um gültige Abläufe für die Fertigung des Produkts zu erhalten, ist sicherzustellen, dass in jedem Fertigungsschritt alle umzusetzenden Attribute desselben Einzelteils zueinander kompatible Prozesse aufweisen. Da erst der Task einen Prozess mit Attributen in Beziehung bringt, gilt gleichermaßen, dass auch alle Tasks zur Umsetzung der einzelnen Attribute kompatibel zueinander sein müssen. Diese Kompatibilität auf Tasks und Attributen wird **Ausführungsäquivalenz** genannt. Eine Menge, die zu einem Task ebenso alle zu diesem ausführungsäquivalenten Tasks enthält, heißt **ausführungsvollständig**. Die formale Definition der Ausführungsäquivalenz und der Ausführungsvollständigkeit ist in Definition 6.2 gegeben:

Ausführungsäquivalenz

Ausführungsvollständigkeit

Definition 6.2. *Ausführungsäquivalenz*
*Auf der Menge der Paare von Prozessen gebe es ein Prädikat $komp : \mathcal{P} \times \mathcal{P} \to \mathbb{B}$, das ausdrückt, dass zwei Prozesse **kompatibel** sind.*

*Bezüglich einer Situation $s \in \mathcal{S}$ sind zwei technische Tasks $t_1, t_2 \in \mathcal{T}$ **ausführungsäquivalent**, i. Z. $t_1 \sim_s t_2$, wenn sie in dieselbe Situation überführen, also $nach(t_1, s) = nach(t_2, s)$, und deren Prozesse kompatibel sind: $komp(process(t_1), process(t_2))$.*

*Eine Menge $T \subseteq \mathcal{T}$ heißt **ausführungsvollständig**, wenn alle Paare $(t_1, t_2) \in T \times T$ ausführungsäquivalent sind, i. Z. $t_1 \sim_S t_2$, und kein weiteres $t_3 \in \mathcal{T} \setminus T$ existiert, das für ein $t \in T$ ausführungsäquivalent $t_3 \sim_S t$ ist.*

Ausführungsäquivalente Tasks setzen also in einer gegebenen Produktsituation das gleiche Set an Attributen um (tatsächlicher Effekt) und es gibt keinen anderen Task, der dies ebenso tun würde. Wird vom Planer für den nächsten Schritt der Montage ein bestimmter Task gewählt, so gelten im Folgeschritt all seine ausführungsvollständigen Tasks als ebenso erledigt.

Mit der Erkennung von Ausführungsäquivalenzen kann die Planung nun sicherstellen, dass die im Ergebnis der Montageplanung enthaltenen Tasks real umsetzbar und nicht etwa durch inkompatible Prozesse von Kollateraleffekten als ungültig anzusehen sind.

6.2 Planung auf Domänenebene

Mithilfe der Domänentaskmodule können für einen beliebigen Planungszustand mögliche Nachfolge-Aktionen auf Domänenebene berechnet werden. Die Komplexität beim Aufbauen und Durchsuchen des gesamten Zustandsraums wächst dabei exponentiell mit der Komplexität des zu planenden Montageproblems. Die Anwendung klassischer Suchalgorithmen (Tiefensuche, Breitensuche, A*, … [75]) kommt hier schnell an seine

Grenzen und verliert sich in unrealistisch langen Planungszeiten. Der Planungsansatz verwendet daher eine auf die Montageplanung zugeschnittene Strategie der Planung und trennt diese zudem in zwei Schichten auf, eine abstrakte Makroschritt-Planung auf Domänenebene und eine detaillierte Mikroschritt-Planung auf Automatisierungsebene. Die Verzahnung beider Ebenen ist in Abschnitt 6.2.1 beschrieben. Auf die Herausforderungen und die konkrete Strategie der Planung auf Domänenebene wird in Abschnitt 6.2.2 und Abschnitt 6.2.3 eingegangen.

6.2.1 Verzahnung zwischen Domäne und Automatisierung

Damit die Prozessplanung auf Domänenebene nicht zuerst den vollständigen Zustandsraum aufbauen muss bevor es zur nachgelagerten Fertigungsplanung mit Robotern weitergeht, erlaubt eine Schnittstelle zwischen den beiden Phasen eine vorgegriffene Fertigungsplanung einzelner Zwischenergebnisse der Prozessplanung. Die Schnittstelle ist als Blackbox konzipiert (von der dahinter liegenden Funktionsweise wird also völlig abstrahiert), die für einen Domänentask eine konkrete Detailplanung auf Ebene der Automatisierung erlaubt. Bei Verwendung der Schnittstelle erhält die Prozessplanung die gefundene Lösung, ist aber gänzlich abgeschottet von der internen Funktionsweise der Fertigungsplanung. Da die Fertigungsplanung Aktionen mit Robotern plant, um die Einzelteile des Bauteils zu manipulieren, ist hier neben dem Wissen über einen Produkt-Zustand ebenso die Kenntnis über den Zustand der Roboterzelle erforderlich. Daher muss sich auch die Prozessplanung sämtliche Informationen über die Roboterzelle merken und diese bei anschließender Verwendung der Schnittstelle der Fertigungsplanung übergeben. Als Beschreibung hierfür wird das Konzept der Planungssituation (siehe Abschnitt 4.1.2) verwendet. Sie erweitert die Produktsituation und enthält neben den Attributen der Domäne zusätzlich sämtliche Attribute der Roboter.

Abbildung 6.6 erläutert die Interaktion zwischen Prozess- und Fertigungsplanung. Angestoßen wird die Planung auf Prozessebene mit der Angabe der initialen Produktsituation p_{init} sowie der Ziel-Produktsituation p_{goal}. Die initiale Produktsituation, die den Zustand des Bauteils zu Beginn der Fertigung beschreibt, wird über eine Schnittstelle der Fertigungsplanung (siehe Abbildung 6.6) in eine initiale Planungssituation s_{init} überführt, die um Informationen aus der Roboterzellen-Definition und Angaben über initiale Roboterposen erweitert ist. s_{init} wird in die Menge M der bisher bekannten Situationen aufgenommen. Auf Prozessebene wird nun eine der bereits bekannten Planungssituationen aus M (zu Beginn also s_{init}) als aktiver Zustand ausgewählt und zu diesem ein möglicher nächster Domänentask als nächster möglicher Schritt der Fertigung selektiert. Die Planungssituation sowie der zu planende Domänentask werden der Fertigungsplanung übergeben. Als Ergebnis der dort stattfindenden internen Planung wird ein Iterator it zurückgegeben, über den sämtliche Montageprogramme – sofern diese existieren – bereitgestellt werden, die als Lösungsmöglichkeiten für den Domänentask infrage kommen. Jedes dieser möglichen Montageprogramme t wird in einer Schleife einzeln betrachtet, und eine resultierende Planungssituation s_{res} wird abgeleitet, die nach einer Ausführung von t in s gelten würde. Diese wird, sofern kein Validator- oder Domänenprüfmodul sie als ungültig identifiziert, ebenfalls in die Menge bekannter Planungssituationen aufgenommen und steht der Montageplanung wiederum

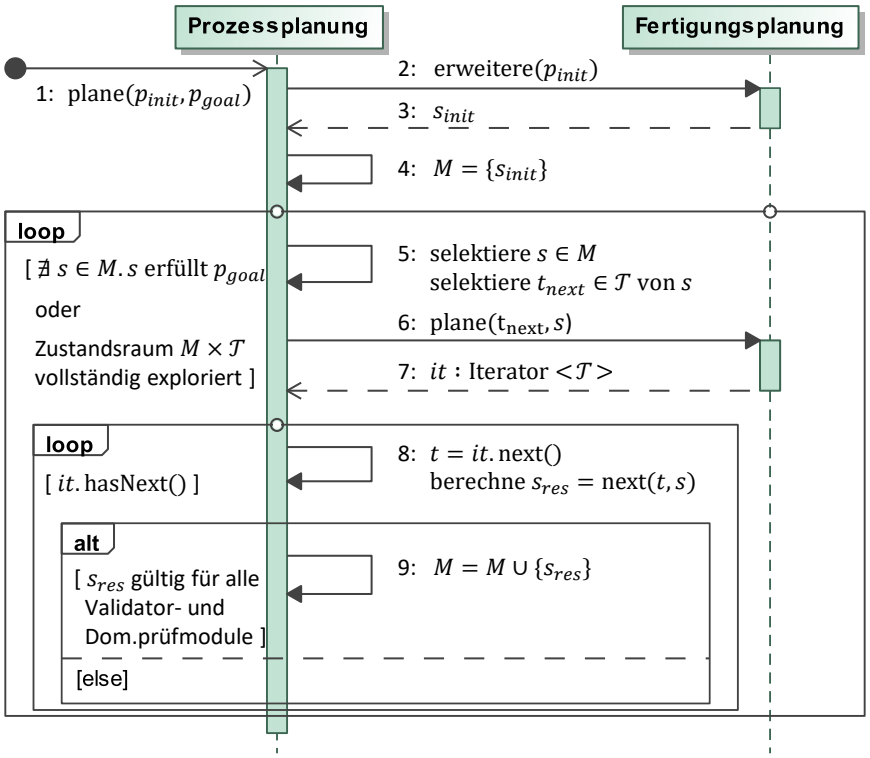
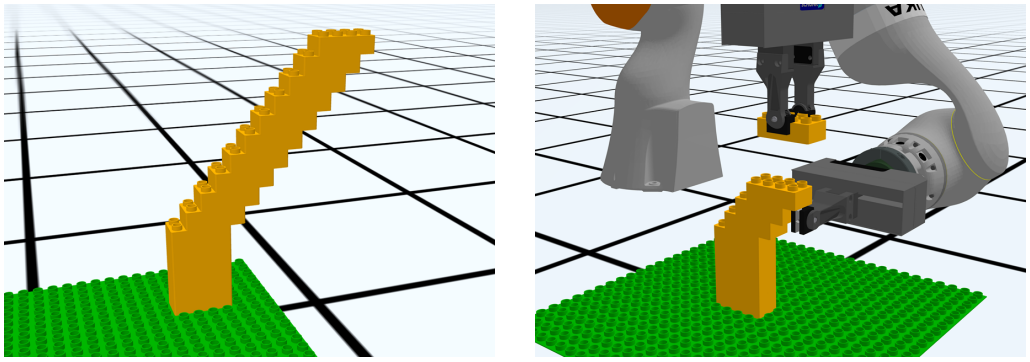


Abbildung 6.6. Interaktion zwischen Prozess- und Fertigungsplanung.

als Startzustand eines nächsten Schritts zur Verfügung. Solange keiner der bisher bekannten Planungssituationen aus M die Ziel-Produktsituation erfüllt (d. h. alle Attribute der finalen Produktbeschreibung enthält), wird die Suche fortgesetzt. Die Suche gilt auch dann als beendet, wenn der komplette Suchraum exploriert wurde, ohne dabei eine gültige Lösung zu finden.

Der Schnittstellenaufruf zur Planung eines Domänentasks (Schritt 6) liefert einen Iterator zurück, der alle möglichen Montageprogramme der Optimalität nach sortiert verfügbar macht. Man möchte nun annehmen, dass nur das erste Montageprogramm des Iterators für die weitere Planung ausreichen sollte – es handelt sich dabei ja um die optimale Lösung. Doch ist durch eine Optimalität der Teilprogramme nicht zwangsläufig gegeben, dass auch die Fertigungsplanung damit ein optimales Planungsergebnis für die gesamte Montage erzielen kann. Denn ein Montageprogramm, mag es aus lokaler Sicht auch optimal sein, kann im Hinblick auf nachfolgende Teilprogramme ein wesentlich ineffizienteres globales Gesamtergebnis verursachen als ein vielleicht weniger optimales Montageprogramm. Es ist sogar möglich, dass die optimale Automatisierungslösung des Domänentasks eine ausweglose Situation hervorruft und dadurch das Finden einer Lösung gänzlich verhindert.

Die in Abbildung 6.7 dargestellte Struktur zeigt das Problem der lokalen und globalen Optimalität an einem anschaulichen Beispiel. Zu errichten ist eine Treppenkonstruktion, die ab einer gewissen Höhe nicht mehr selbständig stehen bleibt und zudem beim



(a) Zu fertigende Treppe aus LEGO **(b)** Kooperation durch Stützen und Absetzen

Abbildung 6.7. Kooperativer Aufbau einer Treppe aus LEGO als Beispiel für die Planung eines Zusammenbaus, die bei Auswahl der Einzelschritte nach einem lokalen Optimalitätskriterium fehlschlagen kann.

Absetzen weiterer Steine zusammenbrechen würde (siehe Problemstellung der Statik, beschrieben in Abschnitt 3.1.3, Abbildung 3.2). Deshalb werden zwei Roboter zur Fertigung eingesetzt, die die Treppe in Kooperation zusammenbauen können. Während ein Roboter die bereits hergestellte Teil-Konstruktion unterstützt, setzt der andere den nächsten Stein (siehe Abbildung 6.7b). Anschließend kann dieser den eben von ihm selbst abgesetzten Stein unterstützen und ermöglicht damit dem anderen Roboter, seine Unterstützung aufzulösen. Der andere Roboter ist damit wieder frei und kann den nächsten Stein holen und absetzen. Daraus ergibt sich eine jeweils alternierende Rolle der Roboter für das Unterstützen und das Absetzen beim Bau der Treppe.

Kann aufgrund der Position der Roboter in der Zelle und ihres dadurch definierten Arbeitsraums die Ablageposition des letzten, obersten Steins nur von einem der beiden Roboter erreicht werden, so stehen die Chancen nun $50\,\%$ zu $50\,\%$, dass die bisherige Montage in der entsprechend richtigen Konfiguration enden wird. Im ungünstigen Fall ist derjenige Roboter, der die Ablageposition erreichen kann, durch das Unterstützen belegt, wohingegen der freie Roboter den letzten Stein nicht setzen kann. Die Planung führt zu keinem gültigen Ergebnis. Für die Entscheidung, welcher Roboter für das Setzen des letzten Steins zur Verfügung steht, ist im vorgenannten Beispiel die Wahl des Roboters für die erste Unterstützung zu Beginn der Planung ausschlaggebend.

Betrachten wir das Szenario etwas genauer: Der erste Stein, der einer Unterstützung bedarf, wird auf Prozessplanungsebene als Domänentask ausgewählt („Setzen des Steins an die vorgesehene Stelle“). Daran geknüpft ist das Umsetzen des Attributs zwischen dem neuen und dem darunterliegenden Stein der Treppe. Für den Domänentask wird im Rahmen der Fertigungsplanung ein Montageprogramm geplant, das mit einem Roboter den darunterliegenden Stein unterstützt und mit dem anderen den neuen Stein absetzt, sodass das Attribut hergestellt ist. Bei der Aufgabenzuordnung zu den Robotern ist die Planung zunächst frei. Nach Untersuchung und Vergleich der verschiedenen Lösungen wird final allerdings diejenige Lösung als optimales Ergebnis zurückgegeben, die die kürzeren Fahrtwege der Roboter verursacht oder prozesssichere Greifaktionen erlaubt.

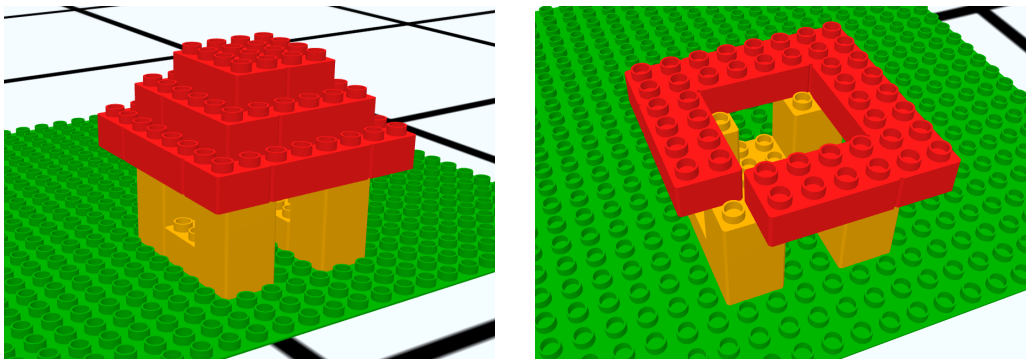
Führt diese Auswahl zu der oben beschriebenen ungünstigen Situation, wonach der letzte Stein nicht gesetzt werden kann, wird die Planung auf Prozessebene im weiteren Verlauf keine Lösung finden können, wenn sie ausschließlich die eine optimale Lösung der Fertigungsplanung berücksichtigt.

Aus diesem Grund muss die Fertigungsplanung eine Möglichkeit anbieten, die interne Planung zu einem Domänentask fortzuführen und eine alternative Lösung bereitzustellen. Sie liefert daher immer mehrere Lösungen zurück, die „lazy“ – also nach Bedarf – evaluiert werden können. Führt eine gewählte Lösung auf Ebene der Prozessplanung nicht zum Erfolg, kann die nächstbeste Lösung abgerufen werden. Softwaretechnisch ist dies über ein Iteratorenkonzept gelöst. Ein Aufruf der in Schritt 6 verwendeten Schnittstelle der Fertigungsplanung liefert einen Iterator zurück, der zum einen über die Funktion $hasNext()$ über das Vorhandensein einer ersten und anschließend weiteren Lösung informiert, und zum anderen über einen Aufruf der Funktion $next()$ ein alternatives Montageprogramm, der Optimalität nach sortiert, bereitstellt. Dies ermöglicht der Prozessplanung, im beschriebenen Fall des Treppenbaus doch noch eine gültige Lösung zu finden und zu planen.

6.2.2 Herausforderungen der zustandsbasierten Planung

Mit der Schnittstelle zur Fertigungsplanung als Blackbox können über die Prozessplanung komplette Abläufe an Domänentasks erzeugt werden, für die jeweils ein konkretes Ausführungsprogramm hinterlegt ist. Wie erfolgreich und schnell die Planung einer gültigen Lösung dabei abläuft, entscheidet maßgeblich die Strategie, wie der Suchraum möglicher Aktionen durchwandert wird. Im Sequenzdiagramm in Abbildung 6.6 passiert dies in Schritt 5 über die Selektion der Planungssituation und des Domänentasks. Die Wahl beider Elemente stellt für die Planung auf Domänenebene zwei Freiheitsgrade dar, die von der Strategie unmittelbar genutzt und beeinflusst werden können: 1. Welcher bereits bekannte Zustand wird zur weiteren Untersuchung aufgegriffen? D. h. ausgehend von welcher Planungssituation soll weiter geplant werden? 2. Welche Aktion wird gewählt, um in einen Nachfolgezustand zu gelangen? D. h. welches der von der Fertigungsplanung bereitstellbaren Montageprogramme wird für den nächsten Schritt des Fertigungsablaufs gewählt?

Verschiedene Strategien gelangen dabei unterschiedlich schnell an eine gültige Lösung. Die Geschwindigkeit wird dabei vorrangig durch Problemstellungen des Planungsproblems bestimmt, die für die verschiedenen Strategien unterschiedlich schwere Herausforderungen darstellen. Abbildung 6.8 zeigt eine solche Problemstellung über ein Beispiel aus LEGO. Zu fertigen ist ein Haus bestehend aus einer Unterkonstruktion mit Eingang und Fenstern, sowie einem pyramidenförmigen Dach. In Abbildung 6.8b ist ein konkreter Montageschritt dargestellt, in dem die Unterkonstruktion gänzlich hergestellt und die erste Dachreihe fast komplett aufgesteckt ist. Lediglich ein Stein am Eck der Dachebene ist noch zu setzen. Für einen nächsten Schritt der Montage kann der Planer unterschiedliche Möglichkeiten an Domänentasks wählen: Entweder wird der fehlende Eckstein gesetzt, oder es wird mit einem Stein der nächsten Dachebene fortgesetzt. Im darauffolgenden Schritt sind eben dieselben Aktionen nochmals möglich, mit



(a) Zu fertigendes Haus aus LEGO **(b)** Montagezustand mit fehlendem Eckstein

Abbildung 6.8. Ein Haus aus LEGO als Beispiel für die Schwierigkeit einer frühzeitigen Deadlockerkennung. Der Umgang damit und dem dadurch bedingten ungültigen Planungsraum stellt eine Herausforderung an die Strategie des Planers dar.

Ausnahme der zuvor gewählten – eventuell kommen sogar noch neue Domänentasks als Optionen hinzu. Die eigentliche Problemstellung des Beispiels liegt darin, dass für den Domänentask des Ecksteins in der aktuellen sowie in allen späteren Situationen der Planung aufgrund der zur Verfügung stehenden Roboterfähigkeiten keine konkret umsetzbare Lösung gefunden werden kann. Denn die mit Parallelgreifern ausgestatteten Roboter würden beim Absetzen des Steins mit ihren Greiffingern an einem der beiden Nachbarsteine anecken. Dabei handelt es sich um die in Abschnitt 3.1.3, Abbildung 3.2 beschriebene Problemstellung der Kollision, die in der gegebenen Situation ebenso zu einem Ausschluss auf Planungsebene führt.

Es drängt sich der Gedanke auf, es handle sich hierbei um ein domänenspezifisches Problem, das über ein entsprechendes Domänenprüfmodul abgefangen werden sollte, das solche Situationen verwirft. Stellt man sich allerdings eine Roboterzelle vor, in der eine Art Saugnapfgreifer den Stein von oben greifend in die Lücke plazieren kann, wäre eine erfolgreiche Planung der Montage aus der gegebenen Situation durchaus möglich, die Situation ist also aus Domänensicht nicht als ungültig anzusehen. Der Ausschluss des Ecksteins repräsentiert damit kein reines Domänenproblem. Würde ein Domänenprüfmodul Situationen mit fehlendem Eckstein verwerfen, blieben beim Einsatz eines Saugnapfgreifers gültige Programme unberücksichtigt. Eine Berücksichtigung der zur Verfügung stehenden Roboterfähigkeiten innerhalb des Domänenprüfmoduls ist ebenso kein empfehlenswertes Vorgehen, da dies dem Trennungskonzept des Planungsansatzes widersprechen würde. Stattdessen ist diese Problemstellung von der Strategie des Planers zu meistern.

Klassische Vertreter der zustandsbasierten Suchstrategien sind die Tiefensuche, die Breitensuche und die Suche optimaler Pfade über A*. Ihre Vor- wie auch Nachteile sind in Tabelle 6.1 gegenübergestellt. Der Umgang der verschiedenen Strategien mit der Problemstellung des LEGO-Ecksteins ist im Folgenden dargelegt.

Tiefensuche

Tiefensuche: Ausgehend von einem initialen Zustand wird eine der möglichen Aktionen gewählt und dessen Nachfolgezustand ermittelt. Von diesem ausgehend wird

Tabelle 6.1. Vor- und Nachteile klassischer Suchstrategien bei der zustandsbasierten Planung. *PP* ist die Abkürzung für Planungsproblem. Die Breitensuche liefert den kürzesten Pfad, was jedoch nicht unbedingt auch das optimale Ergebnis darstellt. Die Performanz von A* bei großen Planungsproblemen wird durch die eingesetzte Heuristik bestimmt.

	Komplexität des PP	Tiefensuche	Breitensuche	A*
Performanz bei PP ohne Deadlocks	klein	+	+	+
	groß	+	−	?
Performanz bei PP mit Deadlocks	klein	+	+	+
	groß	−	−	?
Optimales Ergebnis	klein / groß	−	?	+

unmittelbar eine nächste mögliche Aktion gewählt. Das Vorgehen wiederholt sich, bis ein gewünschter Zielzustand erreicht ist oder keine Nachfolgeaktion mehr möglich ist. Im letzteren Fall wird zum vorigen Zustand zurückgekehrt und eine alternative Aktion gewählt. Angewandt auf das Beispiel mit LEGO kann der Eckstein zwar als erster Domänentask selektiert werden. Da dieser aber nicht lösbar ist, wird mit einer alternativen Route fortgefahren, wonach Steine der nächsten Dachebene gesetzt werden. Die Suche wird fortgesetzt, bis der ganze Teilbaum, in dem der Eckstein nicht gesetzt ist, komplett durchsucht ist. Beim Haus sind dies 21 unnötig zu untersuchende Domänentasks, für die jeweils zudem eine Planung auf Automatisierungsebene erfolgt. In Kombination mit allen möglichen Pfaden, die bei der Montageplanung für das Haus in ein derartiges Ecksteinproblem geraten, wächst die Anzahl der unnötig zu untersuchenden Zustände exponentiell an. Je nach „Pech“ bei der Wahl nächster Aktionen verliert sich die Tiefensuche ständig in bereits zum Deadlock verurteilten Teilbäumen. Grundsätzlich ist die Tiefensuche bei Planungsproblemen, die keine Deadlocks besitzen, eine sehr schnelle Strategie. Im Fall von Deadlocks kann diese Suchstrategie allerdings sehr ineffizient sein. Zudem garantiert sie – sofern sie nicht den gesamten Zustandsraum durchwandert – nicht das Finden eines optimalen Ergebnisses.

Breitensuche

Breitensuche: Ausgehend von einer bestimmten Tiefe des Suchbaums (beginnend mit 1) werden für alle möglichen Pfade dieser Länge alle möglichen Aktionen und deren Nachfolgezustände untersucht. Erst wenn der Suchbaum bis zu dieser Tiefe komplett durchsucht ist, werden alle Möglichkeiten des nächsten Schritts (Tiefe + 1) berechnet. Wird hierbei ein Ergebnis gefunden, so handelt es sich dabei um eine Lösung mit der garantiert geringsten Anzahl an benötigten Aktionen. Da das kürzeste Programm aber nicht zugleich das optimale Programm sein muss, ist dieser Vorteil für die Montageplanung nur von nachrangigem Wert. Im Beispiel des Hauses ist die Anzahl an kombinatorischen Möglichkeiten, in denen die Steine in unterschiedlichen Reihenfolgen gesetzt werden, sehr groß. Mit einem derart hohen Verzweigungsgrad des Suchbaums werden bei einer Breitensuche überwiegend Pfade untersucht, die für die irgendwann

zu findende Lösung keine Rolle spielen. Dies schlägt sich bei bereits etwas komplexeren Planungsproblemen in einer schlechten Performanz dieser Strategie nieder.

A*

A*: Die Strategie des A*-Ansatzes basiert auf einer Bewertung von Zuständen und darauf basierenden Entscheidungen für die Selektion nächster Schritte. So werden für einen Zustand Ist-Kosten berechnet, die notwendig sind, um den Zustand über die Ausführung von Aktionen ausgehend von der initialen Situation zu erreichen. Zugleich wird über eine Heuristik abgeschätzt, wieviele Kosten noch nötig sein werden, um von diesem Zustand bis zum Ziel zu gelangen. Heuristik und Kostenfunktion sind üblicherweise vom Experten manuell zu bestimmen und variieren je nach Planungsproblem, Domäne und Roboterzelle. Die Summe beider Kosten ergibt anschließend den Fitness-Wert des Zustands. Bei der Selektion des nächsten Schritts wählt die A*-Strategie den Zustand mit dem besten Fitness-Wert und führt hiervon ausgehend die Suche fort. Dadurch fokussiert sich die Suche auf vielversprechende Pfade und ist auf das schnelle Finden eines optimalen Ergebnisses ausgerichtet. Das beim LEGO-Haus gegebene Ecksteinproblem stellt für A* allerdings ebenso eine problematische Herausforderung dar, sofern entsprechende Zustände von der Heuristik nicht explizit mit einem schlechten Fitness-Wert versehen werden. Dazu müsste die Heuristik jedoch sehr speziell die Eigenschaften von LEGO kennen und prüfen, sehr genau über die Möglichkeiten der Roboterzelle Bescheid wissen, sowie ein geeignetes Kriterium für die Berechnung des Fitness-Werts benennen (z. B. Gesamtausführungszeit, Qualität der Prozesse, Gleichauslastung aller Roboter, etc.). Dabei darf die Heuristik keinen Sonderfall wie beispielsweise das Ecksteinproblem unerkannt lassen. Dies alles widerspricht nicht nur dem Aufteilungsgedanken des Planungsansatzes zwischen Domäne und Automatisierung. Vielmehr vermag die Erstellung einer solchen generischen Heuristik einen derart hohen Entwicklungs- und Zeitaufwand zu verursachen, dass es im Vergleich zu einer manuellen Ausprogrammierung der Roboterzelle zur Fertigung des Montageproblems keinen Vorteil mehr ergäbe. Zudem kommt hinzu, dass bei einigen Planungsproblemen wie dem LEGO-Haus die Reihenfolge der zu setzenden Steine meist gleichwertig ist und demnach viele Pfade mit ähnlichen Fitness-Werten existieren. Die Strategie von A* ähnelt in einem solchen Fall sehr stark der der Breitensuche, einhergehend mit deren Nachteilen.

6.2.3 Pessimistischer Suchansatz

Abschnitt 6.2.1 erläuterte mit Abbildung 6.6 den groben Ablauf der Planung auf Domänenebene und ihre Verzahnung mit der Fertigungsplanung, lässt dabei aber die Strategie unspezifiziert, nach der der Suchraum aufgebaut und nach einer gültigen Lösung durchsucht wird. Anforderungen an die Suchstrategie ist zum einen eine möglichst kurze Zeit, die benötigt wird, um eine Lösung zu finden, zum anderen die Qualität des Ergebnisses. Diese kann beispielsweise die Ausführungszeit des geplanten Roboterprogramms sein. Zudem soll eine Anwendbarkeit der Suchstrategie auch auf komplexere Planungsprobleme gegeben sein. Keine der drei genannten Vertreter klassischer Suchalgorithmen (Tiefensuche, Breitensuche, A*) erfüllt alle diese Voraussetzungen. Daher verwendet der Planungsansatz zur effizienten Planung von Roboterprogrammen ein abweichendes Vorgehen mit einem eigenem Algorithmus, dem sogenannten **pessimistischen Suchansatz**, der die erst spät erkennbaren Deadlockprobleme besonders berücksichtigt

Pessimistischer Suchansatz

und dabei gleichzeitig auf eine Fitnesswert-Berechnung – die etwa speziell auf das Planungsproblem zuzuschneiden wäre – verzichtet.

Zustände wie der des Ecksteinproblems besitzen oftmals viele fortführende Planungspfade; dabei ist im gesamten dahinter liegenden Teilbaum ein Deadlock unausweichlich. Eine möglichst frühzeitige Kenntnis über „tote" Teilbäume und ein Abbruch der unnützen Suche darin kann die Planungszeit drastisch verkürzen. Die gewählte Strategie für den Planungsansatz $PaRTs$ versucht daher nicht, vielversprechende Pfade zu einer gültigen Lösung zu finden, sondern fokussiert sich ausschließlich auf das Aufspüren von Deadlocks. Demzufolge wird explizit in Richtungen geplant, die mit größter Wahrscheinlichkeit fehlschlagen werden. Ist eine Deadlock-Situation identifiziert, wird diejenige Vorgängeraktion aufgespürt, die dafür verantwortlich ist. Der gesamte Teilbaum hinter dieser Vorgängeraktion wird als „toter" Teilbaum von der weiteren Suche ausgeschlossen, auch wenn ein Großteil der Pfade darin noch nicht untersucht wurde.

Algorithmus 1 beschreibt die Strategie des Planungsansatzes zur Wahl des als nächsten zu untersuchenden Zustands (Planungssituation) sowie der konkreten weiterführenden Aktion (Domänentask) unter Berücksichtigung des besonderen Umgangs mit Deadlocks. Grundsätzlich folgt die Selektion der nächsten Planungssituation einer Tiefensuche, wobei das Backtracken (Zurückwandern) aus einem erfolglosen Pfad eine wichtige Zusatz-Funktion erhält, die ein frühzeitiges Schließen toter Teiläste erlaubt. Damit dies möglich ist, sind dazu allerdings zwei Grundannahmen über die Planung notwendig:

(1) Kann ein Domänentask in einem Zustand (gegebene Planungssituation) nicht im Rahmen der Fertigungsplanung erfolgreich geplant werden, so ist er auch in einem späteren Schritt nicht mehr lösbar. Der von ihm ausgehende Teilbaum kann als „toter Teilbaum" angenommen werden. Kann beispielsweise bei der Montage ein Einzelteil nicht angebracht werden, da beispielsweise ein anderes, zuvor montiertes Bauteil den Weg zum Einsetzen versperrt, so ist auch eine spätere Montage des Einzelteils nicht mehr möglich. Diese Annahme gilt nicht für allgemeine Planungsprobleme. Für Planungsprobleme im Umfeld der Montage wird sie jedoch als zutreffend angenommen.

(2) Zu einem Zustand (gegebene Planungssituation) ermittelt das Domänentaskmodul mögliche nächste Schritte (Domänentasks), die über Domänenprüfmodule nochmals auf Gültigkeit gefiltert werden. Es wird angenommen, dass Domänentaskmodul und Domänenprüfmodule derart konzipiert sind, dass jeder darüber ermittelte Task aus Domänensicht tatsächlich in diesem Zustand umsetzbar ist. Wird für einen Domänentask keine Automatisierungslösung gefunden, so kann demnach der Rückschluss gezogen werden, dass die verfügbaren Fähigkeiten oder die Konstellation der Roboterzelle schuld daran sind. Wird diese Annahme nicht erfüllt, so kann ein fälschlicherweise angebotener Task dazu führen, dass Annahme (1) den nachfolgenden Suchbaum für tot erklärt. Eine existierende Lösung kann dann eventuell nicht mehr auffindbar sein.

Als Eingabe dient dem Algorithmus der Planung die initiale Produktsituation p_{init} sowie die Ziel-Produktsituation p_{goal}, zwischen denen ein gültiges Programm zu finden ist. Zwei Variablen dienen im Weiteren als globale Speicher: Die Liste $ergebnis$

Algorithmus 1 : Prozessplanung mit gezielter Deadlock-Strategie

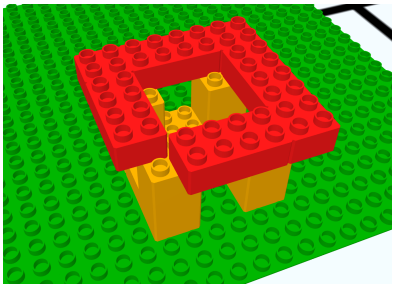
Input : $p_{init} \in \mathcal{S} \restriction Produkt$: Initiale Produktsituation
$p_{goal} \in \mathcal{S} \restriction Produkt$: Ziel-Produktsituation
Output : Abfolge von technischen Tasks $t_1, t_2, .., t_n \in \mathcal{T}$
oder Exception, falls keine Lösung existiert

```
1  global ergebnis, N;
2  Function plane(p_init, p_goal):
3      ergebnis ← [];                          // Ergebnis als Liste von technischen Tasks
4      N ← ∅;                                  // Menge der Domänentasks, die mind. einmal nicht gelöst wurden
5      s ← FP.erweitere(p_init);               // Planungssituation über Schnittst. der Fertigungspl.
6      if (¬planeAbSituation(s, p_goal)) then
7          throw KeinErgebnisGefunden;
8      return ergebnis;
9  Function planeAbSituation(s, p_goal):
10     T ← Alle in s mögl. Domänentasks Richtung p_goal;   // Anhand Domänentaskmodul
11     T ← {t ∈ T | t ist gültig};             // Filterung anhand Domänenprüfmodule
12     V ← ∅;                                  // Menge aller bereits untersuchten Domänentasks
13     while (T \ V ≠ ∅) do
14         t_next ← null;                      // Selektion des nächst zu untersuchenden Domänentasks
15         if (∃t_1 ∈ (T \ V) ∩ N) then
16             t_next ← t_1;                   // Ein bisher bereits fehlgeschlagener Task als nächster Schritt
17         else
18             t_next ← t_2 | t_2 ∈ T \ V;     // Sonstiger Task als nächster Schritt
19         V ← V ∪ {t_next};
           // Bereitstellen von Automatisierungslösungen für den Domänentask über die Schnitt-
20         it ← FP.plane(t_next, s);           // stelle der Fertigungsplanung
21         if (¬it.hasNext()) then             // Existiert keine Automatisierungslösung?
22             N ← N ∪ {t_next};
23             return false;                   // Teilbaum verlassen
24         while (it.hasNext()) do
25             task ← it.next();               // Technischer Task als eine Lösung des Domänentasks
26             ergebnis.append(task);
27             s_res ← nach(task);             // Nachfolge-Situation von task
28             if (s_res ⊩ p_goal) then        // s_res erfüllt Ziel-Produktsituation?
29                 return true;                // Lösung gefunden, hochpropagieren
30             if (planeAbSituation(s_res, p_goal)) then   // Planung rekursiv fortführen
31                 return true;                // Lösung gefunden, hochpropagieren
32             ergebnis.remove(task);          // Nicht erfolgreichen Lösungsschritt entfernen
33     return false;                           // Teilbaum verlassen
```

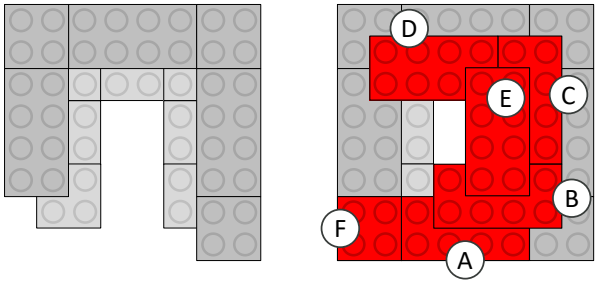
enthält die Automatisierungsprogramme, die bei der Planung für die Domänentasks gefunden wurden. N ist eine Menge an Domänentasks, bei denen die Planung auf Automatisierungsebene mindestens einmal erfolglos blieb.

Zunächst wird die initiale Produktsituation über die Schnittstelle zur Fertigungsplanung zu einer Planungssituation s erweitert, in der zusätzlich der initiale Zustand der Roboterzelle festgehalten ist. Anschließend wird mit der initialen Planungssituation als aktuellem Zustand die rekursive Planung mit der Funktion $planeAbSituation(s, p_{goal})$ begonnen. Für die übergebene, zu untersuchende Planungssituation werden über das Domänentaskmodul sämtliche Domänentasks abgeleitet, die als mögliche Aktionen infrage kommen, und über die verfügbaren Domänenprüfmodule nach Gültigkeit gefiltert. Einer von den Domänentasks wird als nächste Aktion ausgewählt. Dabei spielt eine entscheidende Rolle, ob der Domänentask an einer anderen Stelle der Planung bereits einmal innerhalb der Fertigungsplanung unlösbar, d. h. nicht planbar war (z. B. Setzen des Ecksteins). Denn es werden bevorzugt diejenigen Tasks gewählt, die bisher fehlschlugen und hoffentlich schnell zu einem Deadlock führen (Zeilen 15 bis 19). In Zeile 21 wird über die Schnittstelle zur Fertigungsplanung (FP) ein Iterator abgerufen, der Automatisierungslösungen (technische Tasks) für den gewählten Domänentask zur Verfügung stellt. Kann dieser keine einzige Lösung liefern, so wird der Domänentask in die Menge der fehlgeschlagenen Tasks aufgenommen, und es wird um einen Schritt in der Planung zurückgegangen (Zeile 21 bis 23). Von dort aus wird eben dieser fehlgeschlagene Task bevorzugt ausgewählt, falls er in der aktuellen Planungssituation möglich ist. Liefert die Fertigungsplanung hingegen ein konkretes Ergebnis, so wird dieses in die globale Ergebnisliste aufgenommen (Zeile 27). Erfüllt die nach dem Task geltende Planungssituation die gewünschte Ziel-Produktsituation, so wird die Planung beendet. Ansonsten wird die Planung rekursiv fortgeführt. Nur wenn die rekursive Planung fehlschlägt – d. h. im gesamten Unter-Suchbaum keine Lösung gefunden wird –, ist das Programm des selektierten Tasks wieder aus der Ergebnis-Liste zu entfernen, und es werden stattdessen zunächst andere Automatisierungslösungen für diesen Domänentask, anschließend alternative Domänentasks in einer weiteren Planung untersucht.

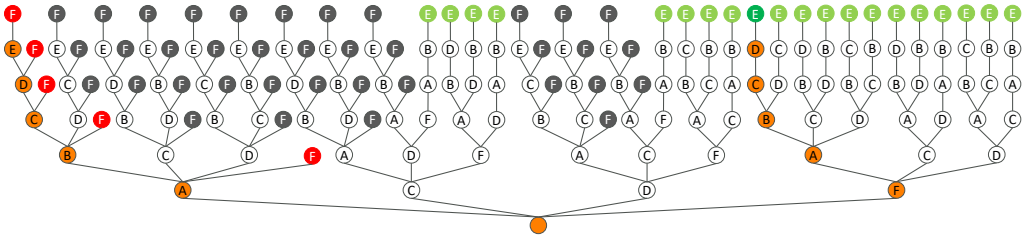
Abbildung 6.9 demonstriert die Anwendung des Algorithmus' auf ein konkretes Planungsproblem. Um die Eignung der Strategie auf spät erkennbare Deadlocks zu untersuchen, wird auf das Haus aus LEGO mit dem Ecksteinproblem zurückgegriffen (siehe Abbildung 6.9a). Die Planung beginnt dabei in der in Abbildung 6.9b gegebenen Situation, wonach Erdgeschoss und ein Teil der ersten Reihe des Hausdachs bereits gesetzt sind. Zu setzen sind nun noch sechs Steine (A-F) des Dachs. Die Roboterzelle bestehe aus einem Roboter mit einem Parallelgreifer, der grundsätzlich jeden der sechs Steine greifen und setzen kann. Dies ist nur dann nicht möglich, wenn es sich um den Eckstein F handelt und der benachbarte Stein A bereits positioniert ist. In Abbildung 6.9c ist der vollständig aufgespannte Suchbaum der Planung gegeben. Jeder Knoten repräsentiert einen Domänentask, der den entsprechenden Legostein setzt. Insgesamt 20 Pfade erreichen eine gültige Lösung (dargestellt durch grüne Blätter des Baums), in 42 Fällen endet der Pfad jedoch in einem Deadlock (schwarze und rote Blätter des Baums), in dem keine Automatisierungslösung für den Domänentask gefunden werden kann. Der Algorithmus durchläuft nun den Baum mit der Strategie einer Tiefensuche und



(a) Eckstein-Problem



(b) Startzustand der Planung und zu setzende Steine A-F



(c) Kompletter Suchraum, mit 42 im Deadlock endenden und 20 erfolgreichen Pfaden

Abbildung 6.9. Erklärung der Suchstrategie des Planungsansatzes anhand eines konkreten Teil-Planungsproblems des LEGO-Hauses.

wählt die erstbeste (im Beispiel immer die am weitest links liegende) Option. Somit erreicht er den Pfad $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$. Für den anschließenden Domänentask, der Stein F setzen würde, kann aufgrund des Ecksteinproblems allerdings keine gültige Automatisierungslösung gefunden werden. Daraufhin wandert der Algorithmus einen Schritt zurück und probiert gezielt den fehlgeschlagenen Task erneut aus, ohne dabei andere Optionen zu berücksichtigen. Doch in diesem vorherigen Schritt – wie auch in den weiter davor liegenden Schritten – ist der Domänentask zwar anwendbar, aber ebenfalls nicht lösbar. Zurück in der ursprünglichen Ausgangssituation ist auch hier der Domänentask F selektierbar. In der vorliegenden Situation existiert nun eine Automatisierungslösung, und die Planung kann die Tiefensuche von dort aus fortführen. Da wieder kein fehlgeschlagener Domänentask bekannt ist, der noch nicht erledigt ist, wird weiter mit der Strategie einer Tiefensuche verfahren. Schließlich wird ein Ergebnis mit der Reihenfolge $F \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$ gefunden.

Insgesamt gibt es im Beispiel sechs zu lösende Domänentasks (A-F). Vom Domänentaskmodul und den Domänenprüfmodulen wird hierfür ein Suchbaum mit insgesamt 167 unterschiedlichen Zuständen aufgespannt, wobei ein Zustand einem der Domäntentasks in einer gegebenen Ausgangs-Situation entspricht. In Abbildung 6.9c sind die Zustände durch beschriftete Knoten dargestellt. Die Strategie des Algorithmus' entscheidet darüber, für wie viele der 167 Knoten eine Fertigungsplanung durchzuführen ist. Der Freiheitsgrad des vorgestellten Algorithmus' bei der Suche besteht in der Methode, den Nachfolge-Domänentask zu wählen, solange kein Problemkandidat bekannt ist. Von den 167 Knoten muss im besten Fall für sechs von ihnen eine Fertigungsplanung

durchgeführt werden. Im schlechtesten Fall – dieser entspricht dem exemplarisch durchgespielten Weg – erfolgt eine Fertigungsplanung für 16 Knoten. Die Lösungen derjenigen Knoten, die auf dem identifizierten Pfad zum Ziel liegen, bilden das Gesamtergebnis der Planung in Form einer Abfolge von technischen Tasks.

Es ist ersichtlich, dass eine reine Tiefensuche in einem ungünstigen Fall eine sehr hohe Anzahl toter Pfade untersuchen würde, bevor sie zu einer Lösung käme. So könnte im Beispiel der gesamte linke Teilbaum, beginnend mit A, exploriert werden. Die gewählte Strategie macht sich hingegen die oben genannten montagespezifischen Annahmen (1) und (2) zunutze, um tote Teilbäume frühzeitig zu schließen. Sind noch keine fehlgeschlagenen Domänentasks bekannt, garantiert die dann ausgeführte Tiefensuche zwar nicht das Finden des optimalen Ergebnisses. Doch zum einen wendet der Planungsansatz bei der untergelagerten Fertigungsplanung Optimierungstechniken an, wonach der zur Verfügung gestellte Iterator die gefundenen Lösungen der Optimalität nach sortiert zurückgibt. Zum anderen ist bei der Montage die Reihenfolge der Montageschritte oftmals nicht ausschlaggebend für ein Optimalitätskriterium. Bei der Montage von LEGO mit nur einem Roboter ist die Reihenfolge der Steine nahezu irrelevant.

Es bleibt unbenommen, die Planung auf Prozessebene auch nach der ersten Lösung in Form eines Anytime-Ansatzes weiter fortzusetzen, um eventuell ein noch besseres Ergebnis zu finden. Ist das Planungsergebnis gut genug, wird die Planung abgebrochen.

Zusammenfassung. Mit der Prozessplanung existieren Abläufe abstrakter Aktionen, die notwendige Prozesse zur Fertigung eines Produkts beschreiben. Dieses Kapitel erläutert Problemstellungen einer Ausführung solcher Prozesse mit realen Robotern und erklärt Konzepte der dafür notwendigen Planung auf der Automatisierungsebene. Da die Anzahl an Möglichkeiten, wie einzelne Prozesse konkret realisiert werden können, und der dadurch bestimmte Planungsaufwand üblicherweise groß ist, werden Mechanismen für eine effiziente Ergebnisberechnung vorgestellt. Zudem wird ein nachträgliches Optimierungsverfahren erläutert, das einen Ausführungsplan insbesondere bei Verwendung mehrerer Roboter hinsichtlich seiner Ausführungsdauer verbessert.

7 Fertigungsplanung

Auf Ebene der Prozessplanung werden mögliche Abläufe an abstrakten Domänentasks gesucht, die zumindest aus Domänensicht gültige Montagereihenfolgen für das Produkt darstellen. Für die Domänentasks sind im Rahmen der anschließenden Fertigungsplanung gültige und möglichst optimale Automatisierungslösungen zu finden, die den vom jeweiligen Domänentask abstrakt beschriebenen Prozess mit Robotern geeignet umsetzt. Abbildung 7.1 ordnet die Phase der Fertigungsplanung in den Gesamtansatz von $PaRTs$ ein. Schritt 5 des Planungsansatzes – die Zuweisung – bezieht die zur Fertigung verwendete Roboterzelle in den Planungsfortschritt mit ein und erlangt über Fähigkeitenmodule der Experten mögliche Automatisierungsabläufe für Domänentasks. Über Validatormodule werden die Lösungen in Schritt 6 (Überprüfung) auf verschiedene Kriterien untersucht und entsprechend ihrer Gültigkeit gefiltert. Beschrieben sind beide Schritte in Abschnitt 7.1. Auf die Konzepte der darauf aufbauenden Planung geht Abschnitt 7.2 genauer ein. Beim Einsatz mehrerer Roboter kann sich die Ausführungsdauer des gefundenen Montageprogramms zusätzlich verbessern, wenn Teilprogramme je nach Roboter und Aufgabe parallel zueinander ausgeführt werden können. Abschnitt 7.3 beschreibt die möglichen Optimierungstechniken hierfür.

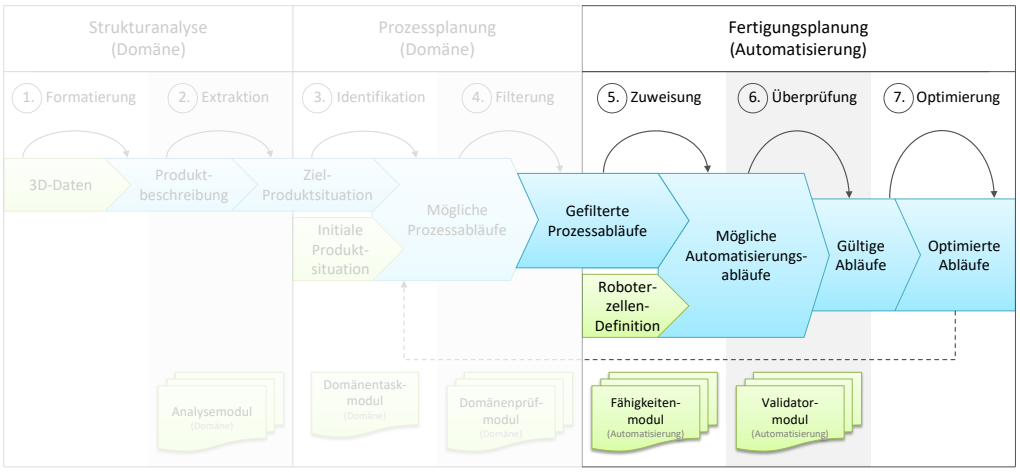


Abbildung 7.1. Einordnung der Fertigungsplanung in den Gesamtansatz $PaRTs$.

7.1 Grundlagen der Fertigungsplanung

Soll für einen abstrakten Domänentask eine konkret ausführbare Automatisierungslösung bestimmt werden, so ist eine präzise Beschreibung der Roboterzelle und der darin befindlichen Objekte für die möglichen Ausprägungen des Ergebnisses ausschlaggebend. Aber auch die zur Verfügung stehenden Fähigkeiten der Roboter und deren Einbeziehung in die Lösung haben einen großen Einfluss auf das Ergebnis der Planung. Damit ein konsistenter Zustand der Planung beschrieben werden kann, wird die Produktsituation um Informationen der Roboterzelle erweitert und in eine Planungssituation überführt. All diese Aspekte werden in den folgenden Unterkapiteln genauer beleuchtet.

7.1.1 Modellierung der Roboterzelle

Das der Montageplanung zugrunde liegende Modell der Roboterzelle ist von einem entsprechenden Experten anzufertigen. Dieses gibt Auskunft über sämtliche Aktuatorik, die in der Roboterzelle verfügbar ist – beispielsweise Roboter, die für die Montage zur Verfügung stehen. Besonders wichtig bei der Offline-Programmierung ist die Kenntnis über die exakte Position der Roboter in der Zelle. Nur damit lassen sich mittels Kinematikberechnungen präzise Endeffektorpositionen bestimmen. Doch ist hier ein Spagat zwischen einer perfekten virtuellen Welt und einer realen, mit Ungenauigkeiten behafteten Welt zu überwinden. Fertigungs- und Montagetoleranzen, sowie Absolut- und Wiederholgenauigkeit der Roboter spielen hier eine Rolle. Daher ist eine Übernahme idealer Roboterpositionen – beispielsweise aus einer Entwurfszeichnung der Roboterzelle – in das Roboterzellenmodell für den Planungsansatz nicht unbedingt ausreichend.

Stattdessen wird das Roboterzellenmodell für eine real existierende Roboterzelle mit speziell dafür angefertigten Vermessungsinformationen ausgestattet. Für die Güte eines Montageschritts ist es nicht ausschlaggebend, wie präzise die Position der Roboterbasis bekannt ist; vielmehr wird die Qualität des Prozesses durch die Position seines

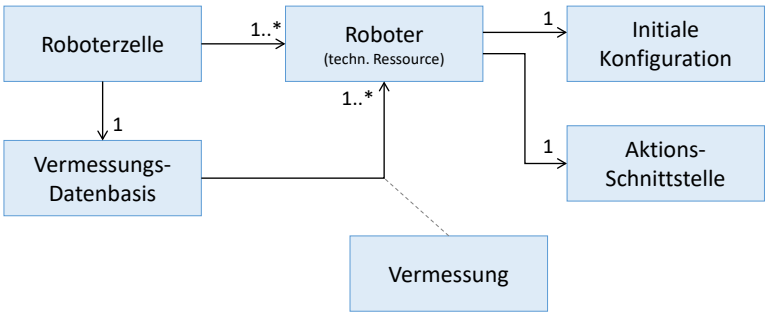


Abbildung 7.2. Definition der Roboterzelle für den Planungsansatz.

Werkzeugs relativ zu den am Prozess beteiligten Objekten beeinflusst. Daher wird die Position sämtlicher Objekte der Roboterzelle, wie Tische und Formen, üblicherweise mit speziellen, an den Flansch der Roboter montierten Messspitzen angefahren. Über die eingenommenen Achsstellungen des Roboters kann daraus eine relative Position berechnet werden. Dieser Vorgang wird *Einmessen* oder *Vermessen* der Roboterzelle genannt. Über die damit erlangten Daten lässt sich die Position der Roboterbasen in einem gemeinsamen Weltkoordinatensystem bestimmen, wodurch eine automatische Programmierung der Roboterzelle in einem Offline-Ansatz möglich wird. Für den Planungsanssatz werden die Messdaten in einer Vermessungs-Datenbasis niedergeschrieben, die als Teil der Roboterzellen-Definition der Planung zur Verfügung stehen.

Manchmal kommt es zu Änderungen an der Roboterzelle, beispielsweise muss ein Ablagetisch ummontiert werden, oder an einem Roboter erfolgt eine Neujustage. In einem solchen Fall stimmt das Modell der Roboterzelle nicht mehr mit der Realität überein. Um über die Offline-Planung weiterhin konsistente und real ausführbare Roboterprogramme zu erhalten, sind von Automatisierungsexperten lediglich die Vermessungen in der Roboterzelle neu anzufertigen und anzugleichen.

Ein weiterer Bestandteil der Roboterzellen-Definition sind detaillierte Angaben über die einzelnen Roboter. Zu Beginn einer Programmausführung – und auch zu Beginn der Planung – soll sich die Roboterzelle in einem definierten Zustand befinden. Anders als bei reinen Softwareapplikationen bezieht sich ein Roboterprogramm auf Elemente der realen Welt, die nicht vor jeder Programmausführung immer den gleichen Zustand einnehmen. Beginnt ein Roboter aus einer undefinierten Stellung sein Programm, so ist weder eine Aussage über Kollisionsfreiheit noch über die Gültigkeit des weiteren Programmverlaufs machbar. Damit der Planungsansatz für jeden Roboter mit einer definierten Ausgangsstellung zu planen beginnt und dem Roboterprogramm eine initiale Routine zum Herstellen des definierten Zustands der Roboterzelle hinzufügen kann, ist jedem Roboter, wie in Abbildung 7.2 dargestellt, eine initiale Konfiguration zugeordnet. Neben konkreten Achsstellungen enthält diese beispielsweise auch die Öffnungsweite des Greifers.

Der Planungsfortschritt wird durch Aktionen erzielt, die ein Roboter ausführt. Ob ein Roboter für eine bestimmte Aktion allerdings geeignet ist, oder ob er eine bestimmte Zielposition im Raum überhaupt erreichen kann, muss ggf. überprüft werden. Dafür

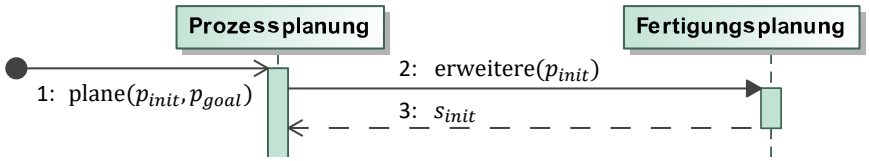


Abbildung 7.3. Schnittstelle der Fertigungsplanung zur Erweiterung der Produktsituation um Attribute der Roboterzelle hin zur initialen Planungssituation.

verweist der Roboter auf eine eigene Aktions-Schnittstelle, die über Berechnungen der direkten wie auch inversen Kinematik für diesen Roboter verfügt und zudem einfache Bewegungsprofile wie Punkt-zu-Punkt-Bewegungen (**PTP**) oder Linearbewegungen (**LIN**) erstellen kann. Solche geplanten Bewegungen werden als atomare technische Tasks zurückgegeben. Auch die Abfrage von Kollisionshüllen des Roboters zu gegebenen Achsstellungen ist über die Aktions-Schnittstelle möglich. Diese werden vom Planungsansatz zur Kollisionsvermeidung bei der Planung herangezogen.

PTP

LIN

7.1.2 Ermittlung der initialen Planungssituation

Die Planung selbst arbeitet auf Basis der Produkt- und Planungssituation – also anhand von Attributen. Die vorliegende Beschreibung der Roboterzelle muss daher in das entsprechende Format der Planung überführt werden. Die Prozessplanung bietet dazu die bereits erwähnte Schnittstelle (siehe Abbildung 7.3) an, über die zu Beginn der Planung die gegebene Produktsituation in eine korrespondierende Planungssituation inklusive aller Attribute der Roboter zum Ausgangszeitpunkt der Montage überführt wird. Die Attribute der Roboter werden dabei anhand von Informationen erstellt, die aus der Definition der Roboterzelle gewonnen werden.

Laut Abschnitt 4.1.2 gibt es drei Arten von Attributen der technischen Ressourcen, die es nun zur Bestimmung einer Planungssituation anhand der Informationen aus der Roboterzellen-Definition abzuleiten gilt: Die **Position** definiert die räumliche Anordnung der Roboterbasis in der Roboterzelle. Anhand der angefertigten Vermessungsdaten eines Roboters kann eine Drehverschiebung zwischen dessen Basiskoordinatensystem und einem gewählten Weltkoordinatensystem berechnet werden, aus der das Positions-Attribut erstellt wird. Die **Konfiguration** beschreibt eine eindeutige Einstellung der technischen Ressource. Bei einem Roboter sind dies die Achsstellungen zu einem bestimmten Zeitpunkt. Mithilfe der direkten Kinematik kann so eine Drehverschiebung zwischen Roboterbasis und dem Flansch des Roboters abgeleitet werden, die damit zu einem bestimmten Zeitpunkt transitiv die Position des Flansches bezüglich des Weltkoordinatensystems bestimmt. Für die Erstellung der initialen Planungssituation wird die initiale Konfiguration, die in der Roboterzellen-Definition für den jeweiligen Roboter gegeben ist, herangezogen, um das Konfigurations-Attribut für diesen zu bilden. Das Greifen eines Einzelteils durch einen Greifer wird über das Attribut der **Belegung** ausgedrückt. Dieses enthält zudem die Information, wie und an welcher Stelle das Objekt gegriffen wurde. Mit dem Wissen über geometrische Eigenschaften des Greifers kann somit eine Drehverschiebung zwischen der Greiferbasis (also dem Flansch des Robo-

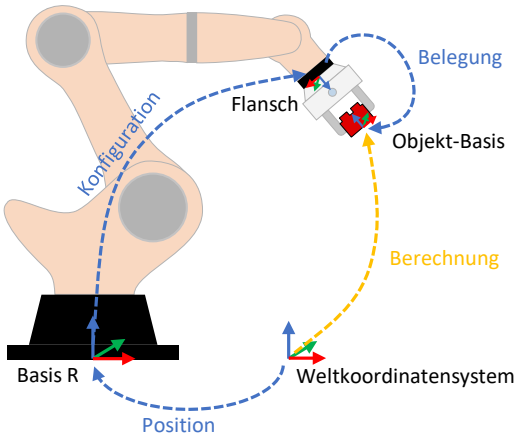


Abbildung 7.4. Eine Planungssituation enthält neben den Attributen der Produktsituation auch Attribute über technische Ressourcen. Die drei Arten dieser Attribute sind am Beispiel eines Roboters aufgeführt.

ters) und dem Einzelteil berechnet werden. Im Gesamten ist damit zu einer gegebenen Planungssituation die Position des Einzelteils bezüglich des Weltkoordinatensystems bestimmbar.

Während sich Attribute der Produktsituation ausschließlich auf Einzelteile beziehen, so werden von Attributen der Roboterzelle weitere Objekte referenziert. Bei der Position ist es die Basis des Roboters, die als Objekt in der Welt verankert wird. Die Konfiguration referenziert als Objekte ebenso die Basis wie auch den Flansch des Roboters. Über die Belegung wird der Flansch und das gegriffene Einzelteil in Beziehung gebracht. Ebenso wie Einzelteile spielen damit auch die Basis sowie der Flansch des Roboters eine besondere Rolle bei der Planung. Damit nicht nur Einzelteile sondern auch Basis und Flansch der Roboter von Attributen referenziert werden können, werden auch sie vom Planer wie existierende Objekte behandelt, obwohl der jeweils dahinterstehende Frame eigentlich fiktiver Natur ist.

Abbildung 7.4 beschreibt die Darstellung einer Planungssituation über ihre Attribute beispielhaft an einem Roboter. Dieser ist mit einem Parallelgreifer ausgestattet, mit dem er gerade einen roten Legostein hält. Die Abbildung hebt die drei Drehverschiebungen, die unmittelbar durch die Attribute des Roboters beschrieben sind, hervor. Ebenso sind der Legostein sowie Basis und Flansch des Roboters als referenzierbare Objekte der Attribute herausgestellt. Ausgehend vom Weltkoordinatensystem sind zum Zeitpunkt einer Planungssituation damit die Position der Roboterbasis, die Position des Flansches sowie die Position des Legosteins bekannt.

7.1.3 Zuweisung technischer Ressourcen: Fähigkeitenmodul

Bei der Fertigungsplanung wird ein gegebener Domänentask in ein konkretes Roboterprogramm überführt. Dabei wird die zu Beginn des Domänentasks gültige Planungssituation berücksichtigt, um eine Folge von technischen Tasks zu finden, die letztendlich das Ziel des Domänentasks erfüllen. Auf eine gegebene Planungssituation werden da-

zu Fähigkeitenmodule angewandt, die dafür eine Menge an möglichen technischen Tasks berechnen. Diese sind so gewählt, dass sie den Bauteilzustand dem Ziel des Domänentasks annähern. Jedes Fähigkeitenmodul repräsentiert eine konkrete Fähigkeit eines bestimmten Roboters, die es in einer ausführbaren Aktion bereitstellen kann. Ihre Schnittstelle (siehe Abbildung 4.6 in Abschnitt 4.1.4) erwartet eine Planungssituation, von der ausgehend geplant wird, sowie die zu erreichende Produktsituation des Domänentasks, die als lokale Ziel-Produktsituation bezeichnet wird. Vergleicht ein Fähigkeitenmodul die Attribute der Planungssituation mit denen der Ziel-Produktsituation, so kann es alle noch zu erledigenden Montageschritte feststellen. Fähigkeitenmodule werden von Automatisierungs- und Prozessexperten entwickelt und dem Planungsansatz beigefügt.

Man stelle sich beispielsweise eine „Pick-and-Place"-Fähigkeit vor, die einen Legostein greifen, transportieren und diesen an einem anderen Ort wieder absetzen kann. Hieran fallen zwei Besonderheiten auf, aus denen weitere Anforderungen an das Konzept der Fähigkeitenmodule abgeleitet sind:

(1) Eine Pick-and-Place-Anwendung resultiert nicht in nur einer einzelnen Bewegung, sondern wird in der Regel durch eine Abfolge von mehreren Aktionen verwirklicht. So bedarf es zunächst eines Anfahrens des Zielobjekts und des Schließens des Greifers, bevor das Objekt ggf. geeignet abgelöst und zum Zielort transferiert werden kann. Auch dort ist eventuell eine spezielle Absetz-Strategie erforderlich, um das Objekt prozesssicher zu positionieren. Ein Fähigkeitenmodul erzeugt im Allgemeinen keinen atomaren technischen Task sondern ein möglicherweise komplexes Programm. Das Modell des technischen Tasks muss dies ausdrücken können.

(2) In einer gegebenen Situation existieren oft mehrere Objekte, für die das Fähigkeitenmodul eine Pick-and-Place-Aktion anbieten kann. Für jede dieser Aktionen stehen dem Roboter zudem zahlreiche Möglichkeiten offen, wie und an welcher Stelle er dieses Objekt greift. In Summe resultiert dies in einer großen Menge an Alternativen, die den Suchraum der Planung exponentiell erweitert. Darüber hinaus steht meist nicht nur eines sondern viele weitere solcher Fähigkeitenmodule zur Verfügung, die alle möglichen Aktionen der Roboterzelle abbilden. Es bedarf eines Konzepts, mit den vielen alternativen Möglichkeiten eines Planungsschritts so umzugehen, dass nicht jede untersucht werden muss, aber die optimale gefunden wird.

Einzeltask

HalO-Task

Komplexer Task

Um über ein Fähigkeitenmodul komplexe Programme ausdrücken zu können (Anforderung 1), gibt es für den technischen Task mehrere mögliche Realisierungen. Abbildung 7.5 untergliedert den technischen Task auf der einen Seite in den **Einzeltask**, der eine atomare Aktion darstellt. Dies kann beispielsweise eine PTP- oder LIN-Bewegung eines Roboters sein, für die eine kartesische Zielposition gegeben ist, oder aber es ist eine Implementierung einer anderweitigen Aktion darin definiert. Auf der anderen Seite kann der technische Task durch einen **HalO-Task** – im weiteren auch oft **komplexer Task** genannt – realisiert werden, der selbst wiederum mehrere technische Tasks als Kindelemente enthält. Die Semantik, die eine Ausführungsbeziehung der Kindelemente

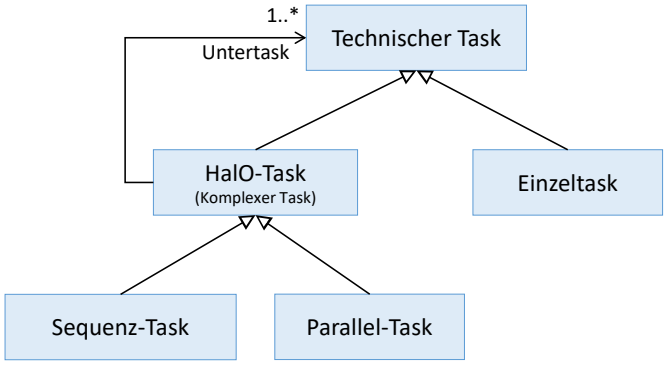


Abbildung 7.5. Arten des technischen Tasks, über den verschachtelte komplexe Programme abgebildet werden können.

zueinander beschreibt, legt der HalO-Task dabei über eine partielle Ordnung (Halb-Ordnung) über seine Untertaks fest. Ein HalO-Task erzwingt dabei eine Zyklenfreiheit der durch ihn beschriebenen partiellen Ordnung. Als Spezialfälle des HalO-Tasks werden im Rahmen des Planungsansatzes zwei Task-Arten explizit zur Verfügung gestellt: Ein **Sequenz-Task** beschreibt eine totale Ordnung über seine Untertasks und damit die strikte Hintereinanderausführung. Der **Parallel-Task** enthält dagegen eine leere Relationsmenge der partiellen Ordnung und drückt damit eine asynchrone Ausführungssemantik sämtlicher Untertasks aus.

Sequenz-Task

Parallel-Task

Ein Fähigkeitenmodul bezieht sich bei der Erstellung seiner Ergebnisse auf die Aktions-Schnittstelle des oder der von ihm beschriebenen technischen Ressource(n). Die Funktionen einer Aktions-Schnittstelle liefern üblicherweise Einzeltasks zurück, die vom Fähigkeitenmodul über die Verwendung komplexer Tasks konkateniert und verschachtelt werden können. Durch das hierarchische Taskmodell ist es dem Fähigkeitenmodul möglich, einfache wie auch komplexe Programme darüber auszudrücken.

Um den Suchraum der Planung gemäß Anforderung 2 beherrschbar zu gestalten, soll die Komplexität der von den Fähigkeitenmodulen gefundenen Lösungsmenge reduziert werden, ohne dabei jedoch potentiell wichtige Alternativen zu verlieren. Auf Ebene der Fertigungsplanung wird dies unter anderem durch eine Klassifizierung erzielt, die ähnliche Ergebnisse eines Fähigkeitenmoduls zusammenfasst. Ähnlich sind zwei Ergebnisse dann, wenn deren resultierende Produktsituationen exakt übereinstimmen – also sowohl der Bauteilzustand wie auch der Zustand der Roboterzelle identisch sind. Man kann sich dies beispielsweise so vorstellen, dass ein Roboter einen Legostein mit einer Vielzahl an möglichen Greifpositionen an einen neuen Ort setzen kann und nach der Aktion in jedem Fall wieder am selben Platz steht. Bei derartigen Alternativen ist der weitere Verlauf der Planung völlig unabhängig von der konkret gewählten Lösungsoption aus dieser Klasse. Wird eine optimale und gültige Lösung identifiziert, können daher alle anderen Varianten der Klasse vernachlässigt werden. Wird im Anschluss an eine Lösung ein Deadlock bei der Planung festgestellt, so führen auch alle anderen Vertreter der Klasse in diesen Deadlock. Lediglich Kollisionen, die während der Ausführung einer gewählten Lösung auftreten können, rechtfertigen das Suchen nach Alternativen innerhalb einer

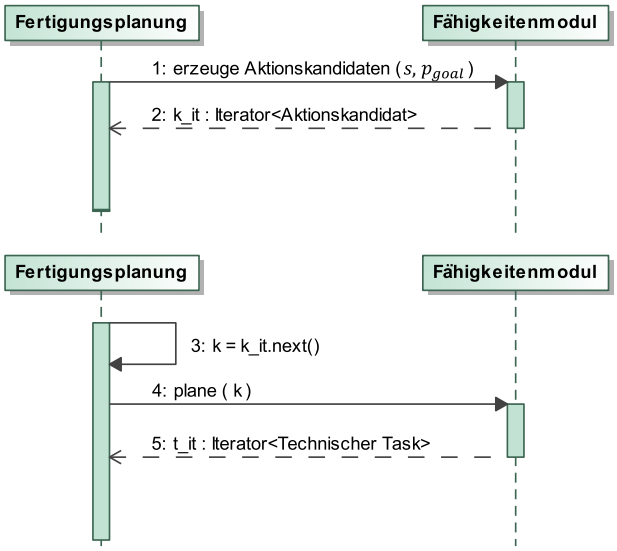


Abbildung 7.6. Interaktion zwischen Fertigungsplanung und Fähigkeitenmodulen über die zwei zur Verfügung stehenden Funktionen der Schnittstelle.

Klasse. Für die Planung folgt daraus, dass der Verzweigungsgrad des Suchbaums nun durch die Anzahl der Lösungsklassen bestimmt ist und nicht durch die größere Anzahl der unterschiedlichen Lösungen. Eine Klasse an Lösungen wird bei der Planung auch **Aktionskandidat** genannt. Eine formale Definition ist in Definition 7.1 gegeben.

Aktionskandidat

Definition 7.1. *Aktionskandidaten*
Für eine initiale Situation $s \in \mathcal{S}$ und eine Nachfolge-Situation $s' \in \mathcal{S}$ sei die Menge $k_{s,s'}$ gegeben durch all diejenigen Tasks $t \in \mathcal{T}$, deren Ausführung in Situation s dieselbe Attributänderung $Effekt(t, s)$ bewirkt und in die gemeinsame Situation s' überführt, i. Z.

$$k_{s,s'} = \{t \in \mathcal{T} \mid nach(s, t) = s'\} .$$

*Eine solche klassifizierte Menge wird **Aktionskandidat** für eine Situation s genannt.*

Sei $\mathcal{K}$ die Menge aller Aktionskandidaten, und $\mathcal{K}_s \subseteq \mathcal{K}$ die Menge all derjenigen Aktionskandidaten, die von Situation s ausgehen.

Für die Planung wird die Klassifizierung der Lösungen in Aktionskandidaten konzeptuell über eine zweigeteilte Schnittstelle des Fähigkeitenmoduls realisiert. Abbildung 7.6 stellt die zwei Interaktionspunkte dar, auf die während der Fertigungsplanung zugegriffen wird. Mit dem Aufruf in Schritt 1 wird dem Fähigkeitenmodul die Ausgangssituation der Planung s sowie die zu erreichende Produktsituation p_{goal}, die vom Domänentask als lokales Ziel benannt wird, übergeben und daraus eine Menge an Aktionskandidaten berechnet. Jeder Aktionskandidat beschreibt eine Lösungsklasse, also eine noch abstrakte mögliche Aktion, die mit der Fähigkeit des Roboters in der gegebenen Situation in irgendeiner Form ausgeführt werden kann. In einem späteren Schritt der Fertigungsplanung kann unter Bezug auf einen solchen Aktionskandidaten im Bedarfsfall eine konkrete Lösung berechnet werden. Diese Ausprägung des Aktionskandidaten enthält

dann detaillierte Angaben darüber, wie die Aktion tatsächlich ausgeführt wird. Die Rückgabe erfolgt in Form eines Iterators, damit auch hier gültige Lösungen der Optimalität nach sortiert erlangt werden können.

Greifen wir das Beispiel mit LEGO wieder auf: Das Fähigkeitenmodul „Pick-and-Place“ eines Roboters identifiziert über die Attributdifferenz aus initialer Planungssituation und Ziel-Produktsituation die Menge aller Steine, die noch nicht richtig positioniert sind. Für jeden dieser Steine wird – sofern dieser im Arbeitsbereich des Roboters liegt – im ersten Schritt ein Aktionskandidat erstellt, dem die an alter Stelle abzubauenden sowie die an neuer Stelle aufzubauenden „LegoSteckVerbindung“-Attribute des Steins zugeordnet sind. Erst bei einem konkreten Lösungsbedarf wird in einem zweiten Schritt ein optimales Programm zum Versetzen des Steins in Form eines technischen Tasks geplant, ohne dabei auf höherer Planungsebene alle Möglichkeiten hierzu betrachten zu müssen.

7.2 Planung auf Automatisierungsebene

Während ein Fähigkeitenmodul einen einzelnen Schritt ausgehend von einer zuvor geltenden Situation ermöglicht, soll die Fertigungsplanung mehrere dieser Schritte zu einem kompletten, geeigneten Programmablauf zusammensetzen, um damit ein gültiges Roboterprogramm für einen Domänentask zu erzielen. Die Wahl der Fähigkeitenmodule wie auch der Aktionskandidaten und konkreter Lösungen stellt dabei ein Optimierungsproblem dar, für das in diesem Teilkapitel eine dementsprechend angepasste Suchstrategie vorgestellt wird.

7.2.1 Sequentielle zustandsbasierte Planung

Auch auf Ebene der Fertigungsplanung wird die Suche nach gültigen Programmen zustandsbasiert ausgeführt. Dabei beschreiben Planungssituationen die Zustände und technische Tasks die Aktionen. Abbildung 7.7 zeigt exemplarisch einen Domänentask, der ausgehend von einer aktuellen Planungssituation (vorher) eine herzustellende Produktsituation (nachher) über das Bauteil beschreibt. Ziel der Fertigungsplanung ist es, als Verfeinerung des Domänentasks einen gültigen Ablauf an technischen Tasks zu finden, deren resultierende Planungssituation – die auch den Zustand der Roboterzelle beschreibt – die Ziel-Produktsituation des Domänentasks erfüllt.

Die Fertigungsplanung fokussiert sich bei der Suche gezielt auf das Finden sequentieller Abfolgen von technischen Tasks und vernachlässigt absichtlich Varianten mit teil-parallelen Abläufen. Im Vergleich zu den Lösungs-Graphen mit parallelen Verzweigungen, die in diesem Fall zu durchsuchen wären, ist die Anzahl der möglichen Pfade bei einer Planung sequentieller Programme hingegen deutlich geringer, was auch den Suchraum immens reduziert. Ebenso gestaltet sich die Bestimmung einer resultierenden Planungssituation nach einem technischen Task bei sequentiellen Pfaden deutlich einfacher als bei Graphen. Denn so kann der globale Zustand der Roboterzelle in jeder Planungssituation eindeutig beschrieben werden, wohingegen eine Beschreibung der Situation in einem Teilast des Graphen, die keine Aussage über Roboter in parallelen Verzweigungen macht und garantiert widerspruchsfrei ist, eines besonderen Konzepts bedürfte.

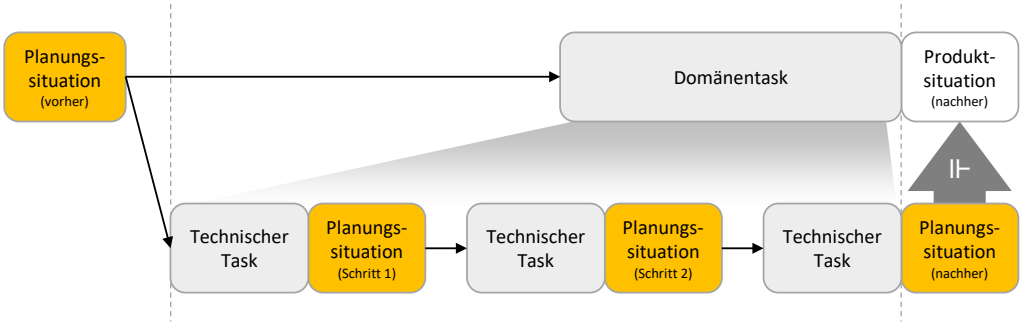


Abbildung 7.7. Fertigungsplanung auf Automatisierungsebene durch Verfeinerung eines Domänentasks. Die resultierende Planungssituation des geplanten Ablaufs muss die Produktsituation des Domänentasks erfüllen.

Ohne Weiteres kann der sequentielle Pfad an technischen Tasks nachträglich parallelisiert und dadurch effizienter gestaltet werden. Die Parallelisierung wird im Nachgang an die Planung im Rahmen einer nachträglichen Optimierung ausgeführt. Dadurch wird der Suchraum nicht mit den möglichen Kombinationen der parallelen Ausführung exponentiell erweitert, stattdessen ist die Optimierung lediglich auf gültige Pfade für den Domänentask anzuwenden. Das Konzept und die Funktionsweise dieser Optimierung ist in Abschnitt 7.3 erläutert.

Da bereits jeder Einzelschritt der Fertigungsplanung den Zustand der Roboterzelle und des Bauteils in einer resultierenden Produktsituation beschreibt (siehe Abbildung 7.7 Schritt 1 und 2), können ungültige Zustände und dadurch ungültige Programme frühzeitig erkannt und von der weiteren Planung ausgeschlossen werden. Abschnitt 4.1.4 führte dazu Validatormodule ein, die eine Planungssituation auf Gültigkeit überprüfen. Diese Module sind von Experten zu konzipieren und können in vielfacher Anzahl dem Planungsansatz hinzugefügt werden, wobei jedes Validatormodul einen bestimmten Aspekt der Montage prüfen kann. Dafür erhält es als Eingabeparameter die zu prüfende Planungssituation. Im Gegensatz zu Domänenprüfmodulen steht es Validatormodulen offen, neben dem Bauteil auch die technischen Ressourcen der Roboterzelle in die Überlegungen mit einzubeziehen. Dadurch kann beispielsweise geprüft werden, ob eine instabile Bauteilkonstruktion (z. B. die Treppe aus LEGO) ausreichend von einem Roboter gestützt wird. Zumindest im Zustand vor sowie nach jedem technischen Task kann über ein Validatormodul auch die Kollisionsfreiheit zwischen Robotern und Bauteilen sichergestellt werden. Oft sind Berechnungen, wie z. B. die Kollisionsprüfung, sehr teuer hinsichtlich der benötigten Zeit, zumal wenn diese auf jede neue Planungssituation erneut angewandt werden. Grundsätzlich kann eine teure Berechnung aber dann vorteilhaft sein, wenn sie dazu beiträgt, den Suchraum signifikant einzuschränken. Die Anwendung der Berechnung auf jeden einzelnen Zustand der Planung hat linearen Einfluss auf die Gesamt-Planungsdauer. Kann dank dieser der kombinatorische Suchraum eingeschränkt werden, reduziert sich dadurch die Gesamt-Planungsdauer um einen exponentiellen Faktor.

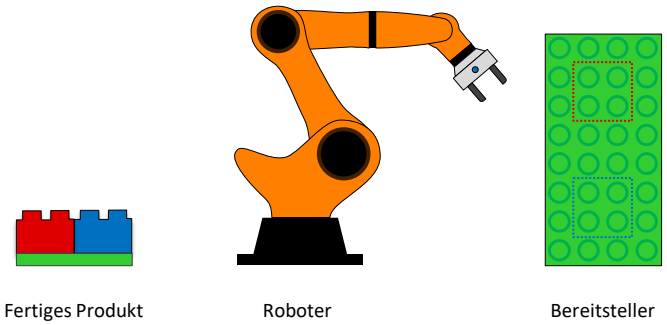


Abbildung 7.8. Beispiel einer Montage: Ein Produkt aus LEGO wird in einer Roboterzelle bestehend aus Roboter und Bereitsteller gefertigt.

7.2.2 Korridor-geführter Suchansatz

Auch wenn die Planung lediglich sequentielle Roboterprogramme untersucht und dadurch den Suchraum stark eindämmt, so gibt es dennoch zahlreiche Freiheitsgrade, die immer noch einen hohen Verzweigungsgrad des Suchraums verursachen. An einem einfachen und besonders kleinen Beispiel soll dies veranschaulicht werden. Abbildung 7.8 beschreibt zwei Legosteine auf einer grünen Platte als fertiges Produkt, das mithilfe eines Roboters zusammengesetzt werden soll. Der Roboter besitzt dazu die Fähigkeit „Pick-and-Place“, um einen Stein an gegebener Stelle aufzunehmen und an einer anderen wieder abzusetzen. Als zweite technische Ressource ergänzt ein Bereitsteller in Form einer Legoplatte die Roboterzelle mit der Fähigkeit „Anbieten“, also dem Bereitstellen von Legosteinen für die Montage. In einem realen Szenario würde diese Aufgabe etwa ein Förder- oder Zubringersystem übernehmen. Der Bereitsteller besitzt jeweils einen separaten Slot für den roten und den blauen Stein, sodass diese unabhängig voneinander angeboten werden können.

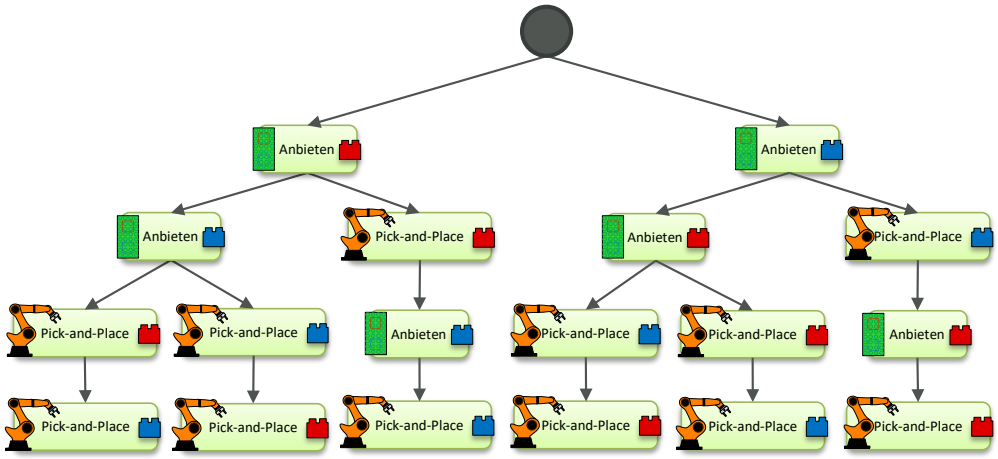


Abbildung 7.9. Vollständiger Suchraum, der über die zur Verfügung stehenden Fähigkeiten der Roboterzelle zur Fertigung des Produkts aufgespannt wird.

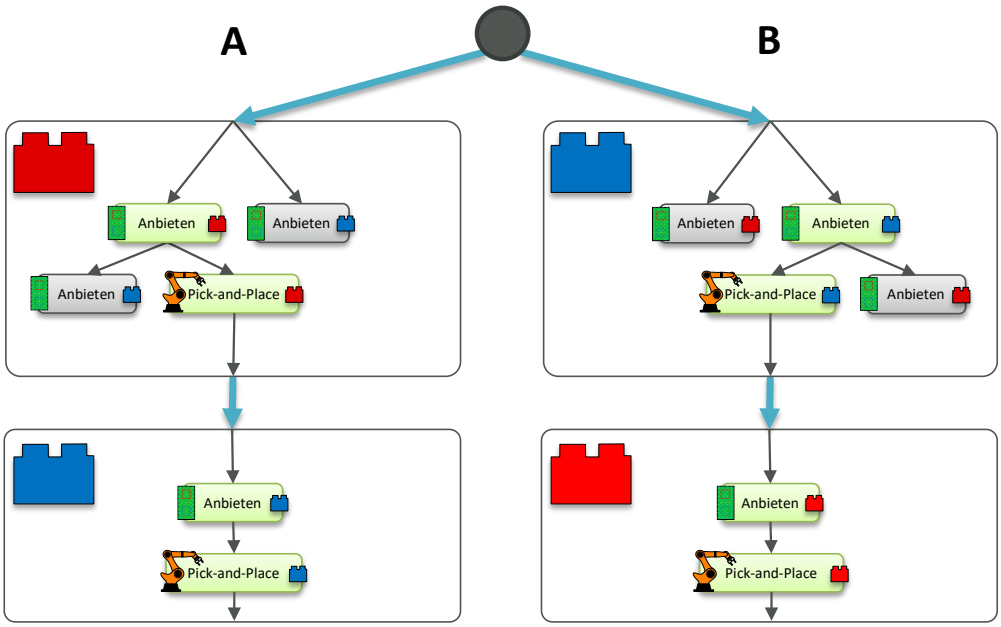


Abbildung 7.10. Planung innerhalb von Domnänentasks auf dem „Korridor relevanter Pfade“.

Die initiale Planungssituation sei dadurch gegeben, dass Roboter und Bereitsteller an einer geeigneten Position im Raum stehen; der Roboter nehme seine Home-Achsstellung ein und der Bereitsteller sei unbelegt. Zudem befindet sich bereits die Grundplatte im Raum, auf der das Produkt montiert werden soll. Abbildung 7.9 beschreibt den Suchraum, der ausgehend von dieser initialen Roboterzelle mit den gegebenen Fähigkeiten möglich ist. In einem ersten Schritt kann die Fähigkeit des Roboters keine anwendbare Aktion zur Verfügung stellen. Der Bereitsteller dagegen kann alternativ einen der beiden Steine anbieten. Im zweiten Schritt kann je nach bereitgestelltem Stein dieser vom Roboter gegriffen und an die Zielposition gesetzt werden. Alternativ stellt der Bereitsteller den zweiten Stein zur Verfügung. Dies wiederholt sich, bis nach dem vierten Schritt schließlich beide Steine auf der grünen Platte abgesetzt sind.

In Summe existieren sechs unterschiedliche gültige Pfade, also sechs Möglichkeiten, wie die Roboterzelle das gegebene Produkt herstellen kann. Würde man einen zweiten Roboter mit der gleichen Fähigkeit dazunehmen, so würden zum Setzen der beiden Steine bereits 24 gültige Pfade existieren. Und dabei wird in diesem Beispiel angenommen, dass jede Fähigkeit lediglich eine konkrete Ausprägung der Aktion, z. B. „Pick-and-Place“ mit nur einer Greifposition, liefert. Jede neue Fähigkeit, die der Roboterzelle hinzugefügt wird, vergrößert den Suchraum exponentiell. Da der Planungsaufwand bei komplexeren Montageproblemen und größeren Roboterzellen förmlich explodiert, können sehr viele Montageprobleme mit einer klassischen Suchstrategie nicht gelöst werden.

Um die Komplexität des Planungsproblems, die überwiegend durch eine Kombinatorik verschiedener Aspekte hervorgerufen wird, zu reduzieren, unterteilt der vorgestellte Planungsansatz die Suche nach Lösungen in zwei Planungsebenen auf. Damit sorgt er zum einen für eine Einschränkung der kombinatorischen Möglichkeiten und dadurch

für eine Reduktion des Suchraums, zum anderen werden durch die gegliederte Planung Suchprinzipien ermöglicht, die ein schnelleres Finden von Lösungen ermöglichen.

Die obere Ebene, die Prozessplanung, plant eine abstrakte Fertigung auf Domänenebene und legt eine konkrete Reihenfolge abstrakter Montageschritte über Domänentasks fest. Auf der unteren Ebene, der Fertigungsplanung, wird für einen gegebenen Domänentask nach gültigen technischen Tasks gesucht, die diesen erfüllen. Dabei bestimmt der Domänentask neben dem Zielzustand auch diejenigen Einzelteile, die von ihm beeinflusst werden. Die Planung wird nun auf solche weiteren Schritte reduziert, die sich auf mindestens eines dieser Einzelteile auswirken. Im Umkehrschluss vermeidet die Planung gezielt Pfade, deren Aktionen sich ausschließlich auf Domänentask-fremde Einzelteile auswirken. Die Suche zur Auflösung eines Domänentasks fokussiert sich sozusagen auf einen **Korridor relevanter Pfade**, wodurch sich der Planungsaufwand in exponentieller Weise reduzieren lässt. Wird beispielsweise ein Domänentask zum Setzen eines roten Steins geplant, so kann das „Bereitstellen“ des blauen Steins (und damit auch alle durch ihn nachgelagert ermöglichten Schritte) verworfen werden, da dies offensichtlich nichts mit dem zu erreichenden Ziel des Domänentasks zu tun hat.

Korridor relevanter Pfade

Abbildung 7.10 skizziert die Planung der oben genannten Legomontage über den beschriebenen Korridor-Ansatz. Die Prozessplanung legt zwei Pfade (A und B) an Domänentasks (große Rechtecke) fest, die im Wesentlichen die Reihenfolge der beiden Steine bestimmen. Jeder Domänentask weist einen der beiden Steine aus, den er beeinflusst. Die Prozessplanung entscheidet sich für Pfad A und übergibt den Domänentask an die Fertigungsplanung. Wie auch im vollständigen Suchbaum gibt es für den ersten Schritt die beiden Möglichkeiten, jeweils einen Stein über den Bereitsteller zur Verfügung zu stellen. Doch ist die Suche auf Aktionen beschränkt, die den roten Stein beeinflussen, somit wird das „Anbieten“ des blauen Steins verworfen. Selbiges trifft auch im nächsten Schritt zu und der Domänentask wird mit dem „Pick-and-Place“ des roten Steins erfolgreich abgeschlossen. Für den anschließenden Domänentask zum Setzen des blauen Steins ist die Planung sogar noch einfacher, da nur noch die notwendigen Fähigkeiten auf die bestehende Planungssituation angewandt werden können. Für die Montageplanung konnte damit eine durchgängige Lösung gefunden werden. Für Pfad B gestaltet sich die Planung analog.

Oftmals existieren allerdings mehrere Möglichkeiten, wie ein Domänentask mit technischen Tasks konkret gelöst werden kann. So kann ein Legostein von einem Roboter mit mehreren möglichen Greifpositionen an die Zielposition gesetzt werden, alternativ bietet vielleicht auch ein anderer Roboter weitere unterschiedliche Greifmöglichkeiten an. In der Theorie könnte alternativ auch ein 3D-Drucker den Legostein direkt an seine Zielposition drucken. Egal welche konkrete Lösung für den Domänentask gewählt wird, alle Lösungen haben eines gemeinsam: Das Zwischenprodukt erreicht immer den gleichen Fertigungszustand, der Legostein sitzt an derselben Position. Demnach erfüllen alle Nach-Produktsituationen der Lösungen stets die Ziel-Produktsituation des Domänentasks. Der einzige Unterschied liegt also lediglich in den Aktuatorattributen, die die Stellungen und Zustände der Roboter nach dem Setzen des Steins beschreiben.

Und hier kann eine weitreichende Annahme getroffen werden: Unterscheiden sich die Nach-Situationen von Alternativlösungen eines Domänentasks lediglich darin, dass

Roboter eine andere Position einnehmen, so hat dieser Unterschied keinen elementaren Einfluss auf den Erfolg der nachfolgenden Planung. Denn es kommen keine Deadlocks in der Planung vor, die lediglich auf die konkrete Position eines Roboters zurückzuführen sind. Es wird angenommen, dass sich dieser stets in die Position einer alternativen Lösung überführen lässt. Demnach ist ein Effekt lediglich in einer minimal erhöhten Ausführungsgeschwindigkeit des nachfolgenden Programms bemerkbar, was zugunsten einer effizienteren Planung akzeptiert wird. Ebenso wird außen vor gelassen, dass ein Roboter in einer ungünstigen Konstellation ggf. derart räumlich eingesperrt wird, dass ein „Freifahren“ nicht mehr möglich ist. Eine Ausnahme von der getroffenen Annahme bilden allerdings Roboter, die in der Nach-Situation noch an Einzelteile gebunden sind. Ein Beispiel stellt der Treppenbau aus LEGO mit zwei Robotern dar (siehe Abbildung 6.7). Um einen Legostein zu setzten, wird ein Roboter ausgewählt, der die Konstruktion geeignet unterstützt. Dieser ist auch nach Erledigung des Domänentasks an die Legostruktur gebunden, damit diese nicht zusammenfällt. Da der Roboter damit für die weitere Planung zunächst belegt ist, steht er im nächsten Schritt für keine andere Aktion zur Verfügung. In einem ungünstigen Fall kann dies dazu führen, dass eine Fortführung der Montage und damit eine Lösung der Planung nicht mehr möglich ist. Die Wahl des Roboters für Aufgaben, die ihn auch nach der Ausführung noch an Einzelteile bindet, kann also einen entscheidenden Einfluss auf die weitere Planung nehmen und über das Auftreten von Deadlocks bestimmen.

Für die Strategie des Planungsansatzes $PaRTs$ bedeutet dies konkret: Wird aufgrund eines Deadlocks (es existiert keine Automatisierungslösung für einen Domänentask) in der Planung zurückgegangen, so wird für den Vorgänger-Domänentask nur dann eine Alternativlösung gewählt und mit dieser die Planung erneut versucht, wenn deren Aktuatorattribute, die an Einzelteile gebunden sind, von denen der ursprünglichen Lösung abweichen. Nur dies verspricht, nicht in denselben Deadlock zu laufen. Dieser Vergleich der Lösungen von Domänentasks und die Unterscheidung nach Einzelteilgebundenen Aktuatorattributen wird **Pfadzusammenführung** genannt.

Pfadzusammenführung

Abbildung 7.11 illustriert ein Beispiel, wonach ein roter Legostein auf eine Konstruktion gesetzt werden soll, die zuvor geeignet zu unterstützen ist. Es existieren insgesamt vier unterschiedliche Pfade, wobei zunächst entweder das Anbieten des Steins oder der Support der Struktur mit einem von zwei unterschiedlichen Robotern erfolgt. Da durch den Support einer der Roboter belegt ist, ergibt sich der übrige Roboter für die Ausführung des „Pick-and-Place“. Bei den zwei linken der vier Pfade ist im Anschluss an den Domänentask der orange-farbene Roboter so lange belegt, bis in einem zukünftigen Domänentask der Support wieder aufgehoben wird. In den rechten beiden Pfaden betrifft dies den gelben Roboter. Wählt die Planung nun den ersten, am weitesten links stehenden Pfad zuerst und findet bei der anschließenden, ggf. aufwendigen Planung der späteren Domänentask keine Lösung, so würde ohne das Konzept der Pfadzusammenführung der Planer den zweiten Pfad des vorliegenden Domänentasks wählen. Dieser resultiert allerdings in derselben aufwendigen weiteren Planung, die kein Ergebnis liefern wird. Unter Verwendung der Pfadzusammenführung, die dank der zweischichtigen Planung möglich ist, kann stattdessen bereits bei der ersten Umplanung des Domänentasks gleich Pfad 3 gewählt werden, der potentiell zu einer Lösung führt.

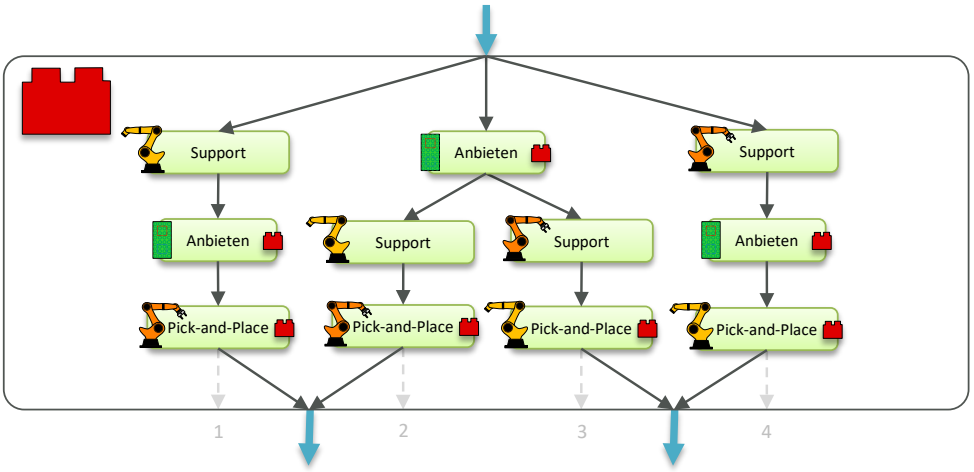


Abbildung 7.11. Beispiel der Pfadzusammenführung innerhalb eines Domänentasks zur Reduktion des nachgelagerten Planungsaufwands.

7.2.3 Komplexitätsabschätzung und Vergleich des Korridoransatzes

Durch die Separierung der Planung in zwei Ebenen und die damit erreichbare Fokussierung auf einen Korridor relevanter Pfade wird die Anzahl der möglichen Aktionen pro Situation stark eingeschränkt. Dadurch reduziert sich gleichermaßen auch die Anzahl an möglichen Zuständen, die die Planung zu bewältigen hat. Damit der Suchraum des Korridor-geführten Suchansatzes (im Folgenden mit „KORRIDOR“ benannt) mit dem einer klassischen Planung („VOLLSTÄNDIG“) verglichen werden kann, sei ein abstraktes Planungsbeispiel wie folgt definiert: Für die Montage einer Struktur sei n die Anzahl der Einzelteile des Bauteils. Vereinfachend sei angenommen, dass diese jeweils unabhängig voneinander montierbar sind. Für den vorgestellten Planungsansatz bedeutet n die Anzahl der zu erledigenden Domänentasks $t_{dom} \in \mathcal{T} \restriction Produkt$. Zur Montage eines jeden Einzelteils sei k die Anzahl der dafür notwendigen Aktionen auf Fähigkeitenebene und vereinfachend für jedes n gleich. Jede Fähigkeit bildet auf einen ggf. . hierarchischen technischen Task ab, daher entspricht k auch der Anzahl dieser Tasks $t_{tech} \in \mathcal{T} \setminus \mathcal{T} \restriction Produkt$, die für jedes Einzelteil erforderlich sind.

In einer ersten Betrachtung (im Weiteren „UNSORTIERT“ genannt) sei die Reihenfolge der k Aktionen pro Einzelteil irrelevant, d. h. jede Aktion darf zu einem beliebigen Zeitpunkt erfolgen. Dadurch ergibt sich eine gewisse Anzahl an kombinatorischen Möglichkeiten. In einer zweiten Betrachtung („SORTIERT“) sei die Reihenfolge der k Aktionen pro Einzelteil fest definiert, d. h. alle Aktionen eines Einzelteils sind in einer bestimmten Reihenfolge anzuwenden. Beispielsweise ist ein Legostein zunächst bereitzustellen, bevor er aufgenommen und an die Zielposition gesetzt werden kann.

Abschätzung: UNSORTIERT, VOLLSTÄNDIG

Für den vollständigen Suchraum ergeben sich in einem ersten Schritt $n * k$ mögliche Aktionen – also k Aktionen für jedes Einzelteil –, die in einen neuen Zustand führen. In

einem zweiten Schritt existieren für jedes der $n * k$ Möglichkeiten jeweils $(n * k) - 1$ weitere Möglichkeiten, eine davon wurde ja im vergangenen Schritt bereits erledigt. Für die insgesamt $n * k$ erforderlichen Schritte ergibt sich die folgende Berechnung für die Anzahl der möglichen Zustände in Gleichung 7.1. Für den kleinsten Fall $n = k = 1$, also ein Einzelteil und eine Aktion pro Einzelteil, wertet sich die Summe der Gleichung zu 1 aus, da $\sum_{i=1}^{1} \frac{1}{(1-i)!} = 1$. Nähert sich $n * k$ für eine große Anzahl an Einzelteilen und Schritten pro Einzelteil an unendlich, so konvergiert die Summe der Gleichung gegen die Eulersche Zahl e, da für ein beliebiges $x \in \mathbb{N}$ gilt: $\lim_{x\to\infty} \left(\sum_{i=1}^{x} \frac{1}{(x-i)!} \right) = e$. Somit kann für die Anzahl möglicher Zustände eine Abschätzung mit unterer und oberer Schranke getroffen werden:

$$|states_{voll}^{\sim}| = \sum_{i=1}^{kn} \frac{(kn)!}{(kn-i)!} = (kn)! \sum_{i=1}^{kn} \frac{1}{(kn-i)!} \tag{7.1}$$
$$(kn)! \leq |states_{voll}^{\sim}| \leq e * (kn)!$$

Für die Werte $k = [1, 10]$ und $n = [1, 10]$ ist eine Auswertung dieser Gleichung in Abbildung 7.12a grafisch dargestellt. Die Achse, die die Größe des Zustandsraums angibt, ist dabei logarithmisch gewählt. Bereits für kleine Werte für k und n lässt sich beobachten, dass der Zustandsraum enorm schnell – entsprechend der Fakultät – anwächst. Dabei sind beide Variablen in ähnlicher Weise für einen Anstieg verantwortlich.

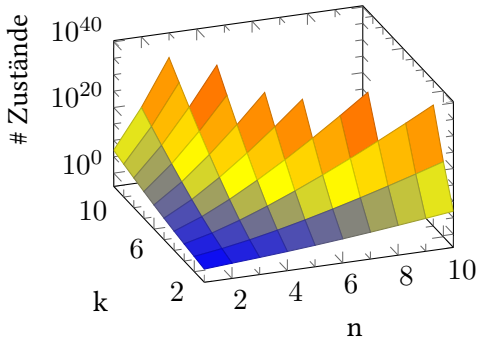
Die Anzahl möglicher Pfade (Reihenfolgen von Aktionen) des Zustandsraums bestimmt sich durch Kombinatorik mit „Ziehen ohne Zurücklegen mit Beachtung der Reihenfolge“. Dies ist gleich der Anzahl der Zustände im letzten Durchgang, die somit die Blätter aller möglichen Pfade darstellen:

$$|paths_{voll}^{\sim}| = (kn)! \tag{7.2}$$

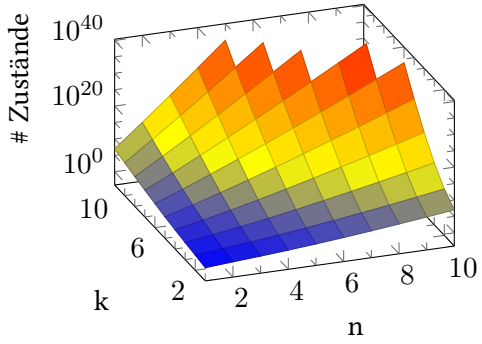
Die Anzahl der Pfade wächst demnach ähnlich zu der Anzahl an Zuständen mit einem Faktor von $[\frac{1}{e}, 1]$.

Abschätzung: UNSORTIERT, KORRIDOR (PaRTs)

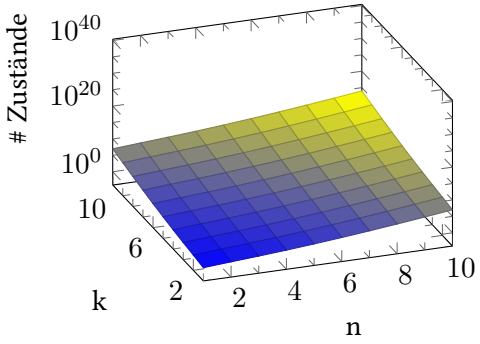
Für den Korridor-geführten Suchansatz werden zwei unterschiedliche Szenarien untersucht, in denen das Konzept der Pfadzusammenführung gänzlich oder gar nicht stattfindet. Zunächst wird der ungünstigste Fall untersucht, wonach das Konzept der Pfadzusammenführung in keinem Domänentask angewandt werden kann. Die Anzahl der Zustände wird erwartungsgemäß aufgrund des größeren Verzweigungsgrades also höher ausfallen. Zunächst wird von den Domänentasks einer ausgesucht, der die erste Korridorplanung vorgibt. Dafür existieren n Möglichkeiten. Zur Lösung des Domänentasks existieren für die k zu erledigenden Schritte insgesamt $k!$ Möglichkeiten, was in der Gesamtbetrachtung zu $n * k!$ Pfaden und $n * \left(\sum_{j=1}^{k} \frac{k!}{(k-j)!} \right)$ darin enthaltenen Zuständen führt. Nun wird der zweite Domänentask für die folgende Korridorplanung



(a) Graph zu Gleichung 7.1, vollständig mit unsortiertem k



(b) Graph zu Gleichung 7.3, Korridor mit unsortiertem k, ohne Pfadzusammenführung ($PaRTs$)



(c) Graph zu Gleichung 7.5, Korridor mit unsortiertem k und maximal möglicher Pfadzusammenführung ($PaRTs$)

Abbildung 7.12. Größe des Zustandsraums der Montage mit n als Anzahl an Einzelteilen und k als Anzahl notwendiger Aktionen pro Einzelteil. Dargestellt ist der Fall, in dem für die k Aktionen eines jeden Einzelteils keine Reihenfolge vorgegeben ist.

ausgewählt. Für jeden der bisherigen Pfade gibt es dafür $(n-1)$ Möglichkeiten (verbleibende Domänentasks). In dieser zweiten Korridorplanung existieren wiederum $k!$ mögliche Pfade zur Lösung des Domänentasks. Somit sind es in der Gesamtbetrachtung nach zwei gelösten Domänentasks nun $n * k! * (n-1) * k!$ mögliche Pfade und $n * \left(\sum_{j=1}^{k} \frac{k!}{(k-j)!}\right) + n * k! * (n-1) * \left(\sum_{j=1}^{k} \frac{k!}{(k-j)!}\right)$ Zustände. Führt man dies für alle n Domänentasks weiter, ergibt sich daraus folgende Formel in Gleichung 7.3 zur Berechnung der Anzahl der Zustände:

$$
\begin{aligned}
|states^{\sim}_{korr,nojoin}| &= \sum_{i=1}^{n} \left((k!)^{i-1} * \frac{n!}{(n-i)!} \right) * \sum_{j=1}^{k} \frac{k!}{(k-j)!} = \\
&= n! * \sum_{i=1}^{n} \frac{(k!)^{i}}{(n-i)!} * \sum_{j=1}^{k} \frac{1}{(k-j)!}
\end{aligned} \tag{7.3}
$$

Setzt man $k = 1$, so wertet die zweite Summe (mit Laufvariable j) in Gleichung 7.3 zu 1 aus. Für $k \to \infty$ nähert sich diese asymptotisch an e an. Mit einem Vergleich der ersten Summe (Laufvariable i) mit $(k!)^n$ konnte folgende Abschätzung für die Anzahl der Zustände über einschlägige Mathematik-Programme ermittelt werden:

$$n! * (k!)^n \leq |states^{\sim}_{korr,nojoin}| < (e+1) * n! * (k!)^n$$

Eine Graphische Auswertung von Gleichung 7.3 ist in Abbildung 7.12b dargestellt. Vergleicht man diese mit der Auswertung von Gleichung 7.1 in Abbildung 7.12a, so ist erkennbar, das die Zustandsmenge sowohl für zunehmende k als auch zunehmende n deutlich langsamer ansteigt.

Die Anzahl der möglichen Pfade für den Korridoransatz mit unsortiertem k ist gleich der Anzahl von Zuständen des letzten Schritts. Dazu können die Pfade innerhalb aller Domänentasks des letzten Schritts gezählt werden:

$$\begin{aligned} |paths^{\sim}_{korr,nojoin}| &= k!n * k!(n-1) * \cdots * k!(n-n) = \\ &= (k!)^n * n! \end{aligned} \tag{7.4}$$

Für den Fall, dass die Pfadzusammenführung optimalerweise lediglich einen ausgehenden Pfad pro Domänentask zulässt, gestaltet sich die Bestimmung der Zustandsmenge analog zu Gleichung 7.3 – allerdings mit der Ausnahme, dass der zusätzliche Verzweigungsgrad innerhalb der Domänentasks entfällt, denn hier existiert nur exakt ein Pfad. Der Planung stehen auf Domänenebene zunächst n mögliche Domänentasks zur Verfügung. Anschließend sind für jeden der bisherigen n Pfade jeweils weitere $(n-1)$ Domänentasks auswählbar. Dies wird für jeden der insgesamt n Schritte fortgeführt. Innerhalb eines Domänentasks existieren auch hier jeweils $\sum_{j=1}^{k} \frac{k!}{(k-j)!}$ Zustände. Die Gesamtanzahl an Zuständen ergibt sich demnach als:

$$\begin{aligned} |states^{\sim}_{korr,join}| &= \sum_{i=1}^{n} \frac{n!}{(n-i)!} * \sum_{j=1}^{k} \frac{k!}{(k-j)!} = \\ &= k! * n! * \sum_{i=1}^{n} \frac{1}{(n-i)!} * \sum_{j=1}^{k} \frac{1}{(k-j)!} \end{aligned} \tag{7.5}$$

Auch hier kann mit bereits genannten Eigenschaften eine untere und obere Abschätzung der Gleichung ermittelt werden:

$$k! * n! \leq |states^{\sim}_{korr,join}| \leq e^2 * k! * n!$$

Abbildung 7.12c stellt die Gleichung für den Bereich $n = [1, 10], k = [1, 10]$ graphisch dar. Es fällt auf, dass die Anzahl der Zustände gerade bei großen n und k im Vergleich zur Variante ohne Pfadzusammenführung nochmals drastisch reduziert wird. k und n bewirken jeweils unabhängig voneinander einen fakultätischen Anstieg – es gibt darüber hinaus keinen zusätzlichen Exponenten (siehe Gleichung 7.3) und keine Fakultät

über deren Produkt (siehe Gleichung 7.1). In der Realität wird die Pfadzusammenführung weder stets auf einen Pfad pro Domänentask reduzieren (Optimalfall) noch die komplette mögliche Verzweigung eines jeden Domänentasks beibehalten (schlechtester Fall). Dies ist abhängig vom konkreten Planungsproblem, der Domäne, der zur Verfügung stehenden Roboterzelle und dem konkreten Montageschritt. Gerade Montageschritte, die einer darüber hinausreichenden Fixierung des Bauteils bedürfen (z. B. Legotreppe), erlauben keine vollständige Pfadzusammenführung. Andere, in sich abgeschlossene Aktionen erlauben dahingegen oftmals eine komplette Reduktion auf einen Pfad. Sicher ist aber, dass die Pfadzusammenführung in den meisten Fällen einen bestimmten Grad an Verzweigungen reduzieren kann. Die Zustandsmenge wird für jedes Planungsproblem zwischen Gleichung 7.5 und Gleichung 7.3 liegen, was dank des Korridor-geführten Suchansatzes und der dadurch möglichen Pfadzusammenführung eine deutliche, unter Umständen enorme Verbesserung im Vergleich zum vollständigen Ansatz mit Gleichung 7.1 darstellt.

Die Anzahl der Pfade für den Fall maximaler Pfadzusammenführung ergibt sich auf Domänenebene aus einem „Ziehen ohne Zurücklegen mit Beachtung der Reihenfolge“, mit n Elementen. Innerhalb eines jeden Makroschritts existieren $k!$ mögliche Pfade. Insgesamt existieren in einem ersten Schritt auf Domänenebene damit $n * k!$ Pfade, in einem zweiten Schritt sind es jeweils $(n-1) * k!$ weitere Verzweigungen. Daher:

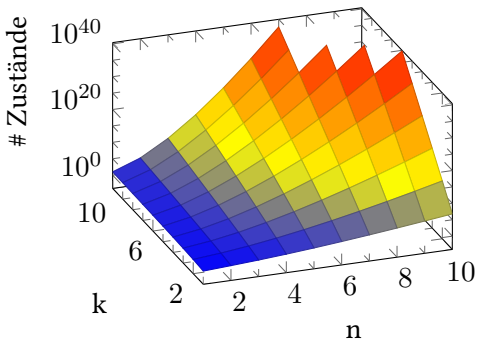
$$|paths^{\sim}_{korr,join}| = \prod_{i=1}^{n}(n+1-i)k! = k!n! \tag{7.6}$$

Auch anhand der Anzahl von Pfaden lässt sich im Vergleich zwischen Gleichung 7.6 und Gleichung 7.2 erkennen, dass der Korridoransatz eine hochgradige Reduktion der Plaungskomplexität ermöglicht.

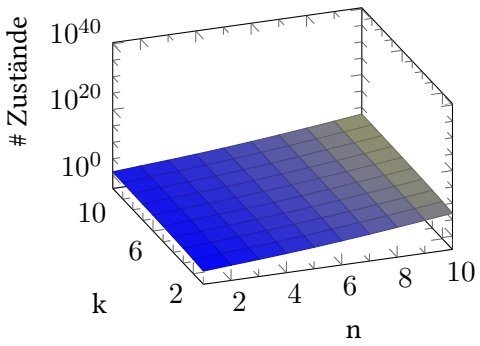
Abschätzung: SORTIERT, VOLLSTÄNDIG

Nun wird ein Szenario untersucht, in dem die Reihenfolge der k Aktionen pro Einzelteil mit einer festen Sortierung gegeben ist, also eine eindeutige Ordnung über die Aktionen existiert. Die Erfassung der Anzahl von Zuständen gestaltet sich hierfür jedoch schwierig, da in einem Baum, in dem Zustände die Knoten und Aktionen die Verzweigungen sind, sich der Verzweigungsgrad in verschiedenen Ebenen und in verschiedenen Pfaden sehr unterschiedlich verhält – je nachdem, von wie vielen der n Einzelteile bereits alle k Schritte erledigt sind. Beispiel: 4 Legosteine seien bereitzustellen und in einem Produkt zu verbauen. Nach den ersten 4 Schritten der Planung ist der unterschiedliche Verzweigungsgrad bereits ersichtlich. Wurden in einem Pfad alle 4 Legosteine bereitgestellt, so gibt es im nächsten Schritt 4 mögliche Aktionen: Das Setzen irgendeines dieser Steine. Wurden in einem anderen Pfad dagegen 2 der Legosteine sowohl bereitgestellt als auch gesetzt, so stehen im nächsten Schritt nur 2 Möglichkeiten zur Verfügung, nämlich das Bereitstellen eines der verbleibenden 2 Legosteine.

Für eine solche dynamische Entwicklung der Verzweigung ist das Finden einer exakten und einheitlichen Gleichung nicht trivial. Um den Suchraum des Korridor-geführten Suchansatzes dennoch gegen den vollständigen Fall abgrenzen zu können, wird für



(a) Graph zu Gleichung 7.7, vollständig mit sortiertem k



(b) Graph zu Gleichung 7.10, korridor mit sortiertem k ($PaRTs$)

Abbildung 7.13. Größe des Zustandsraums der Montage mit n als Anzahl an Einzelteilen und k als Anzahl notwendiger Aktionen pro Einzelteil. Dargestellt ist der Fall, in dem für die k Aktionen eines jeden Einzelteils eine eindeutige Reihenfolge vorgegeben ist.

letzteren eine Best-Case-Abschätzung überlegt: In den ersten k Schritten ist noch keines der n Einzelteile vollständig bearbeitet. Daher gibt es hier für jedes der n Einzelteile noch mindestens eine Aktion, es existieren insgesamt also mindestens n Nachfolgezustände in jedem Schritt. In den ersten k Schritten sind dies demnach $\sum_{j=1}^{k} n^j$ Zustände. In den Schritten k bis $2k$ ist höchstens eines der n Einzelteile vollständig bearbeitet, daher existieren zu jedem bisherigen der n^k Pfade in jedem weiteren Schritt jeweils mindestens $(n-1)$ weitere Nachfolgezustände, also $n^k * \sum_{j=1}^{k} (n-1)^j$. Dies wiederholt sich n-mal mit jeweils k Schritten. Zusammenfassend ergibt sich:

$$\begin{aligned} |states^{\rightarrow}_{voll}| &\geq \sum_{i=0}^{n-1} \left(\left(\tfrac{(n!)^k}{((n-i)!)^k} \right) \sum_{j=1}^{k} (n-i)^j \right) = \\ &= (n!)^k \sum_{i=0}^{n-1} \left(\tfrac{1}{((n-i)!)^k} \sum_{j=1}^{k} (n-i)^j \right) \end{aligned} \tag{7.7}$$

Die geschachtelte Summe über i nimmt für $n = 1$ und $k = 1$ den Wert 1 an. Für ein gegen unendlich gehendes n lässt sich über einschlägige Mathematik-Programme eine Annäherung der Summe an $(k+2)$ bestimmen. Daher kann für die Anzahl der Zustände eine untere Abschätzung angegeben werden:

$$|states^{\rightarrow}_{voll}| \geq (k+2) * (n!)^k$$

Eine graphische Auswertung von Gleichung 7.7 ist für $n = [1, 10]$ und $k = [1, 10]$ in Abbildung 7.13a gegeben. Auch hier ist ein schnelles fakultativisches Anwachsen des Zustandsraums ersichtlich, auch wenn dieser im Vergleich zu Gleichung 7.1 deutlich kleiner ausfällt. Aufgrund der Tatsache, dass die Variation der Reihenfolgen der notwendigen Einzelteilaktionen beschränkt ist, ist der Verzweigungsgrad des Suchbaums stark eingedämmt.

Die Anzahl möglicher Pfade kann für den vollständigen und sortierten Fall wiederum exakt berechnet werden. Ein Pfad ist als Reihenfolge der $k * n$ Aktionen zu betrachten, wobei die k Aktionen eines jeden Einzelteils darin strikt sortiert sein müssen. Bei einem Pfad der Länge $k * n$ gibt es für ein erstes Einzelteil $\binom{kn}{k}$ Möglichkeiten, seine k Aktionen auf diesen Pfad unter Einhaltung der Reihenfolge zu legen. Dies entspricht einem „Ziehen ohne Zurücklegen ohne Berücksichtigen der Reihenfolge“. Anschließend gibt es noch $k * (n-1)$ freie Stellen des Pfades, an denen die k Aktionen des nächsten Einzelteils stattfinden können: $\binom{k*(n-1)}{k}$. Um die Anzahl aller Pfade zu erhalten, sind die Möglichkeiten aller Schritte miteinander zu multiplizieren:

$$|paths^{\rightarrow}_{voll}| = \prod_{i=1}^{n} \binom{k(n-(i-1))}{k} = \frac{1}{(k!)^n} \prod_{i=1}^{n} \frac{(k(n-i)+k)!}{(k(n-i))!} \tag{7.8}$$

In jedem Fall gilt, dass ein Suchbaum gleich viele oder mehr Zustände als Pfade besitzt. Daher kann die Anzahl der Pfade in Gleichung 7.8 als eine untere Abschätzung der Zustandsmenge in Abschnitt 7.2.3 angesehen werden. Berechnet man allerdings beide Gleichungen für $n = 5$ und $k = 5$, so erhält man:

$$|paths^{\rightarrow}_{voll}| = 11.732.745.024 \gg 47.651.157 = |states^{\rightarrow}_{voll}| \tag{7.9}$$

Demnach wären es also deutlich mehr Pfade als Zustände. Durch eine simulierte vollständige Planung konnte zudem die Anzahl der tatsächlich verfügbaren Zustände auf $35.899.051.100$, statt der $47.651.157$ über die Gleichung errechneten Zustände, bestimmt werden. Das zeigt, dass die Abschätzung der Zustände in Gleichung 7.7 sehr ungenau ist und den tatsächlichen Wert deutlich unterschätzt. Dies ist jedoch akzeptierbar, wenn wir dennoch zeigen können, dass der Korridoransatz selbst diese Abschätzung deutlich unterschreitet. Dieser Fall wird als nächstes untersucht.

Abschätzung: SORTIERT, KORRIDOR (PaRTs)

Ähnlich wie beim unsortierten Korridor-Fall werden auch im sortierten Fall alle möglichen Permutationen an Domänentasks ermittelt und mit der Anzahl der jeweils darin vorkommenden Zustände multipliziert. Diese ist für alle Domänentasks gleich, da die k Aktionen in der einen möglichen Reihenfolge anzuwenden sind – es gibt also nur einen möglichen Pfad und k Zustände pro Domänentask. Das Konzept der Pfadzusammenführung spielt hierbei demnach keine Rolle. Die Anzahl möglicher Permutationen von Domänentasks summiert sich aus der jeweiligen Taskanzahl der einzelnen Schritte, die bereits in Gleichung 7.5 ermittelt wurde: $\sum_{i=1}^{n} \frac{n!}{(n-i)!}$. Für die Summe aller möglichen Zustände ergibt sich daraus Gleichung 7.10. Daraus lassen sich eine untere und eine obere Schranke ableiten:

$$\begin{aligned} |states^{\rightarrow}_{korr}| &= k * n! \sum_{i=1}^{n} \frac{1}{(n-i)!} \\ k * n! \leq |states^{\rightarrow}_{korr}| &\leq e * k * n! \end{aligned} \tag{7.10}$$

Tabelle 7.1. Größe des Zustandsraums und Anzahl der Pfade in Abhängigkeit der Anzahl an Einzelteilen n der Montage sowie der Anzahl von erforderlichen Aktionen k pro Einzelteil, im Fall sortierter k.

	n	1		2		3		4		5	
k		voll	korr.	voll	korr.	voll	korr.	voll	korr.	voll	korr.
1	$\|states\|$	1	1	4	4	15	15	64	64	325	325
	$\|paths\|$	1	1	2	2	6	6	24	24	120	120
2	$\|states\|$	2	2	18	8	270	30	7.363	128	$>32*10^4$	650
	$\|paths\|$	1	1	6	2	90	6	2.520	24	$>11*10^4$	120
3	$\|states\|$	3	3	68	12	5.247	45	$>11*10^5$	192	$>49*10^7$	975
	$\|paths\|$	1	1	20	2	1.680	6	$>36*10^4$	24	$>16*10^7$	120
4	$\|states\|$	4	4	250	16	$>11*10^4$	60	$>19*10^7$	256	$\gg 10*10^8$	1.300
	$\|paths\|$	1	1	70	2	$>34*10^3$	6	$>63*10^6$	24	$>30*10^{10}$	120
5	$\|states\|$	5	5	922	20	$>24*10^5$	75	$>35*10^9$	320	$\gg 14*10^{10}$	1.625
	$\|paths\|$	1	1	252	2	$>75*10^4$	6	$>11*10^9$	24	$>62*10^{13}$	120

Sehr deutlich ist hier zu sehen, dass die Anzahl k der Aktionen pro Einzelteil lediglich einen linearen Faktor der Zustandskomplexität ausmachen. Abbildung 7.13b verdeutlicht die Gleichung graphisch für die Wertebereiche $n = [1, 10]$ und $k = [1, 10]$. Im Vergleich zu Gleichung 7.7 (siehe Abbildung 7.13a) gibt es lediglich mit n ein fakultativisches Wachstum, k hingegen lässt den Zustandsraum lediglich linear anwachsen.

Da jeder Domänentask lediglich einen möglichen Pfad an technischen Tasks enthält, nämlich den mit der einen gültigen Reihenfolge, ist die insgesamte Anzahl an möglichen Pfaden gleich der Pfadanzahl auf Domänenebene. Diese bestimmt sich durch „Ziehen ohne Zurücklegen mit Beachtung der Reihenfolge“ in Gleichung 7.11.

$$|paths^{\rightarrow}_{korr}| = n! \tag{7.11}$$

Auch diese Gleichung spiegelt eine deutlich erkennbare Reduktion wider – sie ist gänzlich unabhängig von der Anzahl k der auszuführenden Aktionen pro Einzelteil.

Für den Fall „k sortiert“ wurden beide Ansätze, die vollständige und die Korridor-geführte Suche, simulativ ausgewertet und die möglichen Zustände bzw. Pfade mitgezählt. Tabelle 7.1 enthält die dabei erfassten Werte für $k = [1, 5]$ und $n = [1, 5]$.

Die Unterschiede der Planungsansätze zeigen sich schon bei sehr niedrigen Anzahlen von Tasks und Fähigkeiten, und werden überdeutlich schon bei etwas höheren Anzahlen von Tasks und Fähigkeiten: In Abschnitt 7.2.2 wurde ein einfaches Beispiel angeführt, in dem zwei Legosteine von einem Roboter nebeneinander auf eine Platte abzusetzen sind. In Abbildung 7.9 und Abbildung 7.10 sind die möglichen Abfolgen von Aktionen gegeben, wobei in der Planung für jede untersuchte Aktion (in den Abbildungen grün

eingezeichnet) ein Nachfolgezustand (Planungssituation) berechnet wird. Schlagen wir für dieses Beispiel mit $n = 2$ Steinen und $k = 2$ anzuwendenden Fähigkeiten pro Stein die Werte in Tabelle 7.1 nach: Es existieren 18 Zustände für den vollständigen bzw. 8 Zustände für den Korridor-geführten Suchansatz, wobei 6 bzw. 2 Pfade jeweils als Lösung möglich sind. Bei einem Planungsproblem mit beispielsweise $n = 5$ und $k = 3$ sind es bereits erheblich mehr: Knapp eine halbe Milliarde Zustände spannen bei einer vollständigen Suche den gesamten Zustandsraum auf, insgesamt 168 Millionen Pfade sind möglich. Beim Korridor-geführten Suchansatz existieren hingegen weniger als 2.000 Zustände, 120 Pfade sind möglich.

Die Gleichungen zur Bestimmung der Pfadanzahl (Gleichung 7.8 und Gleichung 7.11) sowie zur Bestimmung der Zustandsanzahl im Korridor-Szenario (Gleichung 7.10) entsprechen einer exakten Berechnung. Die darüber berechneten Werte stimmen mit denen der simulativen Erfassung überein. Lediglich für den Bereich $k = 5, n = [4, 5]$ konnte in akzeptabler Zeit simulativ kein Ergebnis mehr erzielt werden. In Tabelle 7.1 sind die entsprechenden Werte (mit „$\gg$“ angegeben) über die unterschätzende Gleichung 7.7 berechnet. Hier ist nochmal klar ersichtlich, dass die Gleichung so stark unterschätzt, dass sie sogar die Anzahl der möglichen Pfade deutlich unterschreitet.

Dadurch wird noch deutlicher, welchen Effekt der Korridoransatz und das Konzept der Pfadzusammenführung auf die zu bewältigende Komplexität der Planung haben. Mit den in Abschnitt 6.2.3 postulierten Annahmen, die bei einer Montage im Allgemeinen gelten, ist durch die Wegnahme von ungültigen Zuständen und durch das radikale „Abschneiden“ von Pfaden keine Einschränkung des Lösungsraums gegeben, sodass jeder gültige Montageablauf erhalten bleibt und über die der Suche auffindbar ist. Zusammenfassend können mit $PaRTs$ deutlich komplexere Planungsprobleme der Montage gelöst werden, als es mit klassischen Suchstrategien möglich ist.

7.2.4 Suchstrategie der Fertigungsplanung

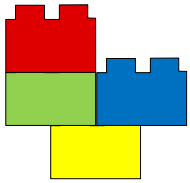
Der Korridor-geführte Suchansatz reduziert den Verzweigungsgrad des Suchbaums und verringert somit das Planungsproblem signifikant. Er legt allerdings keine Suchstrategie beim Durchwandern des Zustandsraums fest, wovon die Effizienz beim Finden eine optimaler Lösung jedoch unmittelbar abhängt. Der Planungsansatz wendet als Strategie auf Ebene der Fertigungsplanung die informierte Suche mittels A* an, da dies die Vorteile der Tiefensuche (Schnelligkeit) und die der Breitensuche (garantiertes Optimum) vereinen kann. Demnach wird diejenige Planungssituation unter allen bisher bekannten weiterverfolgt, deren Fitnesswert am geringsten ist. Dieser bestimmt sich aus den Kosten der Tasks, die ausgehend von der initialen Planungssituation zum Erreichen der aktuellen Planungssituation erforderlich sind, sowie den geschätzten Kosten derer Tasks, die vermutlich noch zum Erreichen der Ziel-Situation erforderlich sind. Zu beachten ist, dass die Strategie von A* bei zu hoch geschätzten Kosten sich der einer Tiefensuche annähern kann, sodass nicht zwangsläufig ein globales Optimum gefunden wird. Unterschätzt man hingegen die zukünftigen Kosten, so tendiert die Strategie zu einer Breitensuche und die Planungszeit erhöht sich.

Es ist nicht trivial, eine auf das konkrete Planungsproblem, auf deren Domäne und auf die Roboterzelle abgestimmte Heuristik zu konzipieren, die möglichst präzise zukünftige Kosten der Planung voraussieht. Um nicht einen übermäßigen Zusatzaufwand für Experten mit einer Modellierung der Fitnesswertbestimmung zu generieren, verwendet der Planungsansatz eine generische und abstrakte Berechnung, die einen Fitnesswert lediglich auf Basis von Attributen und technischen Tasks bestimmt. Denn dank der Korridor-geführten Suche ist der Planungsraum derart stark reduziert, dass sich eine Vereinfachung der Heuristik – die bei konsequenter Unterschätzung der Kosten im schlechtesten Fall eine Breitensuche darstellt – nicht allzu stark auf die Performanz von A* auswirkt. Als Fitnesswert werden zwei unterschiedliche Dimensionen betrachtet: Zum einen wird die Ausführungsdauer des Gesamtprogramms berücksichtigt, zum anderen spielt auch ein Qualitätsfaktor, also ein Maß für Güte und Genauigkeit der Ausführung der technischen Tasks durch die Roboter, eine Rolle. Als nächster Task wird derjenige selektiert, dessen Fitnesswert die beste Qualität ausweist. Existieren mehrere mit gleicher Qualität, so wird der mit der kürzeren Ausführungsdauer gewählt.

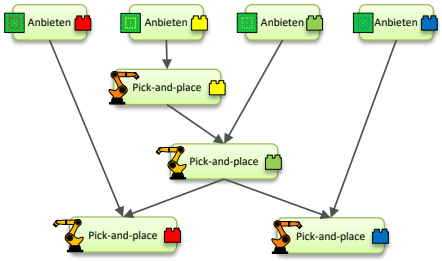
Die Kostenfunktion des Planungsansatzes berechnet die bis zum gegebenen Zustand notwendigen Kosten anhand der Gesamt-Ausführungsdauer und der Gesamt-Qualität aller Vorgänger an technischen Tasks. Für die Heuristik, die die Abschätzung der noch ausstehenden Kosten vornimmt, spielen die Attribute eine Rolle, die nach einem Vergleich der aktuellen Produktsituation mit der Ziel-Produktsituation noch umzusetzen sind. Jedes dieser Attribute beschreibt einen Prozesstyp, der im weiteren Verlauf der Planung irgendwann noch anzuwenden ist. Unter der Annahme, dass für jeden Prozesstyp eine ähnliche Folge an technischen Tasks notwendig sein wird, kann aufgrund bereits erfolgter Planungen der Prozesstypen eine Abschätzung der zukünftigen Kosten erstellt werden. Unbekannte Prozesstypen werden allerdings mit Nullkosten angerechnet, weshalb die Strategie zu Beginn der Planung in Form einer Breitensuche startet und sich im Laufe der Planung verbessert.

Nach jedem einzelnen Schritt der Fertigungsplanung wird aufgrund der manipulierten Attribute des technischen Tasks eine nachfolgend geltende Planungssituation abgeleitet. Dabei kann es bei einer realen Ausführung des technischen Tasks zu kollateral umgesetzten Attributen kommen, die in der Berechnung der technischen Tasks durch Fähigkeitenmodule nicht berücksichtigt sind. Wie bereits bei der Prozessplanung kann auch hier das Verfahren der Strukturanalyse (siehe Abschnitt 6.1.3) angewandt werden, um Kollateraleffekte von technischen Tasks zu ermitteln und diese in den Nachfolgezuständen der Planung zu berücksichtigen.

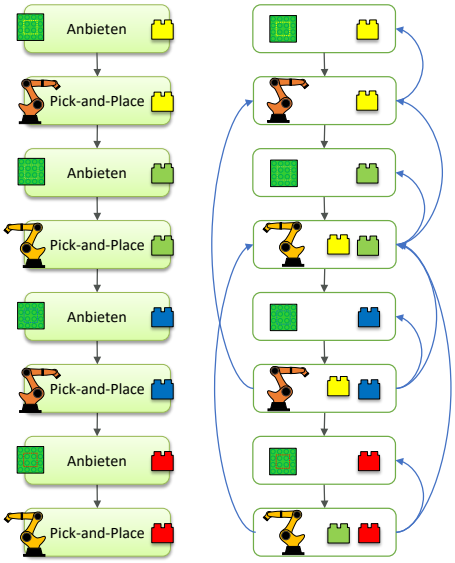
Die Schnittstelle zwischen Prozess- und Fertigungsplanung sieht die Rückgabe eines Iterators vor, der die Lösungen der Optimalität nach sortiert bereitstellt. Dazu wird die Suche über A* in einen Iterator eingebettet, der den Zustand der Planung intern speichert. Erst bei einer Abfrage der ersten Lösung, angestoßen über $hasNext()$ oder $next()$, wird der Algorithmus der Suche gestartet. Wird ein optimaler Ablauf an technischen Tasks identifiziert, dessen Nachfolge-Planungssituation die Ziel-Produktsituation erfüllt, so wird die Suche angehalten und der Ablauf als Lösung zurückgegeben. Bei wiederholtem Abruf der nächsten Lösung über den Iterator wird die Suche fortgesetzt und stellt somit die Lösungen dieser lokalen Suche auf Abruf zur Verfügung.



(a) Zu fertigende Legostruktur



(c) Abgeleiteter Ausführungsplan



(b) Beispiel eines Ausführungsplans, Abhängigkeiten seiner technischen Tasks

Abbildung 7.14. Parallelisierung von Ausführungsplänen am einem Beispiel mit LEGO (a). Anhand der Roboter und Einzelteile, die die technischen Tasks des Ablaufs belegen und beeinflussen, wird ein Abhängigkeitsbaum erstellt (b) und daraus ein parallelisierter Ausführungsplan abgeleitet (c).

7.3 Optimierung für Multiroboter-Anwendungen

Im Rahmen der Fertigungsplanung entsteht ein Ausführungsplan, der als Sequenz technischer Tasks gegeben ist. Die Tasks können in der Roboterzelle in der gegebenen Reihenfolge ausgeführt werden, wobei jeder von ihnen auf die Beendigung seines jeweiligen Vorgängers warten muss. Abbildung 7.14b zeigt ein Beispiel eines solchen Ausführungsplans an einem Beispiel mit LEGO, das eine einfache Struktur bestehend aus vier 2×2-Steinen zeigt (siehe Abbildung 7.14a). Es handelt sich dabei um ein mögliches Ergebnis von Schritt 6 des Planungsansatzes (der Überprüfung, siehe Abbildung 7.1). Als technische Ressourcen der Roboterzelle sind vier Bereitsteller gegeben, die analog zum Beispiel aus Abbildung 7.8 jeweils einen der vier Steine anbieten können. Zudem existieren zwei Roboter, die jeden Stein über ein „Pick-and-Place“ umpositionieren können. Im gegebenen Fall ist ersichtlich, dass besonders bei der Verfügbarkeit mehrerer Roboter eine strikt sequentielle Ausführungsreihenfolge der Tasks eine ungewollte Einschränkung darstellt. Denn einige Aktionen sind unabhängig voneinander und könnten parallel ausgeführt werden. Würde die strikte Sequenz des Programms aufgehoben und durch eine teilweise Parallelausführung der Programmteile ersetzt werden, so könnte die Ausführungszeit deutlich reduziert werden.

Um aus einem allgemeinen Plan ein verbessertes Roboterprogramm abzuleiten, wird über eine im Rahmen der Fertigungsplanung integrierte Optimierung die bisherige Reihenfolge von Tasks aufgebrochen und eine neue Ordnung der Ausführung bestimmt, die eine Parallelausführung unabhängiger Teile erlaubt. Dazu wird zunächst die hierarchische Verschachtelung von komplexen Tasks aufgelöst und sämtliche Einzeltasks werden mit ihren gegenseitigen Reihenfolgenbeziehungen in nur noch einer Ebene dargestellt. Jeder Einzeltask beschreibt über seine Attribute, die er auf- oder abbaut, auch die darin referenzierten Objekte, die er damit bei seiner Ausführung beeinflusst. Dies können Einzelteile oder technische Ressourcen der Roboterzelle sein. Für jeden Einzeltask werden diejenigen Vorgängertasks in der ursprünglichen Ausführungsreihenfolge bestimmt, die mindestens eine seiner beeinflussten Objekte ebenso als beeinflusst markieren. Zwischen diesen und dem Einzeltask werden Abhängigkeitsbeziehungen aufgestellt. Ebenso werden Reihenfolgenbeziehungen, die ursprünglich in der einheitlichen Produktbeschreibung zwischen den Verarbeitungsschritten der Einzelteile enthalten waren, berücksichtigt. Nach einer vollständigen Bestimmung aller Abhängigkeiten resultiert daraus eine partielle Ordnung über die Einzeltasks, die eine optimierte Ausführung des Roboterprogramms mit allen möglichen Parallelisierungen beschreibt. Dabei gilt, dass der ursprüngliche Ausführungsplan für diese partielle Ordnung eine gültige topologische Sortierung darstellt. Aus der partiellen Ordnung über die Einzeltasks wird ein neuer HalO-Task (Task, für dessen Untertasks eine Halbordnung existiert, siehe Abschnitt 7.1.3) formuliert, der das Endergebnis der Optimierung darstellt.

Am LEGO-Beispiel zeigt Abbildung 7.14b die technischen Tasks mit jeweils dem Stein, der im Fokus der Aktion steht, sowie dem Roboter, der diese Aktion ausführt. Zur Vereinfachung werden hier alle technischen Tasks der Sequenz als Einzeltasks angenommen – bei der Planung wurde von den Fähigkeitenmodulen also stets ein Einzeltask als Teillösung erzeugt. Der zweite Ablauf ordnet jedem technischen Task zusätzlich zu seinen unmittelbar beeinflussten Objekten all diejenigen Einzelteile und technischen Ressourcen zu, die über die Attribute des Tasks referenziert werden. Dies umfasst insbesondere auch den darunter liegenden Legostein, auf den abgesetzt wird. Die blauen Pfeile symbolisieren die Abhängigkeit von jeweils einem Task auf diejenigen seiner Vorgänger der ursprünglichen Sequenz, die mindestens eines seiner referenzierten Objekte beeinflussen. Abbildung 7.14c stellt den abgeleiteten Ausführungsplan für das gegebene Beispiel dar. Da der gelbe Legostein sowohl vom Pick-and-Place des grünen als auch des blauen Steins referenziert wird, können beide Aktionen nicht parallel zueinander stattfinden. Unter der vereinfachten Annahme, alle Tasks würden für sich betrachtet jeweils dieselbe Ausführungszeit erfordern, wird durch die Optimierung im gegebenen Beispiel die Ausführungszeit auf die Hälfte – von acht auf vier sequentiell abhängige Tasks – reduziert.

Zusammenfassung. Für die reale Ausführung eines erlangten Planungsergebnisses gibt es zwei Möglichkeiten: Die Anweisung von Robotern über einen Interpreter oder aber die Code-Generierung für spezifische Robotersteuerungen. Bevor ein offline geplantes Roboterprogramm jedoch in einer realen Fertigung eingesetzt werden kann, muss es einige Überprüfungen und Tests durchlaufen. Dazu eignet sich besonders eine visualisierte Simulation des Planungsergebnisses, über die ein Experte eine Einschätzung über das Programm erlangen kann. Dieses Kapitel beleuchtet beide Aspekte – Simulation und Ausführung – und beschreibt eine technische Umsetzung mithilfe des Roboterframeworks „Robotics API“.

8 Ausführung und Fertigung

Die Offline-Programmierung von Roboterapplikationen bietet gegenüber der Online-Programmierung, die eine Roboterzelle oft über einen längeren Zeitraum belegt, viele Vorteile. So können neue Applikationen parallel zu einer laufenden Produktion entwickelt werden, ohne dabei die Fertigung pausieren oder stoppen zu müssen. Um eine neue, offline programmierte Applikation im Vorfeld zum eigentlichen Produktionsstart ausgiebig zu testen und ggf. anzupassen, werden oftmals Visualisierungs- und Simulationstools eingesetzt. Diese stellen das Roboterprogramm einem Experten bildlich dar oder ermöglichen die Anwendung von Analysewerkzeugen zur Überprüfung bestimmter Prozessparameter. Abschnitt 8.1 stellt die Anbindung einer Simulationsumgebung an den Planungsansatz vor, die ein geplantes Roboterprogramm simulativ ausführen und visuell darstellen kann.

Soll das Roboterprogramm schließlich für die reale Montage eingesetzt werden, so ist ein präziser Abgleich zwischen virtuellem Modell und realer Roboterzelle besonders wichtig. Die Ausführung des Programms kann dann über einen Interpreter erfolgen, der sämtliche Einzeltasks des Ausführungsprogramms in entsprechende Ansteuerungen übersetzt und diese über dafür vorgesehene Schnittstellen an die Roboterzelle übermittelt. Alternativ dazu ist Code-Generierung möglich, worüber das modellierte Programmverhalten in eine proprietäre Programmiersprache eines Roboterherstellers exportiert wird. Damit ist eine Ausführung des Programms auf dem Roboter-eigenen Steuerungsrechner möglich.

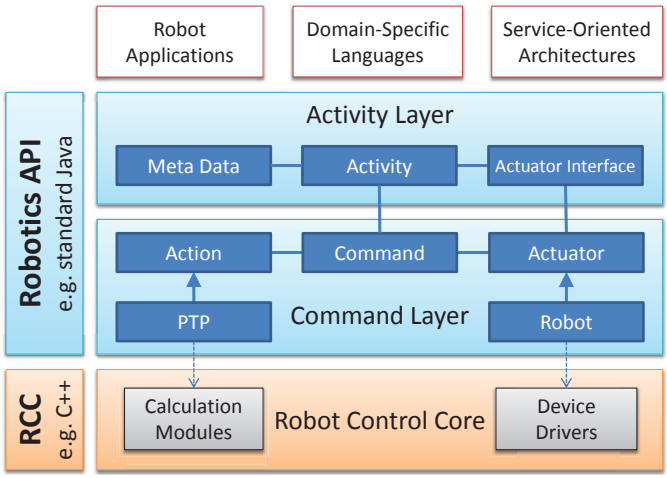


Abbildung 8.1. Der strukturelle Aufbau der Softrobot-Architektur. Insgesamt drei Schichten erstrecken sich über den Modellierungs- und Programmierteil (Robotics API) und den Ausführungsteil (Robot Control Core, RCC). [5]

Abschnitt 8.2 beleuchtet Techniken für den Abgleich des Modells, für die Interpretation von Programmen sowie die herstellerspezifische Code-Generierung und erklärt deren Umsetzung für den Planungsansatz $PaRTs$.

8.1 Simulation des Fertigungsprozesses

Für die Visualisierung des Montagezustands und der Roboterzelle während der Planung wie auch für die simulative Ausführung des gefundenen Roboterprogramms stützt sich der Planungsansatz $PaRTs$ auf die am Institut für Software & Systems Engineering der Universität Augsburg entwickelte Robotics API [5]. Das mehrteilige Software-Framework ermöglicht es, komplexe und echtzeitfähige Roboterapplikationen in einer objektorientierten Hochsprache und losgelöst von Echtzeitanforderungen zu modellieren. Dies gelingt, indem die Erstellung von Verhaltensmodellen (sog. *Activities*) über die zur Verfügung stehenden Aktuatoren und Sensoren auf einer nicht-echtzeitfähigen Programmierebene (**Robotics API**) stattfindet [2], die tatsächliche Ausführung der Programmmodelle hingegen auf einen echtzeitfähigen Steuerungsrechner (Robot Control Core, **RCC**) verschoben wird [122] (siehe Abbildung 8.1). Der RCC verfügt über eine echtzeitfähige Hardwareanbindung an alle Geräte der Roboterzelle.

Robotics API

RCC

Dem Programmierer stehen bei der Anwendungsentwicklung sämtliche Roboter als Objekte der Programmiersprache zur Verfügung, die ihm den Zugriff auf deren *Actuator Interfaces* – einer Art Fähigkeitendatenbank des Robotertyps – erlauben. Diese stellen Funktionen bereit, über die verschiedene Interaktionen des Roboters mit seiner Umwelt und den darin befindlichen Objekten erstellt werden können. Mehrere solcher auf diese Weise entstehenden Activities sind untereinander komplex verschachtelbar und können mit gegenseitigen Ausführungsbedingungen versehen werden, wie etwa dem Reagieren auf bestimmte Sensorwerte. Bei der Modellierung hilft auch das kartesische

Weltmodell, das mit statischen und dynamischen Verbindungsarten stets eine räumliche Zuordnung der Roboter und Objekte in der Roboterzelle sicherstellt. In einer tieferen Ebene besteht jede Activity aus sogenannten *Commands*, die feingranularere Aktionen auf einzelnen Aktuatoren beschreiben. Wird die Ausführung einer Activity angeordnet, so werden die darin verschachtelten Commands und deren Zusammenwirken in ein Datenfluss-basiertes Modell übersetzt und an den Steuerungsrechner (RCC) übertragen. Dieser sorgt mit einem garantierten Taktzyklus für eine echtzeitfähige Ausführung des Datenflussnetzes.

Visualisierung

Die Robotics API bringt einen eigenen Toolstack mit, der unter anderem eine clientbasierte **Visualisierung** enthält. Diese läuft als eigenständige Applikation und verbindet sich per Netzwerk mit der Robotics API. Über die Verbindung werden Informationen zum Weltmodell der Robotics API sowie Visualisierungsmodelle der einzelnen Objekte der Roboterzelle ausgetauscht. Damit die Robotics API stets den aktuellen Zustand der Roboterzelle an die Visualisierung weitergeben kann, bezieht sie selbst permanent über ihre Schnittstelle zum RCC den Zustand sämtlicher Aktuatoren und Sensoren der Roboterzelle und aktualisiert damit kontinuierlich ihr Weltmodell.

8.1.1 Anbindung an die Simulationsumgebung der Robotics API

R.API-Brücke

Der Planungsansatz $PaRTs$ stellt die Planung robotergestützter Fertigungsprozesse unabhängig von der Verwendung des Robotics API-Frameworks bereit. Soll der Planungsfortschritt jedoch planungsbegleitend visuell dargestellt werden, so ist eine Anbindung der Robotics API an die Planung notwendig. Der Planungsansatz bietet dazu eine Andock-Schnittstelle, über die er kontinuierlich über den aktuellen Planungszustand – also die derzeit gewählte Planungssituation – informiert. Technisch ist diese Schnittstelle über das Observer-Pattern umgesetzt, worüber sich externe Dienste für ein Abonnement zu Planungszustandsänderungen registrieren können. Eine sogenannte „Robotics API-Brücke“, kurz **R.API-Brücke**, die mit einer laufenden Instanz der Robotics API verbunden ist, registriert sich an dieser Observer-Schnittstelle und erhält somit Benachrichtigungen über den Planungszustand. Ihre Aufgabe ist es, die erhaltenen Informationen in das Weltmodell der Robotics API zu übersetzen, sodass dieses entsprechend modifiziert an die Visualisierung weitergegeben wird und diese damit den aktuellen Planungszustand anzeigt. In Abbildung 8.2 ist eine Übersicht über die Komponenten und deren Schnittstellen dargestellt.

Jede Zustandsaktualisierung, die der R.API-Brücke während der Planung seitens des Planers mitgeteilt wird, führt zu einer entsprechenden Aktualisierung des Weltmodells in der Robotics API. In der Visualisierung wird dem Benutzer demzufolge eine Abfolge an Einzelbildern präsentiert, wonach Roboter und Einzelteile sprunghaft ihre Position wechseln und das zu fertigende Bauteil im entsprechenden Fertigungszustand angezeigt wird. Nicht dargestellt werden dem Benutzer jedoch die konkreten Bewegungen der Roboter und die auszuführenden Prozesse. Eben diese gilt es im Anschluss an die Planung gleichermaßen darzustellen, um neben der Anzeige rein statisch visualisierter Planungssituationen auch einen Einblick in das Roboter- und Prozessverhalten des Ausführungsplans zu erlangen. Diese Aufgabe übernimmt ein entsprechend konzipierter

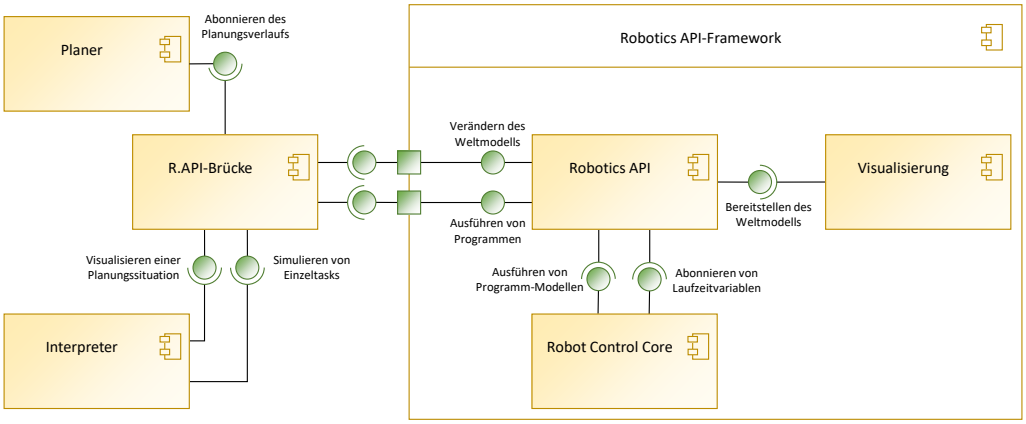


Abbildung 8.2. Die R.API-Brücke als Bindeglied zwischen Planer und Robotics API. Diese bietet unter anderem ihr Weltmodell an, das sie kontinuierlich mit Laufzeitdaten des Robot Control Cores aktualisiert. Die Visualisierung stellt dieses grafisch dar.

Interpreter

Interpreter, der einen verschachtelten komplexen Task auswertet und dessen Ausführungssemantik nachstellt. Zur simulativen Ausführung von Einzeltasks greift er dabei auf eine dafür angebotene Schnittstelle der R.API-Brücke zurück.

Wird ein simulativ auszuführender Einzeltask an die R.API-Brücke übergeben, überführt sie den Task in ein Programm der Robotics API und stößt dessen Ausführung über die Robotics API an (siehe Abbildung 8.2). Die Robotics API stützt sich bei der Ausführung auf den Robot Control Core, dem sie das auszuführende Programm in Form eines zyklisch auszuwertenden Datenfluss-Netzes („RPI-Netz“) übermittelt. Während der Ausführung schickt der Robot Control Core kontinuierlich seine Laufzeitvariablen wie aktuelle Roboterstellungen oder Sensorwerte, die die Robotics API in ihr Weltmodell einpflegt. Dieses dient dann unter anderem der Visualisierung als Eingabe. Doch soll der Einzeltask nicht real sondern simulativ wiedergegeben werden. Das Robotics API-Framework ermöglicht auf Ebene des Robot Control Cores, Bewegungsanweisungen entweder an echte Roboter über deren Kommunikationsprotokolle zu schicken oder alternativ – wie bei der simulativen Ausführung des Planungsergebnisses der Fall – an einen Simulationsadapter weiterzugeben. Simulationsadapter werden ebenso in einem zyklischen Takt ausgewertet und nähern den Verlauf ihrer Laufzeitvariablen (z. B. Achsstellungen oder -geschwindigkeiten) an die ihres Pendants der echten Ausführung an. Seitens des Robotics API-Frameworks wird der sogenannte „JavaRCC“ als Implementierung für den Robot Control Core angeboten, der basierend auf Java eine leichtgewichtige und überwiegend zu Simulationszwecken einsetzbare Variante darstellt. Sie leistet zwar keine Echtzeitgarantien für eine reale Ausführung, doch können bei simulativer Auswertung ideelle Taktschritte für ein realistisches Simulationsergebnis zugrunde gelegt werden.

Zusammengefasst übernimmt die R.API-Brücke zwei Aufgaben, um den Planungsansatz $PaRTs$ mit Funktionalitäten zum Ansprechen der Robotics API zu erweitern:

(1) Abonnieren des Planungsfortschritts und Übersetzen der erhaltenen Planungssituationen in entsprechende Änderungen des Weltmodells der Robotics API. Die Vorgehensweise hierzu sowie die Erläuterung des Weltmodells werden in Abschnitt 8.1.2 behandelt.

(2) Simulative Ausführung von Einzeltasks durch Übersetzen in ein Activity-Programm der Robotics API sowie Benachrichtigung über den Ausführungs-Status. Die Ausführungslogik des Interpreters zur Simulation von Planungsergebnissen auf Basis der Semantik komplexer Tasks ist in Abschnitt 8.1.3 beschrieben.

8.1.2 Visualisierung von Planungssituationen

Zu jedem Schritt der Planung ist die jeweils aktuelle Planungssituation über die Visualisierung der Robotics API wiederzugeben, um dem Benutzer eine Rückmeldung über den Planungsfortschritt zu geben. Intern verwendet die Robotics API dazu ein Weltmodell, das die verschiedenen Objekte der Roboterzelle in eine definierte geometrische Beziehung bringt. Es wird zwischen Objekten (*PhysicalObjects*), Koordinatensystemen (*Frames*) und Beziehungen (*Relations*) unterschieden. Ursprung des Weltmodells bildet der benannte Frame *World Origin*, zu deutsch Weltursprung. Davon ausgehend spannt sich ein Graph an Relations und Frames auf, wobei jede Relation zu je zwei Frames eine relative Transformation ausdrückt. Diese kann statisch und damit unveränderbar sein (*StaticRelation*), oder sie ist dynamisch und von der Laufzeit der Robotersteuerung abhängig (*DynamicRelation*).

Physikalische Objekte, die ein technisches Gerät der Roboterzelle oder auch nur einen einfachen Gegenstand repräsentieren können, besitzen mindestens einen benannten Basis-Frame *Base*. Dieser kann gleichermaßen über Relations in den Framegraphen eingebunden werden, womit dann auch das physikalische Objekt darin verankert ist. Ein Roboter besteht seinerseits aus mehreren physikalischen Teilen, wie seinen Gelenkarmen oder auch *Links* genannt, die seine Basis mit dem Flansch über eine sogenannte kinematische Kette verbinden. Abbildung 8.3 skizziert ein solches Beispiel eines Framegraphen, der einen Roboter enthält. Jeder Link besitzt zwei Frames, seine Basis und einen weiteren Frame, der die Stelle markiert, an der sich die Drehachse zum nächsten Link befindet. Zwischen beiden Frames definiert eine statische Relation deren Transformation. Sämtliche Links des Roboters werden gemäß seiner kinematischen Kette aneinandergereiht, indem der Basis-Frame mit dem Drehachsen-Frame des vorherigen Links verbunden wird. Diese Verbindung wird als dynamische Relation modelliert, die den Freiheitsgrad der dazwischenliegenden Drehachse abbildet. Konkrete Werte dynamischer Relationen sind erst zur Planungs- oder Ausführungszeit bekannt: Sie werden über die Laufzeitvariablen des Robot Control Cores, die die aktuellen Achsstellungen des realen oder simulierten Roboters enthalten, ermittelt. Die Werte für diese Transformationen sind für den Moment der Abfrage in der Roboterzelle gültig.

Der Framegraph ist Grundlage für geometrische Berechnungen und Bewegungsprofile innerhalb von Activities. Er dient gleichzeitig auch dazu, der Visualisierung die aktuellen Positionen der physikalischen Objekte im Raum mitzuteilen. Um ein physikalisches Objekt visuell darzustellen, kann ihm eine Visualisierungseigenschaft in Form eines

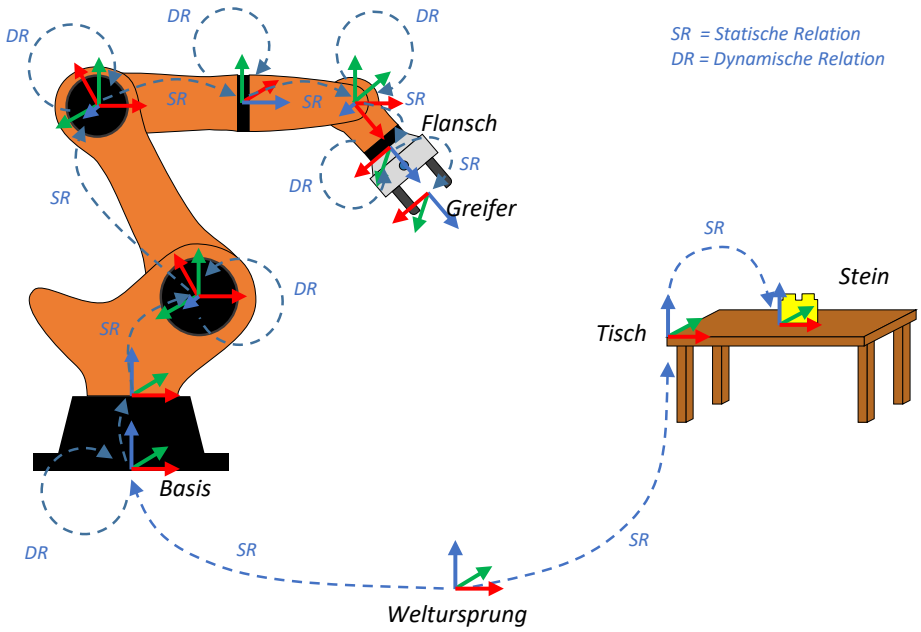


Abbildung 8.3. Beispiel für einen Framegraph als Weltmodell der Robotics API.

gängigen 3D-Beschreibungsformats hinzugefügt werden. Von der Visualisierung wird dieses logisch an den Basis-Frame des physikalischen Objekts angehängt und dort zur Anzeige gebracht. Über Veränderungen des Framegraphen, die unter anderem von den Laufzeitvariablen des Robot Control Cores abhängen, entspricht die visuelle Darstellung dem Zustand der Roboterzelle des RCCs.

Um Zustände des Planungsfortschritts über die Visualisierung der Robotics API anzuzeigen, muss die Planungssituation geeignet in das Weltmodell der Robotics API übersetzt werden. Diese Aufgabe übernimmt die R.API-Brücke, die über die Planungssituation zunächst alle geltenden Attribute sowie sämtliche darüber referenzierten Einzelteile listet. Die Einzelteile werden in der Robotics API verfügbar gemacht, indem ein physikalisches Objekt erzeugt und diesem das Visualisierungsmodell des Einzelteils zugewiesen wird. Um die physikalischen Objekte gemäß der Planungssituation räumlich auszurichten, wird für jedes Attribut eine entsprechende Relation im Framegraphen der Robotics API angelegt. Die Übersetzung von Attributen zwischen Einzelteilen, also von Attributen der Produktsituation, erfolgt nach folgendem Schema:

Beziehung zwischen Einzelteilen: Sie wird in eine entsprechende statische Relation der Robotics API übersetzt, die den entsprechenden räumlichen Versatz ausdrückt.

Eigenschaft eines Einzelteils: Da es sich hierbei nicht um eine geometrische Information handelt, wird sie nicht über Relations im Framegraphen abgebildet.

Fixpunkt eines Einzelteils: Er beschreibt eine fixe Position eines Einzelteils in der Welt. Die Position des entsprechenden physikalischen Objekts wird mit einer statischen Relation zum World Origin festgelegt.

Weiterhin existieren Attribute der technischen Ressourcen, die etwa eine konkrete Roboterpose oder das Greifen eines Objekts beschreiben. Roboter werden von der R.API-Brücke jedoch nicht wie Einzelteile in den Framegraphen eingepflegt. Stattdessen wird ein Roboter in der Robotics API als Objekt instantiiert, für das im Robot Control Core ein entsprechendes Gegenstück geladen und im Framegraph die vollständige kinematische Kette über Relationen aufgebaut wird. Dabei sind die dynamischen Relationen der einzelnen Achsen den entsprechenden Laufzeitvariablen des Robot Control Cores zugeordnet. Sollen die Achsstellungen des Roboters beeinflusst werden, so gelingt dies nur über eine entsprechende Ansteuerung des Robot Control Cores. Für Attribute der technischen Ressourcen gelten daher die folgenden Umsetzungen:

Belegung eines Roboters: Die Beziehung zwischen Endeffektor und einem Einzelteil wird über eine statische Relation zwischen dem Frame des Roboters (z. B. Flansch oder Greifer) und dem physikalischen Objekt mit der entsprechenden Transformation umgesetzt.

Konfiguration eines Roboters: Um die Transformation zwischen Basis und Flansch eines Roboters im Framegraphen auszudrücken, sind die dynamischen Relationen der Achsen über den Robot Control Core zu modifizieren. Zunächst werden die dafür einzustellenden Achspositionen über die inverse Kinematik des Roboters bestimmt. Über die Robotics API wird eine Activity erstellt und an den Robot Control Core zur Ausführung übergeben, die den Roboter an die entsprechenden Achspositionen springen lässt. Für reale Roboter ist dies zwar nicht möglich, simulierte Adapter des RCCs können allerdings eine solche Aktion zum sprunghaften Positionswechsel von Aktuatoren anbieten. Die geänderten Achsstellungen spiegeln sich anschließend im Framegraphen und damit in der Visualisierung wider.

Position eines Roboters: Die Position des Roboters in der Welt wird über eine statische Relation zwischen seiner Basis und dem World Origin mit der entsprechenden Transformation abgebildet.

Werden aufeinanderfolgend mehrere Planungssituationen zur Visualisierung übergeben, so sind Roboterobjekte in der Robotics API ggf. bereits vorhanden und müssen nicht erneut erstellt werden, oder zuvor erstellte Relationen sind womöglich wieder zu entfernen. Es ist die Aufgabe der R.API-Brücke, den derzeitigen Stand der Robotics API zu kennen und die zur Umsetzung einer Planungssituation erforderlichen Änderungen zu bestimmen.

8.1.3 Schrittweise Interpretation von Einzelprozessen

Im Anschluss an die Planung soll das resultierende Montageprogramm über die Robotics API simuliert und über ihre Anzeige visuell dargestellt werden. Auch hierfür stellt die R.API-Brücke entsprechende Funktionalitäten bereit, die das Ausführen von Roboterbewegungen über die Robotics API ermöglicht. Die logische Auswertung des zu simulierenden Tasks übernimmt dabei ein dafür konzipierter Interpreter. Der gesamte Verlauf zur Simulation des Tasks untergliedert sich dabei in drei logische Schritte:

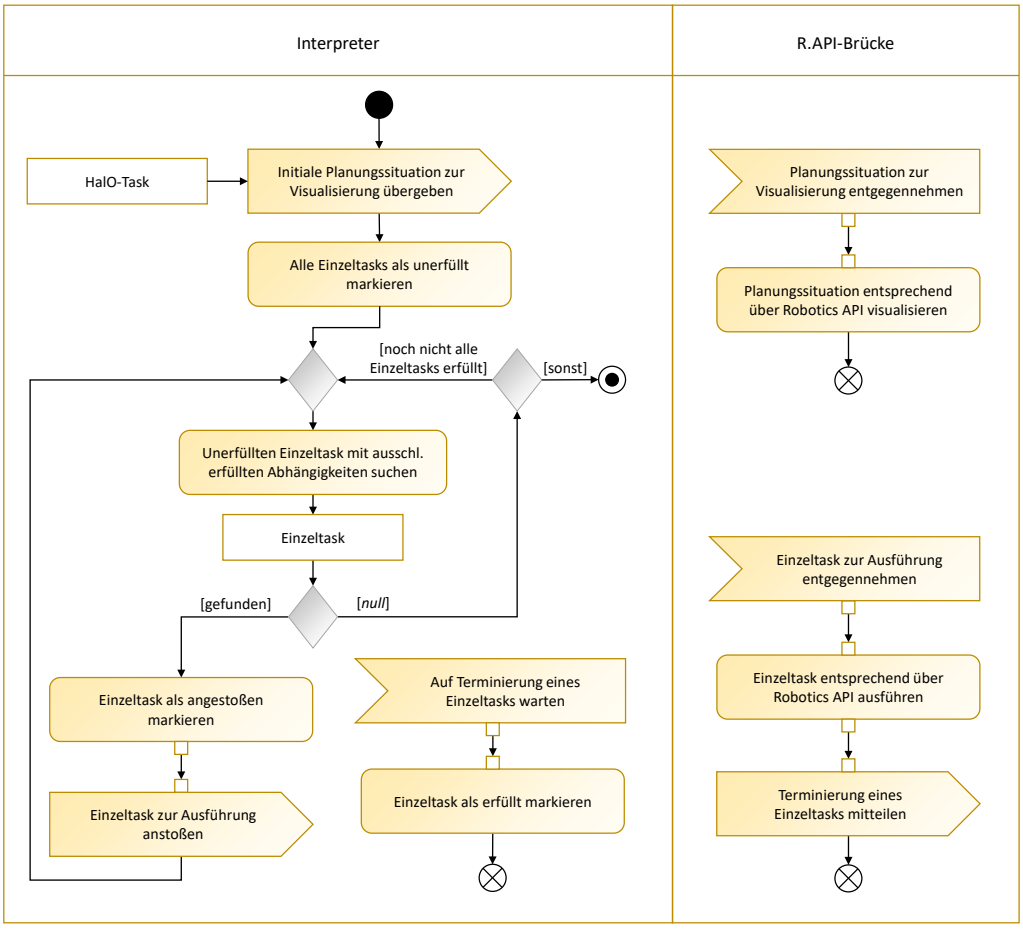


Abbildung 8.4. Darstellung des Ablaufs bei der semantischen Interpretation eines HalO-Tasks.

1. Herstellen der Ausgangssituation: Vor Beginn der simulativen Ausführung wird der Ausgangszustand des Bauteils und der Roboterzelle in der Robotics API hergestellt. Dazu beauftragt der Interpreter die R.API-Brücke mit der Visualisierung der initialen Planungssituation des übergebenen HalO-Tasks. Dies geschieht über das im vorigen Kapitel beschriebene Vorgehen.

2. Semantische Ausführung des HalO-Tasks: Da die Ausführung von Einzeltasks der R.API-Brücke obliegt, kümmert sich der Interpreter um eine korrekte Abbildung der übergeordneten Ausführungssemantik – also der Entscheidung, zu welchem Zeitpunkt welcher Einzeltask gestartet wird. Dazu wird der übergebene HalO-Task, der ggf. über Sequenz-, Parallel- oder HalO-Tasks (siehe Abschnitt 7.1.3) rekursiv verschachtelt ist, in einer abgeflachten Variante des HalO-Tasks betrachtet, der alle Beziehungen des ursprünglich verschachtelten Tasks in einer gemeinsamen globalen partiellen Ordnung beschreibt. Aus Sicht eines Einzeltasks ist in diesem HalO-Task unmittelbar ersichtlich, welche anderen Einzeltasks ausgeführt sein müssen, bevor er selbst gestartet werden darf. Diese Vorgänger werden vom Interpreter als seine Abhängigkeiten benannt. Abbildung 8.4 beschreibt das weitere Vorgehen des Interpreters. Zunächst werden alle

Einzeltasks als zunächst unerfüllt – also noch nicht ausgeführt und auch noch nicht gestartet – markiert. Der Interpreter sucht nun nach einem ersten Einzeltask, der keine Abhängigkeit zu einem Vorgänger besitzt, markiert diesen als angestoßen (gestartet) und übergibt ihn zur Ausführung an die R.API-Brücke. Anschließend führt er die Suche nach weiteren Tasks fort, die zum aktuellen Zeitpunkt gestartet werden dürfen, und stößt auch diese zur Ausführung an. Erhält die R.API-Brücke einen Einzeltask, so wird dieser von ihr über die Robotics API zur Ausführung gebracht. Terminiert die Ausführung auf Seite der Robotics API, so teilt die R.API-Brücke dies dem Interpreter mit, der daraufhin den beendeten Task als erfüllt (erfolgreich terminiert) markiert. Der Interpreter findet nun weitere Einzeltasks, deren Vorgänger bereits erfüllt sind und stößt auch diese zur Ausführung an. Sind alle Einzeltasks als erfüllt markiert, so terminiert der Interpreter. Nur im Falle eines Zyklusses in den Abhängigkeiten, beispielsweise wenn sich zwei Tasks A und B gegenseitig bedingen $A \rightarrow B, B \rightarrow A$, würde der beschriebene Algorithmus nicht terminieren. Für die Eingabe eines HalO-Tasks trifft dies allerdings nicht zu, denn dieser ist per Definition zyklenfrei.

3. Mapping zur Ausführung von Einzeltasks: Wird über die R.API-Brücke ein Einzeltask zur Ausführung angestoßen, so ist sie für eine einwandfreie Übersetzung der Tasksemantik in die Umgebung der Robotics API verantwortlich. Wie bereits für die Visualisierung statischer Planungssituationen ist es auch für die Darstellung dynamischer Programme zunächst erforderlich, dass zu Beginn der Ausführung eines Programms die darin angesprochenen Roboter als Aktuator-Objekte in der Robotics API und im Robot Control Core entsprechend geladen sind. Ist dies nicht der Fall, werden diese zunächst instantiiert. In einem zweiten Schritt werden die durch den Task beschriebenen Aktionen, meist Bewegungen des Roboters oder ein Öffnen/Schließen des Greifers, in für die Robotics API verständliche Activities übersetzt. Die R.API-Brücke verfügt dafür über ein Mapping, das eine Zuordnung von Einzeltasks wie PTPs oder LINs zu ihrem Pendant der Activities ermöglicht. Für alle Einzeltasks, die über die Aktions-Schnittstellen der technischen Ressourcen erzeugbar sind, bedarf es demnach jeweils einer solchen Zuordnung zu Activities, die von Experten zu definieren ist. Die über das Mapping abgeleitete Activity wird entsprechend der konkreten Konfiguration des Einzeltasks parametrisiert (z. B. mit der Zielposition oder der Geschwindigkeit der Bewegung) und an die Robotics API zur Ausführung übergeben. Vom Interpreter wird der weitere Verlauf der Activity-Ausführung überwacht. Während bei einer realen Ausführung durch externe Einflüsse durchaus Fehler oder etwa Notstops eintreten können, verläuft eine Simulation dagegen üblicherweise fehlerfrei. Terminiert die Ausführung auf der Seite der Robotics API, so ist vom Interpreter anschließend noch für eine konsistente Entsprechung des Weltmodells zu sorgen. Denn der Task benennt ggf. Attribute, die er auf- bzw. abbaut und damit das Weltmodell verändert, beispielsweise wenn ein Objekt vom Roboter gegriffen wird und sich damit geometrische Zusammenhänge verändern. Diese Veränderungen sind ebenfalls im Framegraphen der Robotics API über das Entfernen alter oder Hinzufügen neuer Relationen abzubilden. Dies geschieht auf analoge Weise zur Visualisierung von Planungssituationen.

8.2 Fertigung in einer realen Roboterzelle

Sind die Ergebnisse, die aus der Simulation der geplanten Abläufe gewonnen werden konnten, zufriedenstellend, kann das Roboterprogramm in der realen Roboterzelle zur Ausführung gebracht werden. Damit es sich dort ähnlich zur Simulation verhält und die Montageprozesse in einer hohen Präzision erfolgen, muss das Modell der Roboterzelle, auf dessen Basis die Planung erfolgt ist, exakt mit der realen Roboterzelle übereinstimmen. Abschnitt 8.2.1 skizziert das Vorgehen, welche vorbereitenden Maßnahmen im Vorfeld der Programmausführung nötig sind. Die sich anschließende eigentliche Ausführung von technischen Tasks ist im vorgestellten Ansatz auf zwei Arten vorgesehen: Eine Online-Ausführung (Abschnitt 8.2.2) wird über ein simultanes Auswerten des technischen Tasks und einer direkten Ansteuerung der Roboter realisiert. Demgegenüber steht die Ausführung über die Roboter-eigene Steuerung (Abschnitt 8.2.3), die ein Kompilieren des technischen Tasks in die proprietäre Programmiersprache des Roboterherstellers erfordert. Werden mehrere Roboter in einem Programm angesprochen, so ist die darin stattfindende Interaktion zwischen den Robotern über entsprechende Konstrukte dieser Programmiersprache abzubilden.

8.2.1 Anpassung des Zellenmodells: Vermessung, Teaching

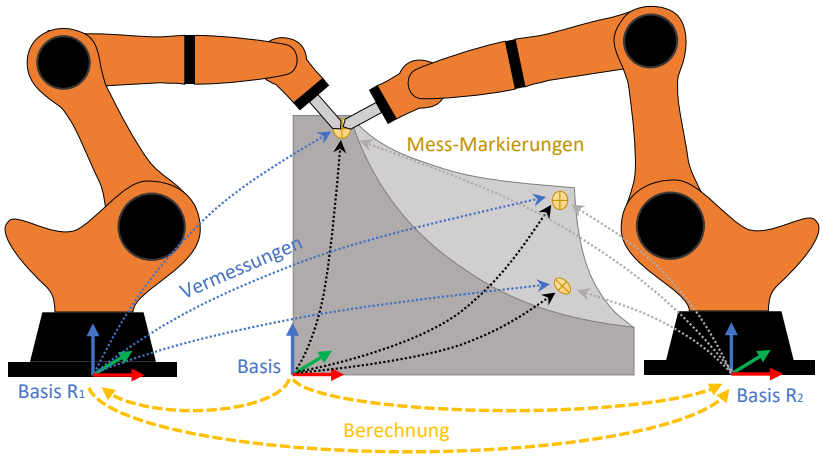
Damit die Simulation des Programms mit einer realen Ausführung möglichst genau übereinstimmt, muss das Modell der Roboterzelle, das für die Offline-Planung herangezogen wird, exakt mit der realen Roboterzelle abgeglichen werden. Dies ist möglich, indem der Experte die tatsächlichen Positionsdaten der Roboter und Objekte in die Vermessungs-Datenbasis der Roboterzellen-Definition einträgt und damit der Planung zur Verfügung stellt. Um diese Werte aus der realen Roboterzelle zu erhalten, gibt es verschiedene Möglichkeiten. Eine davon ist, Objekte über markante Punkte mit einem externen Tracking- und Vermessungstool anzupeilen und somit deren relative Position zueinander zu bestimmen. Eine andere, in der Praxis weitgehend übliche Variante ist, die in der Roboterzelle vorhandenen Roboter selbst als Vermessungstool zu verwenden. Hierbei werden die markanten Punkte der Objekte mit einem definierten Tool des Roboters manuell angefahren und deren Position über die eingenommene Stellung des Roboters mit seiner kinematischen Beschreibung berechnet.

Damit eine Vermessung von Objekten über einen Roboter erfolgen kann, bedarf es eines an den Roboterflansch montierten Tools, dessen Geometrie exakt bekannt ist. Oft wird als ein solches Tool – wie in Abbildung 8.5b gezeigt – eine Messspitze eingesetzt, da sie ein hochgenaues Anfahren von Punkten ermöglicht. Zu dieser ist die Bestimmung des sogenannten Tool-Centerpoints (**TCP**) notwendig, der die exakte Position der Spitze in Bezug zum Roboterflansch definiert. Im Rahmen der **Werkzeugvermessung** kann ein Experte diese Transformation bestimmen. Eine Möglichkeit dazu ist die von KUKA-Robotern verwendete **XYZ 4-Punkt-Methode** [39]. Mit insgesamt vier möglichst unterschiedlichen Roboterposen wird ein fester Punkt im Raum manuell angefahren. Über geometrische Berechnungen wird daraus eine eindeutige Transformation vom Flansch zum TCP abgeleitet, der damit die Geometrie des Tools beschreibt.

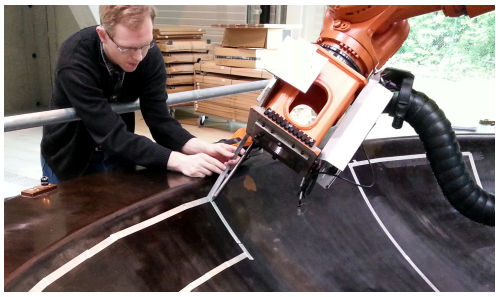
TCP

Werkzeug-vermessung

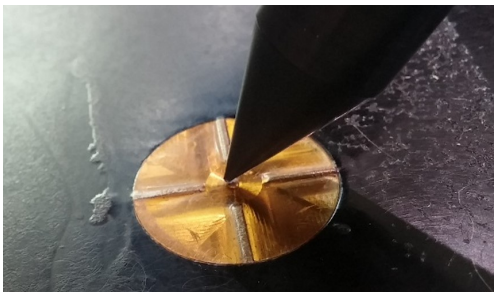
XYZ 4-Punkt-Methode



(a) Positions-Vermessung von Robotern und Form in einer Roboterzelle



(b) Montage einer Messspitze am Roboter



(c) Angefahrener Messpunkt der Form

Abbildung 8.5. Die Vermessung der Roboterzelle erfolgt über eine am Roboter montierte Messspitze und den an den Objekten der Roboterzelle ausgewiesenen Messpunkten.

Teachen von Koordinaten-systemen

3-Punkt-Methode

Mithilfe der geometrisch erfassten Messspitze können nun Punkte in der Roboterzelle relativ zum Roboter bestimmt werden. Mit der Vermessung von Punkten kann dann auch die Position samt Ausrichtung von Objekten der Roboterzelle eindeutig ermittelt werden, indem für diese ein eindeutiges Koordinatensystem festgelegt und **eingeteacht** (eingelernt) wird. Ein gängiges Vorgehen stellt dabei die **3-Punkt-Methode** dar, wonach der Ingenieur zunächst den zukünftigen Ursprung des Koordinatensystems mit der Messspitze anfährt, anschließend über einen Punkt auf der zukünftigen X-Achse diese festlegt und den verbleibenden Freiheitsgrad über das Anfahren eines beliebigen Punkts auf der positiven X-Y-Ebene einschränkt [1]. Alternativ kann, falls ein exaktes Modell des zu vermessenden Objekts vorliegt, auch die Position eines virtuellen Koordinatensystems des Objekts eingeteacht werden. Dies ist besonders dann sinnvoll, wenn die Geometrie des Objekts keine geeigneten Kanten oder Ebenen besitzt, oder wenn ein innerhalb des Objekt liegendes und unzugängliches Koordinatensystem als Basis dienen soll. Hierbei werden mit der Messspitze des Roboters mindestens drei markante Punkte des Objekts angefahren, deren relative Position zum virtuellen Basis-Koordinatensystem exakt bekannt ist – beispielsweise über ein Computermodell. Denn mit drei Punkten ist nicht nur die Position sondern auch die Orientierung eindeutig festgelegt. Abbildung 8.5c

zeigt ein Beispiel einer in das Objekt eingebrachten Markierung zur Verwendung als Messpunkt. Die über den Roboter ermittelten Messwerte werden aus der Steuerung ausgelesen und entsprechend verrechnet, sodass daraus der räumliche Bezug zwischen der Position des Basis-Koordinatensystems und der Basis des Roboters entsteht.

In Abbildung 8.5a ist eine Roboterzelle mit zwei Robotern und einer Form skizziert. Die Form besitzt ein Basis-Koordinatensystem sowie drei ausgezeichnete Mess-Markierungen, deren Transformation zur Basis jeweils fest und bekannt ist. Über das Anfahren aller drei Messpunkte mit der Messspitze kann die relative Position der Form zur Roboterbasis eindeutig berechnet werden. Erst nach allen drei erfolgten Vermessungen für jeden Roboter ist die Basis der Form jeweils eingeteacht und die jeweiligen geometrischen Abhängigkeiten können berechnet werden. Damit ist eine automatische Programmierung der Roboterzelle in einem Offline-Ansatz möglich.

Die Roboterzellen-Definition des Planungsansatzes erlaubt für die Vermessung von Objekten sowohl die Angabe eines eingeteachten Koordinatensystems in Form einer Transformation als auch die Verwendung der Vermessung über drei Markierungen. Im zweiten Fall werden in der Roboterzellen-Definition drei Mess-Markierungen für das Objekt relativ zu einem gemeinsamen Basis-Koordinatensystem spezifiziert. Die bei der Vermessung des Objekts ermittelten Daten für jede dieser Markierungen werden in Form dreier Transformationen direkt in die Vermessungs-Datenbasis übernommen, woraus sich automatisch die räumlichen Gegebenheiten für die Planung ableiten. Ein Plausibilitäts-Check vergleicht die Soll- und Ist-Streckenlängen zwischen jeweils zwei Mess-Markierungen. Wird dabei ein für das Objekt oder für das Planungsproblem individuell spezifizierbarer Schwellenwert für die Mindestgenauigkeit überschritten, wird die weitere Planung unter der Annahme verweigert, dass die damit geplanten Roboterbewegungen die geforderte Zielgenauigkeit nicht erreichen. War die Vermessung erfolgreich, bildet das erfasste Modell der Roboterzelle die reale Welt ausreichend gut ab und kann für eine (erneute) Offline-Planung verwendet werden.

8.2.2 Online-Ausführung über die Robotics API

Basiert ein geplantes Roboterprogramm auf einem realistischen Zellenmodell, d. h. die Vermessung der Roboterzelle ist mit einer geforderten Genauigkeit erfolgt, so ist das erwartete Verhalten einer realen Ausführung äquivalent zur Simulation. Da eine Online-Ausführung von Tasks ebenfalls dieselbe Ausführungslogik benötigt wie die Simulation, kann hierfür das Vorgehen der simulativen Ausführung (siehe Abschnitt 8.1) wiederverwendet werden. Die Semantik komplexer Tasks wird auch hier vom Interpreter ausgewertet, und die Ausführung von Einzeltasks wird über die R.API-Brücke an die Robotics API weitergereicht.

Die Robotics API ermöglicht es dank ihrer Abstraktionsebenen dem Anwender, einzelne Komponenten auszutauschen, ohne dass sich die Programmierschnittstelle dabei ändert. Für eine reale Ausführung des Roboterprogramms wird daher der leichtgewichtige JavaRCC durch einen echtzeitfähigen RealtimeRCC ersetzt, der auf einem dedizierten Linux-Rechner läuft. Anstelle von Roboter-Adaptern, die im JavaRCC lediglich ihren internen Zustand interpolieren, werden bei einer Ansteuerung über den RealtimeRCC

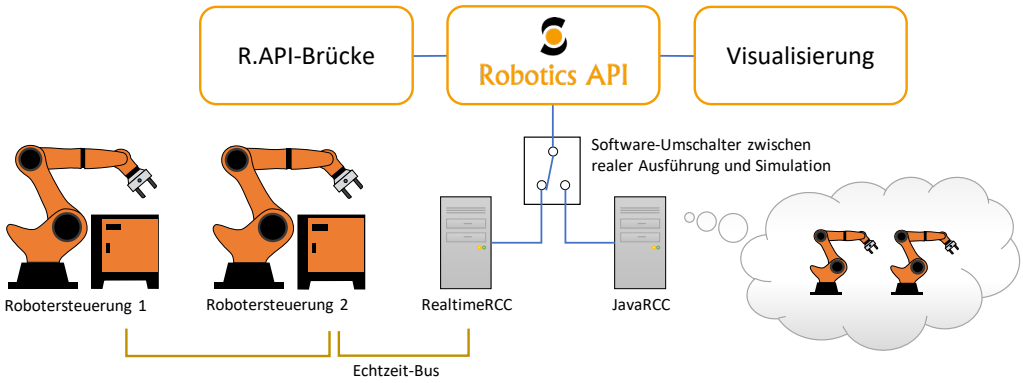


Abbildung 8.6. Die Robotics API als umschaltbare Abstraktion von einer realen Ausführung und einer Simulation.

hingegen echte Roboter angesprochen. Dazu wird für jeden Roboter ein sogenannter Treiber im Robot Control Core geladen, der in gleichbleibender Taktfrequenz über einen verfügbaren Kommunikationsbus wie CAN [32] oder EtherCAT [31, 60] mit der Steuerung des Roboterherstellers kommuniziert. Abbildung 8.6 stellt den Zusammenhang zwischen JavaRCC und dem RealtimeRCC sowie den echten Robotersteuerungen schematisch dar, wobei über einen Software-Umschalter zwischen realer Ausführung und Simulation umgeschaltet werden kann. Die Robotics API verbindet sich mit dem entsprechend gewählten Robot Control Core, wobei ihre Schnittstelle zur R.API-Brücke und zur Visualisierung davon unberührt bleibt.

Als einziger Unterschied zur Simulation ist bei einer realen Ausführung ein Springen von Aktuatoren, wie etwa beim Anzeigen einer Planungssituation, nicht möglich. Insbesondere das Herstellen der initialen Planungssituation vor der Task-Ausführung, das bei der Simulation wie auch bei der realen Ausführung eine Rolle spielt, muss daher auf anderem Wege erfolgen. Statt einer instantanen Veränderung des Weltmodells überführt die R.API-Brücke die Roboterzelle über einen Prozess in die Ausgangssituation, indem der Reihe nach jeder Roboter eine PTP-Bewegung an seine jeweilige Startposition ausführt. Bei dieser initialen Fahrt bedarf es eines besonderen Augenmerks des Experten, denn aufgrund des unbekannten vorherigen Zustands sind Kollisionen möglich. Ist die initiale Planungssituation hergestellt, findet die weitere Ausführung in der realen Roboterzelle analog zur Simulation, wie in Abschnitt 8.1.3, beschrieben statt.

8.2.3 Code-Generierung zur Ausführung auf proprietären Plattformen

Oftmals kommt bei der industriellen Fertigung eine interpretierende Ausführung des Roboterprogramms auf einem externen Steuerungsrechner wie dem Robot Control Core, der dabei die volle Kontrolle über sämtliche Roboter besitzt, nicht infrage. Groß sind Vorbehalte gegenüber solch neuartigen Systemen wegen der Sorge, ein Ausfall könne die Produktion lahmlegen und dadurch hohe Ausfallkosten verursachen. Es wird außerdem befürchtet, dass im Fehlerfall aufgrund des Mangels an Fachwissen über das neue System eine Behebung nicht ausreichend schnell erfolgt. Anders ist dies bei den Steuerungen

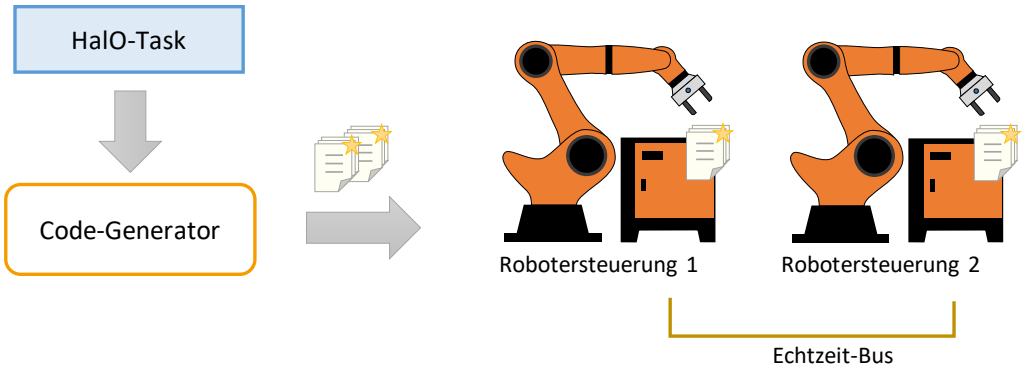


Abbildung 8.7. Code-Generierung für eine verteilte Ausführung eines geplanten Programms auf dedizierten Robotersteuerungen.

KRL

der Roboterhersteller und deren proprietären Programmiersprachen. Bei KUKA ist dies beispielsweise die KUKA Robot Language (**KRL**), der schweizer Roboterhersteller Stäubli bietet die hauseigene VAL 3 Programmiersprache an, von ABB wird die Sprache RAPID zur Roboterprogrammierung ihrer Roboter verwendet. Diese Systeme haben sich über viele Jahre hin bewährt; ihrer Funktion wird vertraut und die Wahrscheinlichkeit sowie die Folgen eines Ausfalls scheinen dem Betreiber weitgehend kalkulierbar. Zudem hat er angestelltes Fachpersonal, das mit dem eingesetzten System vertraut ist und das aufgrund seiner Erfahrung mit eventuellen „Macken“ umzugehen weiß. Tritt ein Fehler auf, der die Produktion stilllegt, kann schnell und gezielt eingegriffen werden, um die Produktion schnellstmöglich wiederaufzunehmen.

Code-Generator

Aus diesem Grund soll es möglich sein, einen über den Planungsansatz $PaRTs$ erstellten technischen Task in Roboter-spezifischen Programmcode zu überführen anstatt ihn über einen Online-Interpretationsansatz auszuführen. Abbildung 8.7 verdeutlicht das Vorgehen bei der Code-Generierung. Sämtliche Robotersteuerungen der in der Zelle befindlichen Roboter sind über einen echtzeitfähigen Kommunikations-Bus verbunden, der eine koordinierte Ausführung ermöglicht. Über einen **Code-Generator**, den der Planungsansatz zur Verfügung stellt, wird das geplante Roboterprogramm (HalO-Task) in ein gleichbedeutendes, dem proprietären Format der Hersteller-eigenen Programmiersprache entsprechendes Programm kompiliert. Im Falle mehrerer für die Fertigung verwendeter Roboter entsteht ein solches Programm je Steuerung. Die Programme werden von einem Inbetriebnehmer auf die Robotersteuerungen geladen und können dort zur Ausführung gebracht werden.

Lässt man zunächst die Ansteuerung mehrerer Roboter außer Acht, ist eine Programmdatei im roboterspezifischen Format zu erzeugen, die sämtliche Einzeltasks des HalO-Tasks sowie deren zeitliche Abhängigkeiten berücksichtigt. Für jeden Einzeltask wird dazu ein entsprechender Programmteil erzeugt, der beispielsweise eine Bewegung des Roboters über einen adäquat parametrisierten PTP oder LIN sowie eine Aktion wie das Öffnen und Schließen des Greifers darstellt. Für parallel auf einer Steuerung auszuführende Einzeltasks kann dabei das Konzept der sogenannten Trigger verwendet werden, über die z. B. zu einer PTP-Bewegung des Roboters eine gleichzeitige Ansteuerung des Grei-

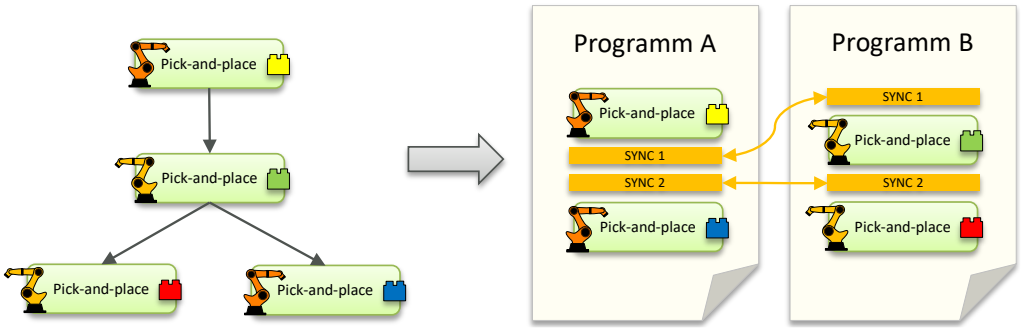


Abbildung 8.8. Beispiel der Code-Generierung für mehrere Steuerungsrechner mit synchronisierten Ablaufplänen.

fers möglich ist. Prinzipiell kann, wenn ein Greifer über das Setzen eines Output-Werts angesteuert wird, dieser auch vor Beginn der Bewegung entsprechend gesetzt werden und im Anschluss der Bewegung wird über einen Wartebefehl über einen gegebenen Input auf die Fertigstellung der Greiferaktion gewartet. Da das Setzen von Output-Werten instantan erfolgt, ist das nachgelagerte Schalten mehrerer Outputs der parallelen Ausführung gleichbedeutend.

Findet die Code-Generierung für einen HalO-Task mit mehreren Robotern statt, so wird nicht nur ein einzelnes Dokument sondern je Steuerung ein separates Dokument erstellt. Jede Programmdatei enthält dabei die Programmteile der von der jeweiligen Steuerung kontrollierten Geräte. Um eine zeitliche Kopplung bei der Ausführung zu ermöglichen, damit eine Steuerung ggf. auf die Beendigung einer Aktion einer anderen Steuerung wartet, ist in den meisten Roboter-Programmiersprachen über eingebaute Konzepte der Synchronisierung möglich. KUKA bietet über das Erweiterungspaket KUKA.RoboTeam [74] einen Programmierbefehl *PROGSYNC* an, über den gemeinsame Warte- und Fortsetzungsstellen in allen beteiligten Roboterprogrammen definiert werden können. ABB verfügt mit dem Befehl *WaitSyncTask* in seiner Programmiersprache RAPID über ein analoges Konzept. Ein Beispiel für die Verwendung von synchronisierten Wartestellen ist in Abbildung 8.8 gegeben. Abhängigkeiten zwischen Einzeltasks, deren Ausführung auf unterschiedlichen Robotersteuerungen erfolgt, werden über gekoppelte Wartestellen (SYNC 1 und SYNC 2) in den einzelnen Programmen realisiert. Bei Erreichen einer Wartestelle wird mit der Fortführung darauf gewartet, dass der andere Teilnehmer ebenfalls die entsprechende Wartestelle erreicht. Die Fortsetzung des Programms erfolgt auf beiden Steuerungen gemeinsam.

Für jeden Roboterhersteller kann ein spezieller Generator zur Verfügung gestellt werden, der einen HalO-Task in das entsprechende Format der herstellerspezifischen Programmiersprache übersetzt. Generell kann ein proprietäres Roboterprogramm jedoch nur dann generiert werden, wenn sämtliche verwendete Aktionen der Einzeltasks von der Robotersteuerung unterstützt werden. Je nach Implementierung können sogar kooperativ von mehreren Robotern ausgeführte Aktionen (siehe Explizite Kooperation, Abschnitt 4.2) in die Zielsprache übersetzt werden, indem spezielle Programmierkonzepte wie geometrische Kopplungen (bei KUKA: GEOLINK) gezielt eingesetzt werden.

Zusammenfassung. Der vorgestellte Planungsansatz $PaRTs$ wird anhand eines Beispiels mit LEGO evaluiert. Dieses Kapitel beschreibt zum einen die konkreten Szenarien, für die eine automatische Planung durchgeführt wird. Zum anderen wird dargelegt, welche domänenspezifischen Erweiterungen für den Planungsansatz zur Durchführung dieser Fallstudie notwendig sind. Die mit unterschiedlichen Evaluationsläufen erzielten Ergebnisse werden erfasst und statistisch ausgewertet. Einzelne Planungsergebnisse werden sowohl über die simulative als auch reale Ausführung untersucht.

9

Evaluation der Fallstudie 1: Montage von Strukturen mit LEGO

Der Planungsansatz $PaRTs$ ermöglicht ein durchgängiges und toolgestütztes Vorgehen bei der Automatisierung von Montageprozessen mit der Möglichkeit zur Simulation und zur realen Ausführung in einer Roboterzelle. Experten aus Domäne, Prozess und Automatisierung werden dabei in die Lage versetzt, unabhängig voneinander und generisch ihr Wissen zur Planung beizutragen, sodass dieses zur Beschreibung eines Montageproblems dienen und auch in die konkrete Umsetzung in der Roboterzelle einfließen kann. Ein Anwendungsgebiet im Bereich der Montage, auf Basis dessen die Tauglichkeit und

Anwendbarkeit des Planungsansatzes in einer ersten Versuchsreihe evaluiert wird, ist das Bauen mit LEGO.

Kernaspekte der Evaluation sind Untersuchungen, wie sich die Integration von Montageprozessen der Domäne LEGO in den Planungsansatz verhält und inwieweit die vorgestellte Planung mit den benannten Herausforderungen der Domäne zurechtkommt. Weiterhin liegt die Anwendbarkeit auf Multiroboter-Problemstellungen im Fokus, die insbesondere ein Zusammenspiel der Roboter in Form von impliziten Kooperationen erfordern, ebenso die nachgelagerte Optimierung der Ergebnisse im Hinblick auf eine parallelisierte Ausführung. Untersucht wird auch die Verwendung der Simulation von geplanten Monatageprozessen mit mehreren Robotern. Die Ausführung in einer realen Roboterzelle wird für ein Montagebeispiel mit einem Roboter evaluiert.

In Abschnitt 9.1 werden die für die Evaluation herangezogenen Montagemodelle und die dabei verwendeten Legosteine, in Abschnitt 9.2 der Aufbau der Roboterzelle sowie das Vorgehen bei deren Vermessung beschrieben. Neben einer Erläuterung des Bauplans werden in Abschnitt 9.3 sämtliche Erweiterungen vorgestellt, die dem Planungsansatz in Form von roboter- und domänenspezifischen Experten-Modulen zur Lösung des Planungsproblems beigesteuert werden. Die Auswertung über die Anwendung des Planungsansatzes auf die Domäne LEGO und die erzielten Ergebnisse werden in Abschnitt 9.4 vorgestellt.

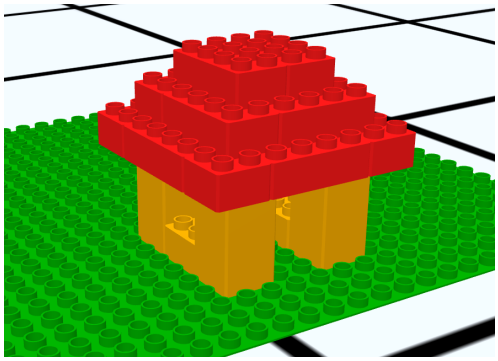
9.1 Zu fertigende Produkte aus LEGO

Insgesamt zwei Produkt-Strukturen, zu denen ein Roboterprogramm zur Montage geplant werden soll, werden für die Evaluation des Planungsansatzes mit der Domäne LEGO genauer betrachtet. Abschnitt 9.1.1 beschreibt das LEGO-Haus, das die Planung vorrangig im Hinblick auf die Ecksteinproblematik untersucht. Die LEGO-Brücke, die noch weitere Eigenschaften als Herausforderung für die Planung aufweist, ist in Abschnitt 9.1.2 beschrieben.

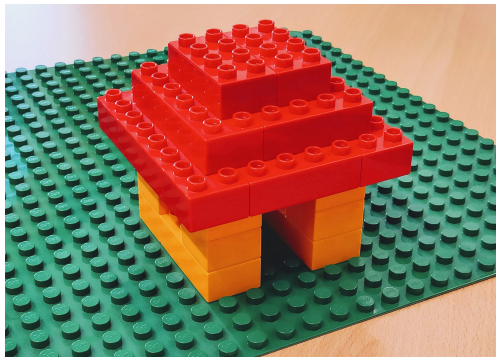
9.1.1 LEGO-Haus

Das LEGO-Haus (siehe Abbildung 9.1) hat eine Grundfläche von $128\,\mathrm{mm} \times 128\,\mathrm{mm}$ und eine Höhe von ungefähr $116\,\mathrm{mm}$. Es besteht aus einem unteren Teil, dem Erdgeschoss, mit einem Eingangs- und drei Fensterseiten sowie einem pyramidenförmigen Dach. Das Haus wird auf einer Basisplatte aus insgesamt 26 Legosteinen und in sechs aufeinanderliegenden Ebenen errichtet. Lediglich die zwei Typen an Legosteinen, 2×2 und 4×2, werden für den Hausbau benötigt. Für das Erdgeschoss werden neun 2×2 und drei 4×2 orangenfarbene Legoteine verwendet, das Dach besteht aus insgesamt vier 2×2 und zehn 4×2 roten Legosteinen. Das Haus kann sowohl mit einem als auch mit mehreren Robotern gefertigt werden. Unabhängig davon birgt es mehrere Deadlock-Situationen aufgrund der vier roten 2×2-Ecksteine des Dachs (Ecksteinproblem).

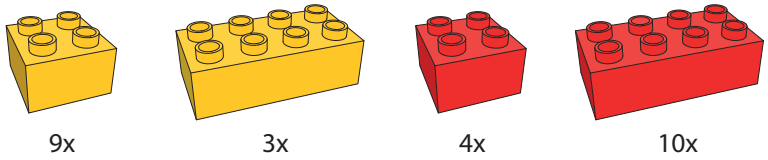
Für das Zusammensetzen der Legosteine gibt es viele unterschiedliche Möglichkeiten. Je nach Situation müssen die einzelnen Steine an unterschiedlichen Positionen gegriffen werden, damit beim Absetzen keine Kollision zwischen Greiferbacken und benachbarten



(a) Digitaler Zwilling



(b) Reales Produkt



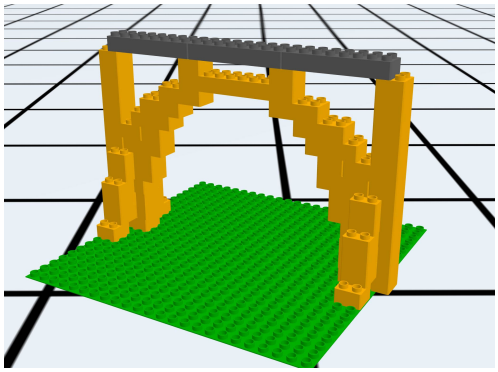
(c) Benötigte Steine für die Montage des Hauses aus LEGO

Abbildung 9.1. Das LEGO-Haus als Evaluation für die frühzeitige Deadlockerkennung.

Steinen entsteht. Daher werden für einen Stein oft mehrere Möglichkeiten angeboten, wie und an welcher Stelle dieser konkret gegriffen werden kann. Lässt man außer Betracht, ob gewisse Reihenfolgen der zu setzenden Steine tatsächlich möglich oder ungültig sind, so existieren bei der Fertigung mit einem Roboter, der jeden Stein auf 12 unterschiedliche Arten greifen kann, mehr als $210 * 10^{642}$ mögliche Abläufe, die den Suchraum einer ungefilterten Planung bestimmen. Dabei sind die Variationen darüber, wie jeder Stein über ein Magazinsystem bereitgestellt werden kann, noch außer Acht gelassen. Viele dieser Abläufe sind allerdings ungültig und führen in das Ecksteinproblem. Bei der Planung sind diese Deadlocks in der Situation, in der sie unvermeidlich entstehen, noch nicht erkennbar; erst im späteren Verlauf wird die Auswirkung sichtbar. Um den Planungsansatz $PaRTs$ auf seine Performanz mit derartigen, spät erkennbaren Deadlocks hin zu untersuchen, wird er mit den zahlreich auftretenden Ecksteinproblemen des LEGO-Hauses konfrontiert. Die Ausführung des resultierenden Roboterprogramms wird sowohl in der Simulation als auch in einer realen Roboterzelle ausgeführt.

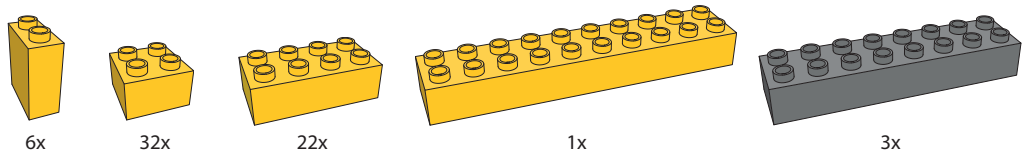
9.1.2 LEGO-Brücke

Die LEGO-Brücke (siehe Abbildung 9.2) wird auf einer Basisplatte errichtet und überspannt eine Distanz von etwa $290\,\mathrm{mm}$. Sie besteht aus insgesamt 64 in 14 Ebenen aufgebauten Legosteinen in fünf unterschiedlichen Größen und zwei Farben. Sechs doppelt hohe 1×2-Steine sind in den Kämpfern verbaut, 32 Legosteine im Format 2×2 und 22 im Format 4×2 bilden den Hauptbestandteil des Brückenbogens in der Farbe hell-orange. Kernelement des Bogens ist ein 10×2-Legostein, das als oberstes Element



(a) Digitaler Zwilling

(b) Reales Produkt



(c) Benötigte Steine für die Montage der Brücke aus LEGO

Abbildung 9.2. Die LEGO-Brücke als Evaluation für implizite Multi-Roboter-Kooperation.

die Verbindung der Brückenhälften darstellt und diese statisch stabilisiert. Drei graue 8×2-Legosteine dienen als Fahrbahn, die auf die Brückenpfeiler aufgesetzt ist.

Im Gegensatz zum LEGO-Haus ist die LEGO-Brücke nicht mit nur einem einzelnen Roboter herstellbar. Schon beim Aufbau einer Hälfte der Brücke würde diese ab einer gewissen Höhe beim Absetzen des nächsten Legosteins zusammenbrechen. Daher ist hier das Mitwirken von mindestens einem weiteren Roboter notwendig, der die Konstruktion beim Setzen weiterer Steine geeignet stützt. Beim Errichten der anderen Seite der Brücke sind ebenfalls mindestens zwei Roboter erforderlich. Gleichzeitig muss aber die bereits stehende Bogenhälfte weiterhin von einem Roboter gestützt werden. Zuletzt wird der Querstein, der die beiden Hälften erstmals miteinander vereint und diese stabilisiert, von einem Roboter gesetzt, während zwei andere Roboter die beiden Unterkonstruktionen stützen. Insgesamt sind daher drei Roboter für die Montage der Brücke erforderlich.

Die LEGO-Brücke ist ein geeignetes Beispiel, den Planungsansatz $PaRTs$ auf die Verwendung mehrerer Roboter zu testen. Besonders herausfordernd ist das koordinierte Zusammenspiel der Roboter, das für einen erfolgreichen Zusammenbau der Brücke erforderlich ist. Dabei wissen die unabhängigen Fähigkeiten nichts von der gegenseitigen Existenz. Stattdessen ist die automatische Planung gefordert, die nötige koordinierte Roboterkooperation zu erkennen und auszuarbeiten.

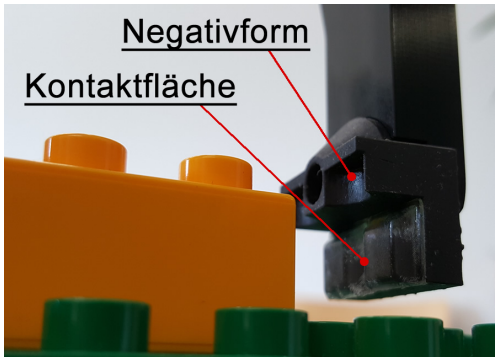
Bei der Fertigung mit mehreren Robotern besteht grundsätzlich die Gefahr der gegenseitigen Kollision. Die Planung mit $PaRTs$ stellt in jedem Planungsschritt automatisch sicher, dass die resultierende Planungssituation mit den darin beschriebenen Positionen der Einzelteile und Stellungen der Roboter keine Kollision beinhaltet. Im Weiteren gilt die Annahme, dass die von den Fähigkeitenmodulen erzeugten Bewegungen der Roboter

zu keinen Kollisionen in der Fertigungszelle führen. Über die nachgelagerte Optimierung des Planungsansatzes werden die zunächst sequentiellen Abläufe an Aktionsschritte in einen möglichst parallelisierten Ablauf überführt. Jede Aktion, die zuvor exklusiv in einer definierten und statischen Umgebung stattgefunden hat, sieht sich nun einer dynamisch verändernden Umwelt ausgesetzt, in der sich andere Roboter möglicherweise zeitgleich bewegen und indes das Produkt der Montage beeinflussen. Das Auftreten von Kollisionen wächst hierbei zu einem völlig neuen Gefahrenpotential. Grundsätzlich existieren Ansätze, die sich mit Strategien zur Kollisionsvermeidung bei parallelen Roboterbewegungen beschäftigen. Diese sind nicht Bestandteil dieser Arbeit; der Einbau solcher Ansätze in den Planungsansatz $PaRTs$ ist aktueller Forschungsgegenstand. Kollisionsbehaftete Montageprogramme sind zwar nicht ohne Beschädigungen an Robotern oder an Teilen der Roboterzelle real ausführbar, bei einer simulativen Ausführung richten Kollisionen allerdings keinen Schaden an. Aus diesem Grund werden sämtliche Planungsergebnisse für die Montage der LEGO-Brücke mit den drei parallel arbeitenden Robotern über die Simulation ausgeführt und ausgewertet.

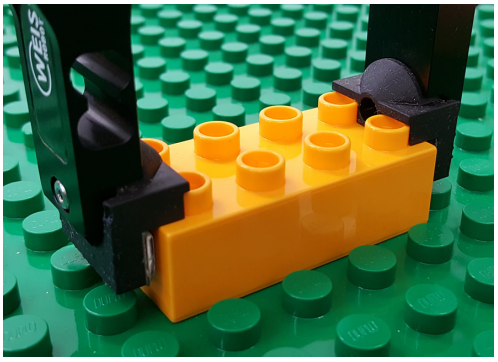
9.2 Aufbau der Roboterzelle

Die Qualität und der Erfolg der Fertigung von Haus und Brücke mit LEGO wird überwiegend dadurch bestimmt, wie gut die Roboterzelle mit Legosteinen umgehen kann. Auch wenn die Fertigungsplanung z. B. mit einem üblichen Parallelgreifer erfolgreich wäre, so kann es dennoch bei einer realen Ausführung des Programms zu Ungenauigkeiten kommen. Besonders beim Greifen eines Legosteins muss dieser mit etwas Kraft von der darunterliegenden Platte oder einem anderen Stein abgezogen werden, wobei der Legostein dabei möglicherweise im Greifer verrutscht. Beim anschließenden Setzen des Steins trifft dieser womöglich nicht exakt auf das Rastermaß und im schlechtesten Fall kommt es zu einer Beschädigung der Legoteile.

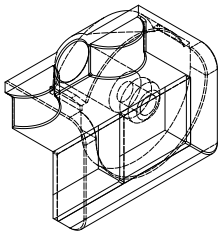
Derartige Ungenauigkeiten gilt es nach Möglichkeit zu reduzieren. Für die Evaluation mit LEGO wurden daher spezialisierte Greifbacken (siehe Abbildung 9.3) entwickelt, die an die Finger eines Parallelgreifers montiert werden können und eine präzise und zuverlässige Aufnahme von Duplosteinen ermöglichen. Das Design der Greiferbacke wurde so gewählt, dass es der Negativform eines Steins entspricht und somit ein formschlüssiges Greifen erlaubt (siehe Abbildung 9.3b). Da ein Roboter in der Realität aufgrund physikalischer Eigenschaften immer eine gewisse Ungenauigkeit beim Anfahren von Punkten aufweist, stehen die Greiferbacken zu Beginn eines Greifvorgangs um ein kleines Offset im Millimeterbereich versetzt zum Stein. Passgenaue Aussparungen der Greiferbacken dienen dazu, während eines Greifvorgangs (Schließen des Greifers) die Pins des Steins auszurichten und die Position des Steins im Greifer eindeutig zu definieren. Das in Abbildung 9.3c angegebene Modell der Greiferbacke wurde über eine additive Fertigung (3D-Druck) in mehrfacher Stückzahl hergestellt. Um zu garantieren, dass sich ein gegriffener Stein beim Hochheben (Abziehen) vom Untergrund löst und nicht der Greifer vom Stein abgleitet, dienen zugeschnittene Gummi-Türstopper als Kontaktfläche zum Stein und bewirken eine hohe Rutschfestigkeit (siehe Abbildung 9.3a). Mit dem gewählten Design sind alle Aktionen möglich, die für die Montage von Produkten aus

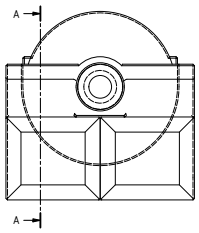


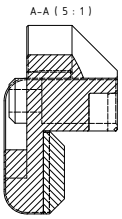
(a) Design und Aufbau der Greiferbacke

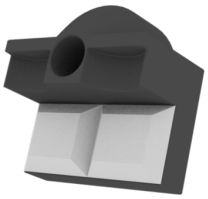


(b) Passgenaues Greifen eines Duplosteins









(c) Technische Zeichnung der Greiferbacke **(d)** Virtuelles Modell

Abbildung 9.3. Zum passgenauen und prozesssicheren Handling von Duplosteinen wurden spezialisierte Greiferbacken entwickelt, mithilfe derer ein Stein präzise eingepasst werden kann.

LEGO notwendig sind: ein prozesssicheres Greifen und Platzieren von Steinen sowie das Stützen von Konstruktionen mit der ebenen Seitenfläche der Greiferbacken.

9.2.1 Objekte und technische Ressourcen

Der Planungsansatz erfordert in der Phase der Fertigungsplanung (ab Schritt (5), Zuweisung, siehe Abbildung 4.1 in Abschnitt 4.1) die Angabe einer Roboterzellen-Definition. Diese muss neben den zur Verfügung stehenden technischen Ressourcen auch deren räumliche Anordnung sowie die initiale Konfiguration der technischen Ressourcen zu Beginn der Fertigung beschreiben. Für die Evaluation mit LEGO setzt sich die Roboterzelle aus insgesamt sechs technischen Ressourcen und einer LEGO-Basisplatte, auf der das zu fertigende Produkt errichtet wird, zusammen. Das virtuelle Modell der Roboterzelle ist in Abbildung 9.4 abgebildet.

Um die Basisplatte im Zentrum der Roboterzelle herum gruppieren sich drei Roboter der Firma KUKA: ein LBR iiwa und zwei LWR 4. Jeder der Roboter verfügt über einen Parallelgreifer des Typs SCHUNK WSG 50, der mit einem entsprechenden Adapter am Flansch des Roboters montiert ist. Die Finger der Greifer sind mit den spezialisierten LEGO-Greiferbacken ausgestattet, sodass jeder Roboter sowohl für eine Greifaktion als auch für das Stützen von Konstruktionen geeignet ist. Zu den Robotern existieren insgesamt drei kleinere LEGO-Platten, die als Bereitsteller von Legosteinen fungieren. Auf jeder dieser Platten sind mehrere Bereiche definiert, die für je einen Legostein-Typ

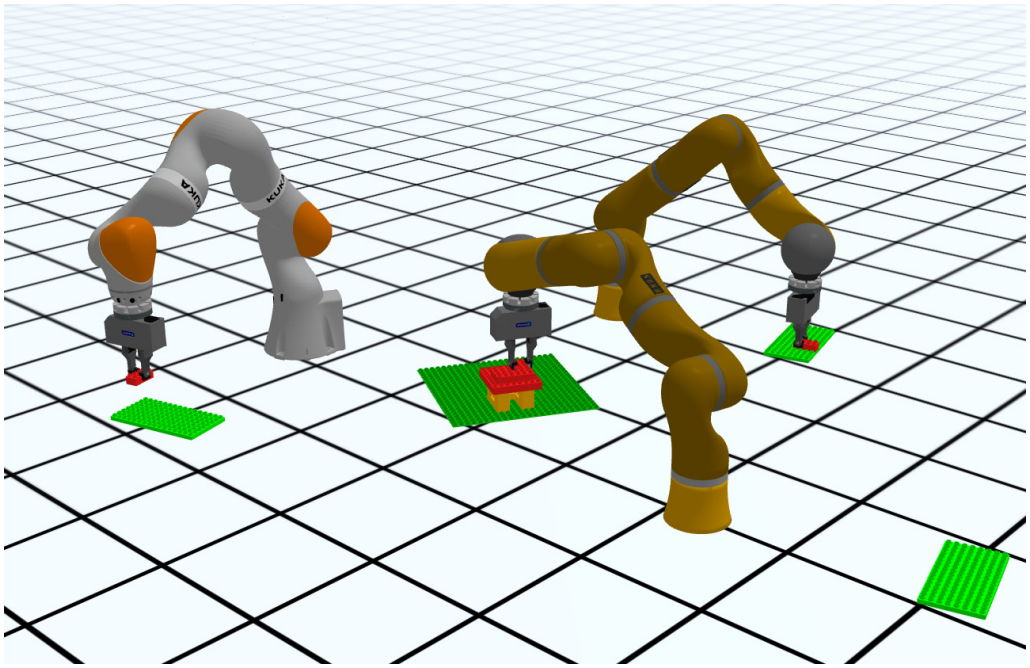


Abbildung 9.4. Virtuelles Modell der Roboterzelle mit drei Robotern und ihren Greifern, einer Basisplatte und drei weiteren Platten zum Bereitstellen von Legosteinen.

bestimmter Größe und Farbe reserviert sind. Auf diese Weise kann ein Bereitsteller gleichzeitig mehrere unterschiedliche Legosteine zur Verfügung stellen In der Roboterzellen-Definition ist als initiale Konfiguration für die Roboter deren Home-Position angegeben, die Greifer sind initial geöffnet.

Neben den Elementen der Roboterzellen-Definition sind auch sämtliche Legosteine als Einzelteile der Produkte Bestandteil der Planung und somit als Objekte in der virtuellen Fertigung verfügbar. Das LEGO-Haus wird in der Simulation mit drei, in der realen Ausführung mit einem Roboter gefertigt. Für die reale Ausführung steht ein Steuerungsrechner (Robot Control Core) bereit, der per Ethernet [61] mit dem LBR iiwa kommuniziert und für das Ansprechen des Greifers eine CAN-Verbindung nutzt.

9.2.2 Einmessen der realen Roboterzelle

Damit die Ausführung eines geplanten Programmablaufs zur Fertigung der LEGO-Strukturen auch real mit einem physischen Roboter funktioniert, ist eine feine Abstimmung der echten Roboterzelle mit dem für die Planung verwendeten Modell der Roboterzelle erforderlich. Die entsprechenden Positionsdaten von Roboter, Basisplatte und Bereitsteller sind dabei in der Vermessungs-Datenbasis der Roboterzellen-Definition hinterlegt und können dort gemäß den tatsächlichen Verhältnissen aktualisiert werden.

Das Vorgehen bei der Vermessung der Roboterzelle gliedert sich in zwei Schritte: Über die Werkzeugvermessung wird die Geometrie des Mess-Werkzeugs bestimmt, um damit beim anschließenden Teachen von Koordinatensystemen die relative Position von

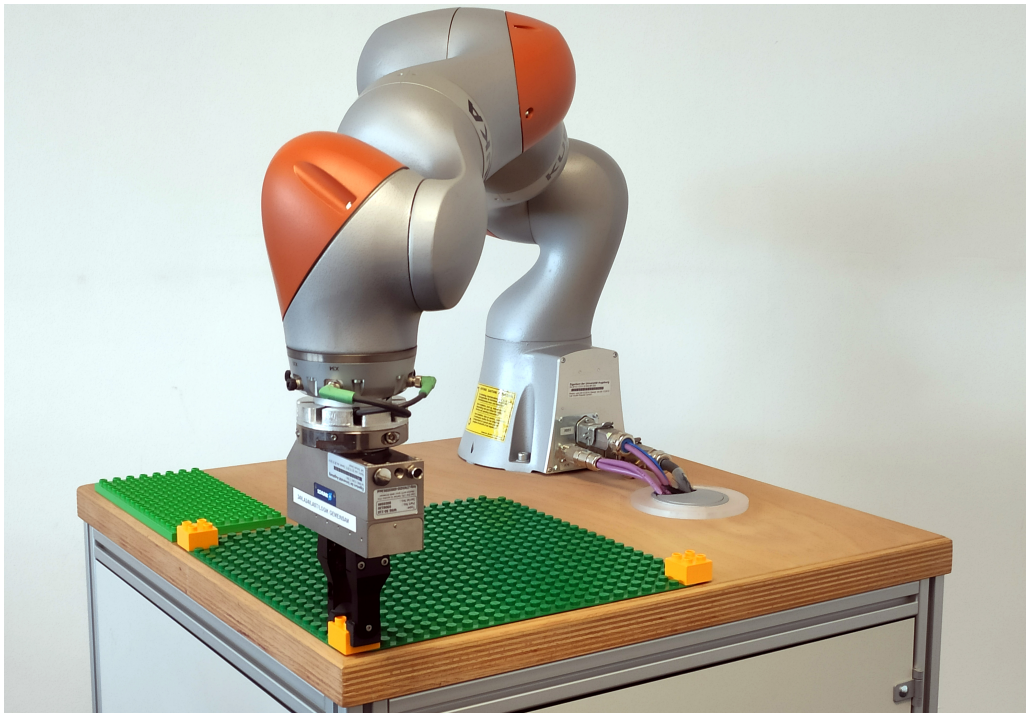


Abbildung 9.5. Die Basisplatte wird analog zur 3-Punkt-Methode über drei an den Ecken aufgesteckte Legosteine vermessen.

Roboter und zu vermessendem Objekt festlegen zu können. Als Werkzeug zum Teachen der Koordinatensysteme soll der Greifer verwendet werden, der später auch für die Produktion eingesetzt wird. Mit diesem können zwar keine Messpunkte wie mit einer Messspitze angefahren werden, doch ist ein definiertes Anfahren von Legosteinen möglich, die hierbei als eine Art Messpunkt dienen können.

Anstelle der Werkzeugvermessung kann im Fall von LEGO auf eine rechnerische Bestimmung der Greifer-Geometrie zurückgegriffen werden. Der Greifer besitzt zwar natürlich bedingt eine gewisse Toleranz, die ohne Anwendung der Werkzeugvermessung unberücksichtigt bleibt. Doch ist diese Ungenauigkeit deutlich geringer als das erlaubte Spiel, das Legosteine in der Größe von LEGO® DUPLO® beim Aufeinanderstecken erlauben. Daher kann die Toleranz der Greifergeometrie in diesem Fall vernachlässigt werden. Um das Werkzeug eindeutig festzulegen, ist die Drehverschiebung zwischen dem Flansch des Roboters und dem Punkt des Greifers zu bestimmen, an dem sich ein Legostein befindet, wenn dieser gegriffen wurde. Die Z-Komponente der Drehverschiebung setzt sich aus der Höhe der Flanschadapterplatte, der Geometrie des Greifers sowie der Geometrie der spezialisierten Greiferbacken zusammen. Die Werte hierfür lassen sich aus den Datenblättern von Flanschadapter und Greifer bzw. aus dem Modell der Greiferbacke ablesen und sind damit bekannt. Die X- und Y-Komponenten der Drehverschiebung sind null – die Greifposition ist damit unabhängig von der Greiferöffnung. Ein gegriffener Legostein befindet sich also mittig unterhalb des Flansches mit einem definierten Z-Abstand.

Über das anschließende Teachen von Koordinatensystemen werden die relativen Positionen der Objekte in der Roboterzelle festgelegt. Dabei kann dem Vorgehen nach der 3-Punkt-Methode (siehe Abschnitt 8.2.1) gefolgt werden, auch wenn in der Roboterzelle für die Fertigung von LEGO-Strukturen keine Messpunkte existieren. Anstelle der Messpunkte werden drei 2×2-Legosteine verwendet, die an den Ecken der Basisplatte aufgesteckt sind (siehe Abbildung 9.5). Statt einem präzisen Anfahren eines Messpunkts mit der Messpitze wird hier der Greifer so über einem Legostein ausgerichtet, dass er die perfekte Position einnimmt, um den Stein zu greifen. Auf diese Weise wird analog zur 3-Punkt-Methode der erste Legostein angefahren, um die Basis des Koordinatensystems festzulegen. Mit Anfahren des zweiten Legosteins wird die X-Achse festgelegt, der dritte Legostein bestimmt zuletzt noch die Richtung der Y-Achse. Mit den hierbei erfassten Daten kann die Position der Basisplatte relativ zur Roboterbasis eindeutig berechnet werden. Dieses Vorgehen wird für die Legoplatte des Bereitstellers wiederholt, um auch deren Position zu bestimmen. Die erfassten Positionsdaten werden in die Vermessungs-Datenbasis der Roboterzellen-Definition eingetragen. Die Planung erfolgt nun auf Grundlage des realen Roboterzellenlayouts.

9.3 Erweiterungen zum modularen Ansatz

Um die Fertigung eines aus LEGO bestehenden Produkts mit dem modularen Planungsansatz $PaRTs$ zu planen, sind einige überwiegend domänenspezifische Erweiterungen von den entsprechenden Experten zur Verfügung zu stellen. Abschnitt 9.3.1 geht darauf ein, wie das zu fertigende Produkt mithilfe der einheitlichen Produktbeschreibung modelliert ist. Die für die Planung mit LEGO erforderlichen Attribut-Typen sowie Analysemodule, die für die Extraktion spezieller Attribute aus der einheitlichen Produktbeschreibung benötigt werden, erläutern Abschnitte 9.3.2 und 9.3.3. Abschnitt 9.3.4 erklärt die Verwendung des Domänentaskmoduls. Die folgenden Abschnitte 9.3.5 bis 9.3.7 gehen auf die hinzugefügten Domänenprüf-, Fähigkeiten- und Validatormodule ein, die zum einen mögliche Aktionsschritte definieren und zum anderen erreichte Zwischenschritte überprüfen und ungeeignete Pläne gegebenenfalls verwerfen.

9.3.1 Modellierung über die einheitliche Produktbeschreibung

Von den beiden LEGO-Strukturen, die in Fallstudie 1 geplant und gebaut werden, sind sämtliche dazu benötigten Einzelteile in Form der einheitlichen Produktbeschreibung zu hinterlegen. Insgesamt sind dies fünf unterschiedliche Formate an Legosteinen mit den Rastermaßen 1×2 (Stein mit doppelter Höhe), 2×2, 4×2, 8×2 und 10×2. Die Basisplatte sowie die Platten der Bereitsteller als Teile der Roboterzelle sind nicht Bestandteil des Produkts, sondern technische Ressourcen oder Hilfsmittel. Die Modellierung über die einheitlichen Produktbeschreibung ist für diese Elemente daher nicht erforderlich.

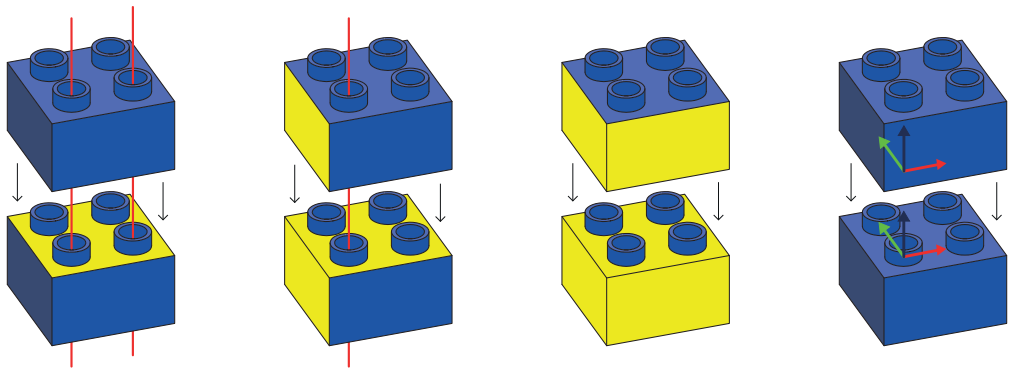
Listing 9.1 zeigt exemplarisch die Einzelteilbeschreibung für einen 2×2-Legostein. Damit der Legostein-Typ in unterschiedlichen Farben verwendet werden kann, ist die abgebildete Einzelteilbeschreibung mit einem gewünschten Farbwert parametrierbar (Zeile 2). Die Farbe spielt bei der Auswahl des Visualisierungsmodells für den Legostein eine Rolle und wird in Zeile 3 für die Angabe der Modelldatei verwendet. Für die spätere

```
1  <!-- Beschreibung eines 2x2-Legosteins -->
2  <Einzelteilbeschreibung id="stein_2_2" params="farbe">
3    <Model src="legostein_2x2_{$farbe}.stp" />
4
5    <!-- Features für Pin 1 -->
6    <Feature id="ach_1_1" typ="achse" aufpunkt="-0.008,-0.008,0"
         richtungsvektor="0,0,1" />
7    <Feature id="pin_1_1" typ="frame" aufpunkt="-0.008,-0.008,0.0191"
         orientierung="0,0,0" />
8    <Feature id="buc_1_1" typ="frame" aufpunkt="-0.008,-0.008,0"
         orientierung="0,0,0" />
9
10   <!-- Features für Pin 2-4 -->
11   ...
12
13   <!-- Features für Seitenflächen -->
14   <Feature id="vorne" typ="ebene" aufpunkt="0.016,0,0" normalenvektor="1,0,0" />
15   <Feature id="links" typ="ebene" aufpunkt="0,0.016,0" normalenvektor="0,1,0" />
16   ...
17 </Einzelteilbeschreibung>
```

Listing 9.1. Einzelteilbeschreibung im XML-Format für einen Legostein mit Rastermaß 2×2.

Verwendung des Bausteins in Baugruppen und Produkten definiert er eine Reihe an Features, über die er mit räumlichen Abhängigkeiten bezüglich anderer Legosteine versehen werden kann. Für jeden Pin wird eine zentrische Achse definiert, die längs durch den Pin hindurchgeht (Zeile 6). Zeilen 7 und 8 definieren zum Pin ein Koordinatensystem auf der Oberseite des Legosteins sowie ein Koordinatensystem auf der Unterseite, an der sich das entsprechende Gegenstück zum Aufstecken eines Pins befindet. Zudem ist für jede Seite des Legosteins ein Ebenen-Feature definiert (Zeile 14ff). Die angegebene Einzelteilbeschreibung für 2×2 existiert analog für alle Formate an Legosteinen.

Die über Einzelteilbeschreibungen definierten Legosteine können in Baugruppenbeschreibungen zusammengesetzt werden. Über Constraints wird dabei ihre geometrische Beziehung zueinander definiert (siehe Abschnitt 5.1.1). Für zwei aufeinander gesteckte Legosteine gibt es mehrere Möglichkeiten, wie diese über ihre gegebenen Features in diese Beziehung gebracht werden können. Drei mögliche Definitionen über Constraints sind in Abbildung 9.6 abgebildet. In Abbildung 9.6a ist die relative Position zweier Legosteine über zwei Achsenkoinzidenzen und eine Ebenenkoinzidenz festgelegt. Hierbei bestimmen die Achsenkoinzidenzen die Ausrichtung der Pins übereinander. Mit der zusätzlichen Ebenen-Koinzidenz von Ober- bzw. Unterseite der Legosteine wird der Kontakt erzwungen. Über nur eine Achsenkoinzidenz stellt Abbildung 9.6b eine gemeinsame Ausrichtung nach oben her und richtet einen Pin beider Legosteine richtig aus. Auch hier wird über eine Ebenen-Koinzidenz der Kontakt erzwungen; der verbleibende Freiheitsgrad in Form einer Drehung um die Achse wird mit einer zweiten Ebenen-Koinzidenz festgelegt. Dieselbe geometrische Beziehung stellt Abbildung 9.6c mit der Verwendung dreier Ebenen-Koinzidenzen her. Abbildung 9.6d nutzt jeweils einen der Pin-Frames auf der Ober- bzw. Unterseite der Legosteine, um diese mit einer Koinzidenz eindeutig zueinander auszurichten. In allen vier Fällen ist die geometrische Beziehung eindeutig definiert. Es gibt weitere Möglichkeiten, dieselbe Beziehung auf andere Art



(a) Zwei Achsen- und eine Ebenen-Koinzidenz

(b) Eine Achsen- und zwei Ebenen-Koinzidenzen

(c) Drei Ebenen-Koinzidenzen

(d) Eine Frame-Koinzidenz

Abbildung 9.6. Beispielhafte Anwendung von Achsen-, Ebenen- und Frame-Koinzidenzen als Constraints zwischen Einzelteilen zur relativen Positionierung zweier Legosteine. Zur Verdeutlichung sind die Legosteine in der Abbildung nicht zusammengesteckt sondern in Form einer Explosionsdarstellung abgebildet.

und Weise auszudrücken, z. B. unter Verwendung von Parallelitäts-Constraints und Offsets. Die in Abbildung 9.6d dargestellte Variante über eine Frame-Koinzidenz ist eine Sonderform des Frame-Offsets mit Translation um null und Rotation um null. Um zwei Legosteine z. B. mit einer Drehung um 90° aufeinander zu stecken, kann bei einem Offset zwischen zwei Frames der Winkel als Drehung um die Z-Achse angegeben werden. Die Verwendung des Offsets zwischen Frames als Constraint ist sowohl für die Modellierung als auch für die automatische Constraintauflösung mit nur minimalem Aufwand verbunden und wird daher vorrangig bei der Erstellung der LEGO-Modelle im Format der einheitlichen Produktbeschreibung eingesetzt.

Das LEGO-Haus ist mit einer Baugruppenbeschreibung definiert, die insgesamt 26 parametrisierte Verwendungen von Einzelteilbeschreibungen im benötigten Rasterformat und in entsprechender Farbe besitzt. Sie enthält zudem 25 Constraints in Form von Koinzidenzen und Offsets zwischen Frames und definiert damit für die 26 Einzelteile eine eindeutige räumliche Beziehung. Für die LEGO-Brücke wurde eine etwas tiefere hierarchische Verschachtelung der Baugruppen gewählt. Da die Brücke symmetrisch aufgebaut ist, kann sie logisch halbiert und von Ebene 1 bis 11 als Halbbogen in einer eigenen Baugruppenbeschreibung modelliert werden. Die darin vorkommenden 27 Legosteine werden über entsprechende Einzelteilbeschreibungen referenziert und mit 26 Constraints geometrisch festgelegt. Listing 9.2 zeigt die Definition einer Baugruppenbeschreibung für die Bogenhälfte der Brücke. Die Bogenhälfte ist mit einer gewünschten Farbe parametrierbar, die an die Unterelemente, also die einzelnen Legosteine, weitergegeben wird. Von besonderer Bedeutung ist der oberste Legostein der Bogenhälfte (Zeile 4), der im fertigen Produkt unterhalb des Verbindungssteins sitzt und bei der Definition der Baugruppenbeschreibung für die Brücke eine spezielle Rolle spielt.

```
1  <Baugruppenbeschreibung id="bogenhaelfte" params="farbe">
2    <!-- Referenzierung von Einzelteilen und Constraints -->
3    ...
4    <!-- Oberster Stein direkt unterhalb des Verbindungssteins -->
5    <Unterelement id="stein_oben" ref="stein_4_2" params="farbe:{$farbe}" />
6    ...
7  </Baugruppenbeschreibung>
8
9  <Baugruppenbeschreibung id="bruecke" params="farbe1,farbe2">
10   <!-- Zweifache Verwendung der Bogenhälfte -->
11   <Unterelement id="b_links" ref="bogenhaelfte" params="farbe:{$farbe1}" />
12   <Unterelement id="b_rechts" ref="bogenhaelfte" params="farbe:{$farbe1}" />
13
14   <!-- Verbindungsstein -->
15   <Unterelement id="verbindung" ref="stein_10_2" params="farbe:{$farbe1}" />
16
17   <!-- Verbindung der beiden Bogenhälften -->
18   <Constraint typ="koinzidenz" feature1="b_links.stein_oben.pin_3_1"
         feature2="verbindung.buc_1_1" />
19   <Constraint typ="offset" feature1="b_rechts.stein_oben.pin_3_1"
         feature2="verbindung.buc_10_2" offset="0,0,0,PI,0,0" />
20
21   <!-- Weitere Unterelemente und Constraints -->
22   ...
23 </Baugruppenbeschreibung>
```

Listing 9.2. Ausschnitte aus der Baugruppenbeschreibung im XML-Format für eine Bogenhälfte und für eine aus zwei dieser Hälften zusammengesetzte Brücke.

Die Baugruppendefinition für die Brücke – in Listing 9.2 ab Zeile 9 ausschnittsweise dargestellt – referenziert zwei dieser Bogenhälften und definiert die Positionen der verbleibenden 10 Legosteine im oberen Bereich der Brücke. Sie erlaubt die Angabe von zwei Farben, einmal für die Bogenkonstruktion der Brücke und einmal für die Fahrbahn. Für den linken sowie rechten Teil der Brücke referenziert die Baugruppenbeschreibung zweimal die vorher definierte Bogenhälfte. Ein Legostein mit dem Raster 10×2 wird als Verbindungsstück zwischen den Bogenhälften erstellt (Zeile 15). Verbindungsstein und linke Bogenhälfte werden über eine Koinzidenz zweier ihrer Frames zusammengefügt, und zwar sollen Buchse (1,1) des Verbindungssteins (feature2 in Zeile 18) und Pin (3,1) von „stein_oben“ der linken Bogenhälfte (feature1 in Zeile 18) deckungsgleich sein. Auch die rechte Bogenhälfte wird in ähnlicher Weise mit dem Verbindungsstein in Beziehung gebracht (Zeile 19); statt einer Koinzidenz ist allerdings ein Offset angegeben, da die zweite Bogenhälfte gespiegelt – in diesem Fall entspricht dies einer Drehung um 180° um die Hochachse – angefügt wird. Bei der Modellierung der LEGO-Brücke wird somit davon profitiert, dass eine einmal definierte Baugruppenbeschreibung mehrfach wiederverwendet werden kann.

9.3.2 Legospezifische Attribut-Typen

Die mittels der einheitlichen Produktbeschreibung erstellten Modelle für das LEGO-Haus und die LEGO-Brücke werden für die Planung automatisch in das Format der Produktsituation – die Ziel-Produktsituation – überführt. Dazu ist es nötig, die durch

Constraints angegebenen räumlichen Beziehungen in räumliche und semantische Attribute zu überführen. Dafür werden spezifische Attribut-Typen verwendet, die speziell für die Montage mit LEGO definiert sind und mit denen geometrische und semantische Eigenschaften von Legosteinen ausdrückbar sind. Folgende legospezifischen Attribute werden für die Planung in der Fallstudie 1 mit LEGO eingesetzt:

LegoSteckAttribut: Sitzt ein Legostein auf einem anderen Stein oder auf einer Platte, so drückt ein LegoSteckAttribut einerseits die semantische Information aus, dass beide Legoteile zusammengesteckt sind. Dies impliziert für den späteren Planungsverlauf einen entsprechenden Umgang mit diesem Attribut, wie es auf- bzw. abgebaut werden kann. Zum anderen enthält das LegoSteckAttribut die geometrische Information darüber, an welcher Stelle und mit welchem Versatz die Legoteile aufeinandergesteckt sind. Gemäß Abschnitt 4.1.2 handelt es sich beim LegoSteckAttribut um eine Beziehung.

LegoSupportAttribut: Ist eine Legokonstruktion instabil und eine Montage weiterer Legosteine deshalb nicht möglich, kann ein Roboter die Konstruktion mit seinen Greiferbacken geeignet unterstützen. Eine derartige Belegung (siehe Abschnitt 4.1.2) des Greifers mit dem Legostein wird über ein LegoSupport Attribut ausgedrückt. Sie enthält den eindeutigen geometrischen Zusammenhang zwischen beiden Objekten und legt Einschränkungen fest, wie ein Auf- bzw. Abbau dieser Beziehung durch den Roboter erfolgen kann.

LegoGreifAttribut: Wird ein Legostein von einem Roboter aufgenommen, also mit seinem Greifer gegriffen, so wird sowohl die geometrische als auch die semantische Information dieser Belegung durch ein LegoGreifAttribut in der Planungssituation ausgedrückt.

Alle drei Attribut-Typen für die Planung mit LEGO sind in Abbildung 9.7 veranschaulicht. Jeder bereits gesetzte Legostein der abgebildeten Treppe ist mit einem oder zwei anderen Legosteinen bzw. der Basisplatte über ein LegoSteckAttribut verbunden. Der Roboter, der den obersten Legostein der Treppe stützt, besitzt in der aktuellen Produktsituation ein LegoSupportAttribut mit diesem Stein. Dadurch ist die Konstruktion stabil genug, dass der andere Roboter den nächsten Stein, der über ein LegoGreifAttribut mit dem Greifer verbunden ist, entsprechend platzieren kann.

Bei der Planung mit LEGO werden drei weitere, nicht legospezifische Attributtypen eingesetzt (siehe Abschnitt 4.1.2): Während Positions-Attribute festlegen, an welchen Stellen sich die Roboter in der Roboterzelle befinden, beschreiben Konfigurations-Attribute die jeweilige Stellung der Roboter. Ein Fixpunkt-Attribut beschreibt die feste Position der Basisplatte in der Mitte der Roboterzelle.

Da ein LegoSteckAttribut zwei Einzelteile des Produkts referenziert, handelt es sich um ein Produktattribut (siehe Definition 4.1). Es kann somit – ebenfalls wie das Fixpunkt-Attribut der Basisplatte – Bestandteil von Produktsituationen sein und spielt daher für die Prozessplanung eine wichtige Rolle. Dagegen referenzieren LegoSupportAttribute, LegoGreifAttribute, Positions- und Konfigurations-Attribute mindestens auch eine technische Ressource der Roboterzelle. Sie stellen daher Aktuatorattribute dar, die erst bei der Fertigungsplanung zur Erstellung konkreter Roboteraktionen Anwendung finden.

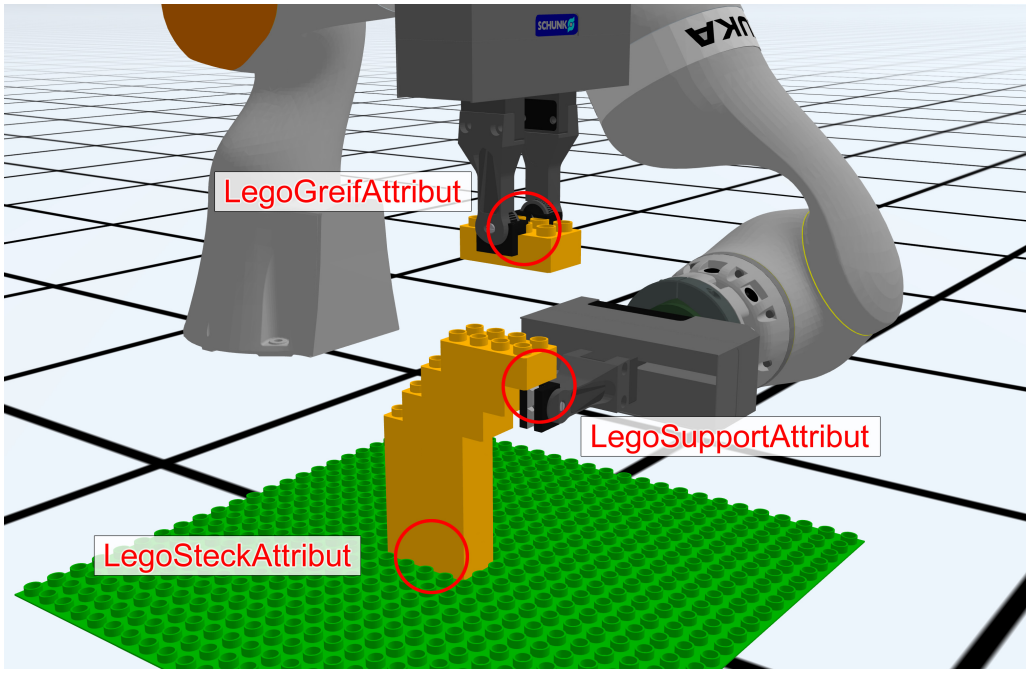


Abbildung 9.7. Beispiel einer Legomontage zur Veranschaulichung der drei legospezifischen Attribut-Typen: LegoSteckAttribut, LegoSupportAttribut und LegoGreifAttribut.

9.3.3 Bereitgestellte Analysemodule

Von den drei Attribut-Typen, die für die Planung der Montage mit LEGO verwendet werden, reicht allein das LegoSteckAttribut aus, um sämtliche Produkte aus LEGO über eine Produktsituation zu modellieren. Um in der ersten Phase der Planung – der Strukturanalyse – eine Produktsituation für das zu fertigende Produkt zu bestimmen, müssen anhand der vorliegenden Produktbeschreibung, die lediglich Einzelteile und räumliche Constraints enthält, eben genau solche LegoSteckAtribute bestimmt werden. Dies ist die Aufgabe von Analysemodulen (siehe Abschnitte 4.1.4 und 5.2.1). Für LEGO existiert insgesamt ein Analysemodul, das eine Produktbeschreibung untersucht und daraus LegoSteckAttribute ableitet. Ein vorgelagerter Constraintsolver löst dazu zunächst die geometrischen Constraints in entsprechende Drehverschiebungen auf und stellt mit diesen für jedes Einzelteil eine eindeutige Position zur Verfügung (siehe Abschnitt 5.2). Auf dieser räumlichen Struktur des Gesamtbauteils führt das Analysemodul geometrische Untersuchungen aus. Für alle Paare an Legosteinen wird anhand ihrer Position eine relative Drehverschiebung bestimmt und auf folgende Kriterien untersucht:

1. Sind die Legosteine zueinander weder um die X- noch um die Y-Achse rotiert?
   → Es existiert keine seitliche Kippung.
2. Ist die Rotation um die Hochachse (Z) ein Vielfaches von 90°?
   → Die Drehung ist kompatibel zum Raster von LEGO.
3. Entspricht die Verschiebung um Z der Höhe eines Steins?
   → Die Unterseite des einen Steins beginnt auf Höhe der Oberseite des anderen.

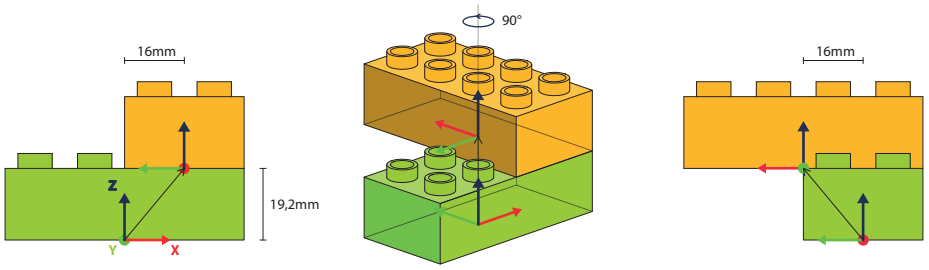


Abbildung 9.8. Betrachtung zweier verbundener Legosteine aus drei Perspektiven. Die fünf Kriterien, anhand derer das Analysemodul ein LegoSteckAttribut identifiziert, sind erfüllt.

4. Ist die Verschiebung um X und Y ein vielfaches des Rastermaßes von $16\,\mathrm{mm}$?
   → Die Legosteine halten das Rastermaß ein.
5. Überschneiden sich die Konturen der Legosteine aus X-Y-Sicht?
   → Die Steine überlagern sich bei Draufsicht von oben in mindestens einem Pin.

Sind alle fünf Kriterien für ein Paar an Legosteinen erfüllt, so beschreibt deren relative Drehverschiebung eindeutig, dass die beiden Legosteine aufeinander gesteckt sind. Abbildung 9.8 zeigt ein Beispiel mit zwei $4{\times}2$-Legosteinen, die eine relative Drehverschiebung mit dem Translationsanteil $[X\colon 0.016,\ Y\colon 0.016,\ Z\colon 0.0192]$ und dem Rotationsanteil $[A\colon \frac{\pi}{2},\ B\colon 0,\ C\colon 0]$ besitzen. Als Basis der Legosteine wird jeweils der Mittelpunkt auf deren Unterseite verwendet. Die fünf oben genannten Kriterien sind hierbei allesamt erfüllt. Das Analysemodul erstellt in diesem Fall ein LegoSteckAttribut für die beiden Legosteine unter Angabe ihrer relativen Drehverschiebung und fügt dies der Ziel-Produktsituation hinzu.

9.3.4 Bereitgestelltes Domänentaskmodul

Im Planungsansatz $PaRTs$ übernimmt das Domänentaskmodul die Aufgabe, ausgehend von einer initialen Produktsituation mögliche Abfolgen von Domänentasks zu identifizieren, die in eine gegebene Ziel-Produktsituation überführen (siehe Abschnitte 4.1.4 und 6.1.1). Bei LEGO entspricht ein Domänentask dem Platzieren eines Legosteins und dem Aufbauen der jeweiligen LegoSteckAttribute – ohne sich dabei auf konkrete Aktionen technischer Ressourcen zu beziehen. Er repräsentiert all das, was dafür getan werden muss: Bereitstellen des Legosteins, ggf. einen Support herstellen, den Stein greifen und an der Zielposition absetzen. Domänentasks stellen sozusagen die abstrakten Teilschritte dar, die für die Montage erforderlich sind.

Der Planungsansatz bietet für die Verwendung eines Domänentaskmoduls zwei Möglichkeiten, die in Abschnitt 6.1.1 vorgestellt sind: Ein Domänenexperte kann ein eigens für die Domäne LEGO entwickeltes Domänentaskmodul zur Verfügung stellen, das nach eigener Logik die nächstmöglichen Schritte für die Prozessplanung berechnet. Alternativ existiert eine generische Variante des Domänentaskmoduls, das Domänentasks anhand der Attribute aus aktueller Produktsituation und Ziel-Produktsituation generiert.

Für die Evaluationen der Fallstudie 1 wird das generische Domänentaskmodul eingesetzt. Da hierbei nur derjenige Legostein in einem Domänentask angeboten wird, der ein

LegoSteckAttribut zu einem bereits montierten Legostein oder zur Basisplatte aufbaut, sind die identifizierten Abläufe grundsätzlich richtig. Allerdings hat das generische Domänentaskmodul keine Kenntnis über die Problemstellungen von LEGO. Somit werden auch Legosteine zum Platzieren angeboten, von deren darunter befindlichen zwei Legosteinen erst einer montiert ist. Derartige Fehler können jedoch über geeignete Domänenprüfmodule erkannt und verhindert werden.

9.3.5 Bereitgestellte Domänenprüfmodule

Im Zuge der Planung werden verschiedene Situationen als mögliche Zwischenergebnisse der Montage angeboten. Um zu prüfen, ob eine solche Situation in der jeweiligen Domäne – im aktuellen Fall also LEGO – gültig ist, werden Domänenprüfmodule zur Bewertung der Gültigkeit eingesetzt (siehe Abschnitte 4.1.4 und 6.2.1). Die Prüfung kann in jedem Schritt der Prozessplanung erfolgen, indem das Domänenprüfmodul auf die zu erreichende Produktsituation des Domänentasks angewandt wird. Auch bei der Fertigungsplanung kann jeder Zwischenschritt überprüft werden, indem die Planungssituation – die neben dem Bauteilzustand auch den Zustand der Roboterzelle enthält – auf eine Produktsituation reduziert wird, also $p = s\restriction Produkt$ für eine Planungssituation $s \in \mathcal{S}$ und $p \in \mathcal{S}\restriction Produkt$ als entsprechende Produktsituation.

Insgesamt können mehrere Domänenprüfmodule bei der Planung eingesetzt werden. Für die Planung mit LEGO ist jedoch ein Domänenprüfmodul ausreichend. Die in Abschnitt 3.1.3 aufgezeigten Problemstellungen „Kollision“ und „Statik“ sind abhängig von den technische Ressourcen der Roboterzelle und daher nicht Gegenstand eines Domänenprüfmoduls. Auch „Überhänge“ sind nicht für Domänenprüfmodule relevant, da diese durch das Domänentaskmodul bestimmt werden. Der „Ausschluss“ eines Legosteins oder einer Teilkonstruktion bleibt allerdings noch als zu adressierendes Problem bestehen. Das legospezifische Domänenprüfmodul erfüllt somit die Aufgabe, einen Ausschluss in einer vorliegenden Situation zu erkennen, und stellt sicher, dass die Planung in einem solchen Fall nicht fortgesetzt wird. Grundsätzlich kann ein Ausschluss nur dann existieren, wenn in der Ziel-Produktsituation durch LegoSteckAttribute ein Zyklus über Einzelteile beschrieben ist. Zur Überprüfung werden zunächst alle Legostseine der Ziel-Produktsituation identifiziert, die in der aktuellen Situation noch nicht an der richtigen Stelle positioniert sind. Da in der Fallstudie 1 mit LEGO das Produkt auf einer Basisplatte zusammengesetzt wird, befindet sich ein Legostein genau dann an der richtigen Stelle, wenn mindestens eines seiner LegoSteckAttribute der Ziel-Produktsituation $p_{goal} \in \mathcal{S}\restriction Produkt$ ebenso in der aktuellen Situation $s \in \mathcal{S}$ existiert. Die Menge E der noch nicht richtig positionierten Einzelteile (Legosteine), die demnach kein Attribut in s und p_{goal} gemeinsam haben, ist in Gleichung 9.1 definiert:

$$E = \{e \in \mathcal{E} \mid attr(e, s\restriction Produkt) \cap p_{goal} = \emptyset\} \tag{9.1}$$

Angenommen, es existieren nun zwei fertig montierte Legosteine $e_1, e_2 \in \mathcal{E} \setminus E$, zu denen ein noch nicht fertig montierter Legostein $e_{nf} \in E$ jeweils ein LegoSteckAttribut a_1 und $a_2 \in \mathcal{A}\restriction Produkt$ besitzt. Erinnern wir uns an die vorher einmal erwähnte Diamantstruktur aus LEGO, in der einer der beiden mittleren Legosteine bei der Montage

„vergessen“ wurde (siehe Abbildung 6.5). Oberer und unterer Legostein der C-förmigen Konstruktion entsprechen e_1 und e_2, der noch nicht montierte Legostein in der Mitte entspreche e_{nf}. Betrachtet man die Prozesse $p_1, p_2 \in \mathcal{P}$, die zum Aufbau von a_1 und a_2 notwendig wären, so liegt dann ein Ausschluss vor, wenn beide Prozesse nicht kompatibel sind, i. Z. $\neg komp(p_1, p_2)$. D. h. es gibt keine zwei ausführungsäquivalente Tasks $t_1, t_2 \in \mathcal{T}$, die diese Prozesse in der gegebenen Situation $s \in \mathcal{S}$ simultan umsetzen können, also $t_1 \not\sim_s t_2$ (siehe Definition 6.2).

Anhand dieser Kriterien identifiziert das legospezifische Domänenprüfmodul Einzelteile, die in der gegebenen Situation von einem Ausschluss betroffen sind. In analoger Weise wird auch für fehlende Teil-Produkte, also Baugruppen mehrerer noch nicht montierter Einzelteile, mithilfe von Techniken zur Zyklenerkennung überprüft, ob für sie ein Ausschluss vorliegt. Somit verhindert das Domänentaskmodul, dass der aktuelle Planungsweg weiter betrachtet wird, wenn das Setzen eines Legosteins einen Ausschluss für einen noch nicht montierten Stein verursacht.

9.3.6 Bereitgestellte Fähigkeitenmodule

Ein mit Robotern ausführbares Programm liegt dann vor, wenn jeder Domänentask des Programmablaufs vollständig durch entsprechende technische Tasks beschrieben ist. Nur der technischer Task beschreibt eine konkrete Aktion eines Roboters. Im Rahmen der Fertigungsplanung werden technische Tasks, die in einer gegebenen Situation ausführbar wären, über Fähigkeitenmodule generiert (siehe Abschnitte 4.1.4 und 7.1.3). Einem Roboter werden ein oder mehrere Fähigkeitenmodule zugewiesen, die für diesen alle möglichen Aktionen erzeugen können. Bewirkt eine Aktion, dass sich der betreffende Roboter bewegt, so gibt es ein Aktuatorattribut $a_{konf} \in \mathcal{A}$, das die Konfiguration, also die Stellung des Roboters, vor Ausführung der Aktion beschreibt. Die Konfiguration des Roboters nach der Ausführung wird durch das Aktuatorattribut $a'_{konf} \in \mathcal{A}$ beschrieben. Für die Fallstudie mit LEGO existieren vier Fähigkeitenmodule, die in jedem Schritt der Planung alle möglichen technischen Aktionen bereitstellen.

Fähigkeitenmodul „Bereitstellen“: Um einen Legostein für die Montage in der Roboterzelle verfügbar zu machen, kann ein Bereitsteller einen noch nicht verfügbaren Legostein an einem definierten Platz auf seiner Platte anbieten. Voraussetzung ist, dass dieser Platz in der gegebenen Situation noch frei ist. Das Fähigkeitenmodul bietet für seinen Bereitsteller in einem Schritt der Planung üblicherweise eine Vielzahl an Bereitstell-Aktionen an – eine für jeden noch nicht verfügbaren Legostein. Jede Aktion bewirkt ein Aufbauen eines Legosteckattributs $a_{steck} \in \mathcal{A}$ zwischen der Bereitsteller-Platte und dem Legostein. Da sich ein Bereitsteller dabei nicht bewegt wie etwa ein Roboter, ist der Effekt und der dadurch beschriebene Situationsübergang einer derartigen Aktion vollständig durch folgende Gleichung 9.2 beschrieben:

$$\begin{aligned} \mathit{Effekt}(t, s) &= (\{a_{steck}\}, \{\}) \\ nach(t, s) &= s \cup \{a_{steck}\} \end{aligned} \tag{9.2}$$

Das Bereitstellen eines Legosteins wird über einen Einzeltask mit der genannten Attributänderung beschrieben. In der Simulation führt dieser Task keine Bewegung im

eigentlichen Sinne aus und benötigt daher keine Zeit. Bei der realen Ausführung wird der Bereitsteller von einem Werker manuell aufgefüllt, sobald ein Legostein vom Roboter entnommen wurde.

Fähigkeitenmodul „Pick-and-Place“: Das Nehmen und Platzieren eines Legosteins wird für einen Roboter dann angeboten, wenn dieser sowohl die ursprüngliche als auch die neue Position des Legosteins kollisionsfrei anfahren kann. Dabei gibt es mehrere Möglichkeiten, wie der Stein gegriffen werden kann: Entlang einer Seite ist ein 2×2-Stein mit dem Parallelgreifer an drei Positionen anfahrbar – in der Mitte und jeweils am Rand. Gleiches gilt für die um $90°$ gedrehte Variante. Da die Orientierung bei Robotern eine Rolle spielt, ist jede der Optionen nochmals in gespiegelter Variante möglich. Insgesamt sind es 12 mögliche Greifpositionen für einen 2×2-Stein; ein 4×2-Stein besitzt 16 Möglichkeiten. Die Greifposition bestimmt die Qualität der Aktion: Je mittiger der Stein gegriffen wird, desto sicherer ist die Aufnahme und das Absetzen. Eine Pick-and-Place-Aktion bewirkt ein Abbauen von einem oder mehreren LegoSteckAttributen zu dem oder den darunterliegenden Legosteinen, i. Z. $a_1^-, \ldots, a_m^- \in \mathcal{A}$. Gleichermaßen erfolgt ein Aufbauen eines oder mehrerer neuer LegoSteckAttribute $a_1^+, \ldots, a_n^+ \in \mathcal{A}$ zu den Legosteinen am neuen Platz. Der Situationsübergang einer solchen Aktion einschließlich der oben genannten Konfigurationsänderung des Roboters ist durch folgende Gleichung 9.3 gegeben:

$$\begin{aligned} \mathit{Effekt}(t,s) &= \big(\{a'_{konf}, a_1^+, \ldots, a_n^+\}, \{a_{konf}, a_1^-, \ldots, a_m^-\}\big) \\ \mathit{nach}(t,s) &= \big(s \setminus \{a_{konf}, a_1^-, \ldots, a_m^-\}\big) \cup \{a'_{konf}, a_1^+, \ldots, a_n^+\} \end{aligned} \tag{9.3}$$

Eine Pick-and-Place-Aktion, die diese Attributänderung umsetzt, wird über einen hierarchisch gegliederten technischen Task beschrieben. Insgesamt neun Einzeltasks, im Folgenden (1) bis (9), werden in einer Sequenz in Reihe geschaltet und beschreiben damit die Greif-Strategie: Zunächst fährt der Roboter mit geöffnetem Greifer den aufzunehmenden Legostein mit kleinem Abstand von oben an (1). Um den Greifer über dem Stein trotz der Ungenauigkeiten der „echten Welt“ möglichst exakt auszurichten, schließt sich der Greifer nun soweit, dass er den Legostein kraftlos berührt (2). Anschließend fährt der Greifer von oben herab auf die Zielposition (3). Der Greifer schließt sich nun komplett (4) und fixiert damit den Legostein mit einer eindeutigen Position. Nach einem Abziehen des Legosteins nach oben (5) wird dieser zur Absetzposition (leicht oberhalb) gebracht (6). Um den Legostein abzusetzen, wird er senkrecht auf den oder die darunter befindlichen Steine gefahren (7), ehe sich der Greifer öffnet (8) und der Roboter mit leerem Greifer wieder senkrecht nach oben fährt (9). Die Ausführungszeit der Pick-and-Place-Aktion wird durch die Dauer der Einzeltasks bestimmt.

Fähigkeitenmodul „Support“: Um instabile Konstruktionen zu stabilisieren, bietet dieses Fähigkeitenmodul Aktionen an, in denen der Roboter die Konstruktion an geeigneter Stelle unterstützt. Hierbei gibt es für die konkret gewählte Stelle, also den Stein, der unterstützt wird, und die dabei eingenommene Haltung des Roboters eine Vielzahl an möglichen Optionen. Eine Support-Aktion bewegt den jeweiligen Roboter und bewirkt ein anschließendes Aufbauen eines SupportAttributs $a_{supp} \in \mathcal{A}$. Die Situationsänderung

ist demnach durch folgende Gleichung 9.4 vollständig beschrieben:

$$\begin{aligned} \mathit{Effekt}(t,s) &= \big(\{a'_{konf}, a_{supp}\}, \{a_{konf}\}\big) \\ nach(t,s) &= (s \setminus \{a_{konf}\}) \cup \{a'_{konf}, a_{supp}\} \end{aligned} \tag{9.4}$$

Eine Support-Aktion wird durch einen hierarchisch gegliederten technischen Task dargestellt. In mehreren Teilbewegungen wird der Greifer zunächst geschlossen und anschließend so ausgerichtet, dass er senkrecht von unten an den zu unterstützenden Legostein heranfahren und andocken kann.

Fähigkeitenmodul „Unsupport“: Soll ein bestehender Support wieder aufgelöst werden – etwa weil ein anderer Roboter den Support übernimmt oder die Konstruktion einen stabilen Zustand erreicht hat –, so bietet das Fähigkeitenmodul „Unsupport“ geeignete Aktionen dafür. Existiert ein bestehendes SupportAttribut $a_{supp} \in \mathcal{A}$, so bewirkt eine Unsupport-Aktion eine Bewegung des Roboters sowie den Abbau von a_{supp}. Die komplette Situationsänderung ist in folgender Gleichung 9.5 gegeben:

$$\begin{aligned} \mathit{Effekt}(t,s) &= \big(\{a'_{konf}\}, \{a_{konf}, a_{supp}\}\big) \\ nach(t,s) &= (s \setminus \{a_{konf}, a_{supp}\}) \cup \{a'_{konf}\} \end{aligned} \tag{9.5}$$

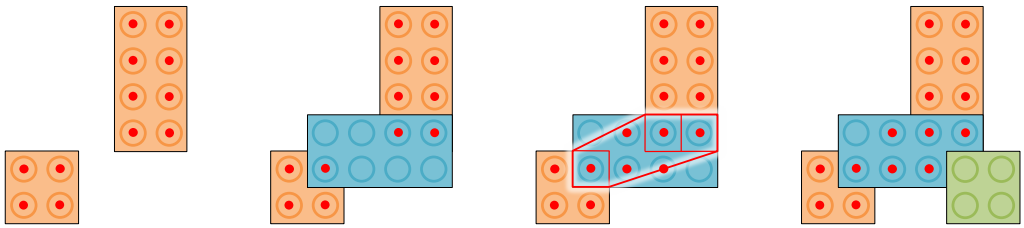
In umgekehrter Reihenfolge zu den Einzeltasks eines Supports wird bei einem Unsupport der Greifer nach unten bewegt, um die Kraft des Supports abzubauen, und über anschließende Bewegungen aus dem Bauteil herausgefahren.

Jedes der vier Fähigkeitenmodule ist einmal implementiert und wird mehrfach für die Aktuatoren der Roboterzelle instantiiert und wiederverwendet. Das Fähigkeitenmodul „Bereitstellen“ wird für jeden der vier Bereitsteller der LEGO-Fallstudie instantiiert. Die übrigen Fähigkeitenmodule „Pick-and-Place“, „Support“ und „Unsupport“ sollen von allen vier Robotern ausführbar sein und werden jeweils für alle vier instantiiert.

9.3.7 Bereitgestellte Validatormodule

Um bei der Fertigungsplanung schon frühzeitig ungültige Zwischenergebnisse zu erkennen und den Planungsaufwand dadurch zu reduzieren, werden Validatormodule zur Prüfung von Planungssituationen eingesetzt (siehe Abschnitte 4.1.4 und 7.2.1). Im Gegensatz zu Domänenprüfmodulen werden hierbei auch die Aktuatoren der Roboterzelle berücksichtigt. Für die LEGO-Fallstudie existieren zwei Validatormodule. Eines führt Kollisionsüberprüfungen zwischen den Objekten der Planung, also Legosteinen und Robotern, aus. Ein zweites überprüft die Stabilität des jeweiligen Zwischenprodukts (siehe Statik in Abschnitt 3.1.3).

Die Problemstellung der Kollision ist bereits in Abschnitt 3.1.3 erläutert und wird dort in Abbildung 3.2 dargestellt, die eine Kollision zwischen Greiferbacken und benachbarter Legosteine beim Absetzen eines Legosteins zeigt. Überprüfungen von Situationen nach jedem Einzeltask, um derartige Kollisionen festzustellen, mag jedes Fähigkeitenmodul für sich selbst ausführen und entsprechende Ergebnisse verwerfen. Doch erlaubt die Verwendung von Validatormodulen eine klare Trennung der Verantwortlichkeiten zwischen unterschiedlichen Expertenbereichen. In diesem Fall sind es Expertisen einerseits



(a) Feste Legosteine und belastbare Pins

(b) Aufgestecker Legostein

(c) Konvexe Hülle der belastbaren Pins

(d) Instabil plazierter Legostein

Abbildung 9.9. Berechnung belastbarer Pins anhand fester Legosteine (a) und konvexer Hüllen von überdeckten belastbaren Pins (b, c). Belastbare Pins sind durch rote Punkte symbolisiert.

im Bereich der Mathematik und der räumlichen Kollisionsberechnung, andererseits im Bereich von Montageprozessen, um Legosteine mit Robotern zu verarbeiten. Im Zuge der Planung findet eine Vielzahl an Kollisionsprüfungen statt, für jede resultierende Planungssituation nach jedem potentiellen Einzeltask. Um dies so effizient wie möglich zu gestalten, werden nicht immer alle kombinatorischen Zweierpaare an Objekten auf Kollisionen geprüft. Stattdessen werden diejenigen Objekte identifiziert, die sich im Vergleich zur vorhergehenden Situation, die als kollisionsfrei angenommen wird, bewegt haben. Lediglich für diese Objekte wird untereinander und mit den übrigen Objekten eine Kollisionsprüfung ausgeführt. Das Validatormodul nutzt dabei eine vom Planungsansatz $PaRTs$ angebotene Schnittstelle zur Kollisionsprüfung.

Das zweite Validatormodul, das für die LEGO-Fallstudie eingesetzt wird, führt eine Statikanalyse auf LEGO-Bauteilen aus, um instabile Konstruktionen zu erkennen. Obwohl eine Physiksimulation dafür zweckdienlich erscheint, würde dies bei einer Überprüfung aller möglichen Zwischenschritte der Planung die Planungszeit enorm in die Höhe treiben. Daher wird ein anderer, leichtgewichtiger Ansatz verfolgt, der das Ergebnis einer Statikanalyse für LEGO-Bauteile approximiert und dabei im Zweifelsfall schlechter schätzt. Dadurch ist eine Stabilität immer garantiert. Das Validatormodul bestimmt für jeden Legostein der Situation, ob es einen zusätzlichen Support benötigen würde. Konkret werden Legosteine in die Kategorien *stabil* und *instabil* eingeteilt. Ein stabiler Legostein leitet seine Gewichtskraft in geeigneter Weise auf benachbarte Steine ab. Existiert ein instabiler Legostein, ist die Konstruktion instabil. Weiterhin wird für jeden Pin eines Legosteins (repräsentiert durch einen Punkt) bewertet, ob dieser *belastbar* ist oder nicht. Auf einen belastbaren Pin kann ein weiterer Legostein gesetzt werden. Besitzt ein Legostein mindestens einen belastbaren Pin, so ist er stabil. Also: Besitzt jeder Legostein einer Konstruktion mindestens einen belastbaren Pin, so ist die Konstruktion stabil. Legosteine oder Basisplatten, die über ein Fixpunkt-Attribut fest in der Roboterzelle verankert sind, werden als *fest* bezeichnet und besitzen ausschließlich belastbare Pins.

Abbildung 9.9 zeigt aus einer Sicht von oben den schrittweisen Aufbau einer Beispielkonstruktion aus LEGO und erläutert die Bewertung von belastbaren Pins. Beide Legosteine in Abbildung 9.9a seien fest am Boden verankert (z. B. über Fixpunkt-Attribute). Demnach sind alle ihre Pins belastbar. Wird ein zweiter Legostein, wie in Abbildung 9.9b abgebildet, von oben aufgesteckt, so bestimmt die konvexe Hülle über alle unmittelbar

darunterliegenden belastbaren Pins die stabilen Pins des neuen Legosteins (siehe Abbildung 9.9c). Wird ein Legostein ausschließlich auf unbelastbare Pins gesetzt (siehe Abbildung 9.9d), so besitzt dieser selbst keinen belastbaren Pin und er ist instabil.

Für einfache Konstruktionen wie einen Turm ist die Bestimmung der Stabilität sehr einfach. Für manch andere Konstruktionen (wie z. B. in Abbildung 9.9 abgebildet) kann die Stabilität damit ebenfalls mit überschaubarem Aufwand berechnet werden. Komplexere Konstruktionen mit *dazwischen hängenden* Legosteinen (z. B. LEGO-Brücke) oder Bauteile, die von einem Roboter gestützt werden, sind mit dem vorgenannten Vorgehen allerdings nicht korrekt klassifizierbar, weil sie von ihm als instabil eingestuft werden. Damit auch derartige Konstruktionen berücksichtigt werden können, werden weitere Legosteine als *fest* definiert. Als *fest* gelten diejenigen Legosteine, die von einem Roboter durch ein SupportAttribut gestützt werden oder die auf einer Teilkonstruktion zwischen zwei *festen* Legosteinen sitzen. Mit letzterem ist nun auch ein überspannender Bogen (z. B. die LEGO-Brücke) als stabil identifizierbar.

Algorithmus 2 beschreibt das Vorgehen des Validatormoduls zur Überprüfung der Stabilität eines LEGO-Bauteils mit der Funktion $validiere$, die eine aktuelle Planungssituation $s \in \mathcal{S}$ als Parameter erhält. Der Algorithmus stützt sich bei der Berechnung auf zwei globale Mengen: Die Menge E enthält alle Legosteine, die zu der gegebenen Situation im Bauteil montiert sind (Zeile 4). Hiervon werden alle Legosteine in die Menge E_{fest} aufgenommen, die als *fest* identifiziert werden. Diese Bestimmung wird in den Zeilen 5 bis 11 ausgeführt. Ein Legostein ist zum einen dann *fest*, wenn er mindestens einen Fixpunkt oder ein SupportAttribut besitzt (Zeile 7). Zum anderen wird ein Legostein auch dann als *fest* markiert, wenn er auf einem transitiven Attribut-Pfad zwischen zwei ebenfalls *festen* Steinen sitzt und bei Draufsicht auf die XY-Ebene innerhalb deren konvexen Hülle liegt (Zeile 10). Die Stabilitätseinschätzung von Brückenbögen wird dadurch ermöglicht. Die Abschätzung wertet allerdings ggf. auch sehr große Konstruktionen als stabil ein, obwohl diese der hohen Eigenlast in Realität nicht standhalten würden.

Basierend auf den *festen* Legosteinen werden zu jedem Legostein die jeweiligen belastbaren Pins ermittelt (Zeile 12). Die Funktion $belastbarePins$ bestimmt dazu für einen bestimmten Legostein und eine gegebene Situation die belastbaren Pins gemäß des oben beschriebenen Vorgehens zum Beispiel aus Abbildung 9.9. Ist der Legostein als *fest* markiert, sind alle seine Pins belastbar (Zeile 15). Andernfalls werden die unterhalb des Legosteins befindlichen Steine zur weiteren Untersuchung herangezogen. In der Menge P_{bel} werden dazu sämtliche belastbaren Pins der unmittelbar darunterliegenden Legosteine gesammelt. Dies ist der Fall, wenn ein Legostein ein gemeinsames Attribut mit dem untersuchten Stein besitzt – es existiert also eine Steckverbindung – und der Legostein der untere von beiden ist (Zeile 17). Die belastbaren Pins der darunterliegenden Steine werden dabei über Rekursion ermittelt. Mit denjenigen Pins von P_{bel}, die mit den Pins des untersuchten Legosteins bei Draufsicht deckungsgleich sind, wird eine konvexe Hülle aufgespannt (Zeile 19). Alle Pins des untersuchten Legosteins, die innerhalb dieser konvexen Hülle in XY-Ebene liegen, sind belastbar. Das Gesamtbauteil, das in der Situation beschrieben ist, wird als stabil beurteilt, wenn jeder Legostein der Konstruktion mindestens einen *festen* Pin aufweisen kann (Zeile 12).

Algorithmus 2 : Statikü̈berprüfung für LEGO-Strukturen

Input : $s \in \mathcal{S}$: zu überprüfende Planungssituation
Output : $boolean$: Gesamtkonstruktion ist stabil (**true**) oder instabil (**false**)

```
1  global E := ∅;                          // Menge aller Legosteine der gegebenen Situation
2  global E_fest := ∅;                     // Menge der als fest markierten Legosteine
3  Function validiere(s):
4      E ← {e ∈ ℰ | attr(e, s) ≠ ∅};       // Bestimmung aller in s montierten Legosteine
       /* Markiere all diejenigen Legosteine als fest, die in der gegebenen Situation über ein
          Fixpunkt-Attribut oder ein SupportAttribut verfügen                          */
5      E_fest ← ∅;
6      foreach (e ∈ E) do
7          if (∃a ∈ attr(e, s). a ist ein Fixpunkt oder ein SupportAttribut) then
8              E_fest ← E_fest ∪ {e};
       /* Markiere Legosteine als fest, die entspr. zwischen anderen festen Steinen liegen */
9      foreach (e ∈ E \ E_fest) do
10         if (∃e1, e2 ∈ E_fest. e liegt auf transitivem Attribut-Pfad zw. e1 und e2
             und e liegt innerhalb konvexer Hülle von e1 und e2 in XY-Ebene) then
11             E_fest ← E_fest ∪ {e};
12     return ∀e ∈ E. |belastbarePins(e, s)| > 0;

   /* Bestimmung all derjenigen Pins eines Legosteins e ∈ ℰ, die in einer gegebenen Situation
      s ∈ 𝒮 belastbar sind                                                             */
13 Function belastbarePins(e, s):
14     if (e ∈ E_fest) then
15         return pins(e);                  // Alle Pins von Legostein e
       /* Ermittlung der belastbaren Pins der darunterliegenden Legosteine             */
16     P_bel := ∅;
17     foreach (e1 ∈ E. attr(e1, s) ∩ attr(e, s) ≠ ∅ und e1 unterhalb von e) do
18         P_bel ← P_bel ∪ belastbarePins(e1, s);
       /* Welche der Pins des Legosteins liegen innerhalb der konvexen Hülle, die von den
          überdeckten belastbaren Pins der darunterliegenden Legosteine aufgespannt wird? */
19     huelle := konvexe XY-Hülle von P_bel ∩ pins(e);
20     return {p ∈ pins(e) | p liegt innerhalb huelle in XY-Ebene};
```

9.4 Evaluation

Um den Planungsansatz $PaRTs$ zu evaluieren, wird sowohl das LEGO-Haus als auch die LEGO-Brücke jeweils mit drei unterschiedlich implementierten Planern geplant. Diese unterscheiden sich in der Strategie, wie sie den Zustandsraum aufspannen und diesen nach einer geeigneten Lösung durchsuchen. Zwei Planer verwenden klassische Suchstrategien, zum einen die Tiefensuche, zum anderen die informierte Suche A^*. Als dritter Planer kommt der vorgestellte Planungsansatz $PaRTs$ zum Einsatz.

9.4.1 Getrennte Modellierung von Expertenbeiträgen

Damit die Fertigung von LEGO-Produkten mit allen drei Planern automatisch geplant werden kann, wurden insgesamt neun Expertenmodule implementiert. Zu Beginn jeder Evaluation wird das Analysemodul eingesetzt, um die Attribute des fertigen Bauteils zu bestimmen und damit die Ziel-Produktsituation festzulegen. Desweiteren verwenden alle drei Planer dieselben Module, um den Zustandsraum aufzuspannen. Über die vier Fähigkeitenmodule, die das Wissen von Automatisierungs- und Prozessexperten bündeln, wird ermittelt, welche weiterführenden technischen Tasks als Transitionen in einem gegebenen Zustand möglich sind. Die jeweils daraus resultierende Planungssituation wird über das Domänenprüfmodul auf legospezifische Eigenschaften sowie über zwei Validatormodule auf Gültigkeit im Kontext mit der Roboterzelle überprüft. Nach jedem Schritt der Planung kann damit ein ungültiger Planungszweig beendet werden. Lediglich das Domänentaskmodul, das nächstmögliche abstrakte Aktionen am Bauteil ermittelt, wird ausschließlich vom $PaRTs$-Planer verwendet, da sowohl Tiefensuche als auch A^* keine Untergliederung in eine Prozess- und eine Fertigungsplanung vornehmen.

Insgesamt erlaubte $PaRTs$ bei der Implementierung der Expertenmodule eine klare Trennung der Kompetenzen unterschiedlicher Experten. So war die Anfertigung der Fähigkeitenmodule mit dem Wissen eines Prozess- und Automatisierungsexperten möglich, der über die notwendige Kenntnis zur Roboteransteuerung verfügt, um Legosteine korrekt zu greifen, zusammenzustecken und zu unterstützen. Module, die ein fundiertes Fachwissen über LEGO voraussetzen – z. B. über die Stabilität von Legokonstruktionen – bedurften der Entwicklung durch einen Domänenexperten. Die Granularität der Expertenmodule und die Aufteilung ihrer Zuständigkeiten, die der Planungsansatz vorgibt, machte es möglich, dass alle drei Planer trotz ihrer unterschiedlichen Vorgehensweise auf dieselben Module zurückgreifen konnten und für keinen der Planer eine weitere LEGO- oder roboterspezifische Implementierung angefertigt werden musste.

9.4.2 Vergleich gegenüber klassischen Suchstrategien

Für die Evaluation werden unterschiedliche Szenarien untersucht, die sich in drei variablen Faktoren unterscheiden:

- **Montageproblem:** Es stehen zwei unterschiedliche Produkte aus LEGO für die Planung zur Verfügung: das LEGO-Haus und die LEGO-Brücke.

- **Planungsansatz:** Drei Planer mit einer jeweils unterschiedlichen Suchstrategie werden jeweils auf alle Montageprobleme angewandt: Tiefensuche, a^* und der vorgestellte Planungsansatz $PaRTs$.
- **Roboterzelle:** Die Planung des LEGO-Hauses wird für den Einsatz zum einen von nur einem Roboter und zum anderen von drei Robotern untersucht. Da die LEGO-Brücke mit nur einem Roboter nicht gefertigt werden kann, wird ihre Fertigungsplanung lediglich für drei Roboter durchgeführt.

Bevor in die Auswertung der Evaluierung eingestiegen wird, ist auf eine Sache vorab besonders hinzuweisen: Die Ergebnisse desselben Szenarios werden aggregiert und in einen Mittelwert und eine Standardabweichung überführt. Eine Standardabweichung ungleich null verrät, wie stark sich die Ergebnisse der einzelnen Planungsdurchgänge voneinander unterscheiden. Doch müsste der selbe Planer für das selbe Montageproblem und die selbe Roboterzelle denn nicht deterministisch stets das gleiche Resultat liefern? Für eine Erklärung ist etwas weiter auszuholen: Jede der drei Suchstrategien entscheidet, welcher Zustand und welche Aktion im nächsten Schritt selektiert und weitergehend untersucht wird. Die Tiefensuche wählt aus den unmittelbar nächsten Aktionen des aktuellen Zustands aus, bei A^* wird aus einer globalen Menge aller bereits bekannten Zustände der vielversprechendste weiterverfolgt, und $PaRTs$ wählt aus den im Korridor der Planung liegenden Aktionen. Die Entscheidung, welche der alternativ zur Verfügung stehenden Möglichkeiten gewählt wird, basiert auf den Kosten, die die Aktionen verursachen. Hierfür werden bei allen drei Strategien zum einen die Ausführungsdauer der geplanten Aktionen, zum anderen die Qualität, also z. B. wie prozesssicher die Greifposition des Legosteins ist, zur Bewertung herangezogen. Für A^*, der einerseits als eigener Planer und andererseits im Rahmen der Fertigungsplanung auch bei $PaRTs$ zum Einsatz kommt, werden zusätzlich die noch zu erwartenden Kosten eines unfertigen Plans geschätzt, um daraus fiktive Gesamtkosten als Fitnessfunktion von A^* zu erhalten. Diejenige Aktion mit den geringsten Kosten (Tiefensuche) bzw. mit dem besten Fitnesswert (A^*) wird als nächste Aktion des Plans gewählt. Sind die Kosten zweier Aktionen jedoch gleich – und dies kommt bei diskreten Montageproblemen wie LEGO desöfteren einmal vor –, so entscheidet das Los. Durch den hier wirkenden Zufall sind die teils größer ausfallenden Standardabweichungen der Evaluation zu erklären. Der Indeterminismus an dieser Stelle ist dabei gewollt; führen ungünstige Entscheidungen zu einer äußerst ineffizienten Planungsdauer, so ist dies dank Indeterminismus nicht auch für alle Folgedurchgänge festgeschrieben.

Ein zweiter Hinweis sei im Hinblick auf die Optimalität der gefundenen Pläne gegeben. Üblicherweise findet A^* das globale Optimum, also den bestmöglichen Pfad an Aktionen, um das Bauteil herzustellen. Bei den Testläufen mit A^* ist dies allerdings nicht der Fall, weshalb in den Auswertungen auch bei der Ergebnisqualität eine Standardabweichung existiert. Das Finden des Optimums setzt nämlich voraus, dass die Kostenschätzung der noch fehlenden Aktionen annähernd perfekt ist. Werden für einen aktuellen, halb fertigen Plan die noch ausstehenden Kosten zu hoch geschätzt, wird die nächste erstbeste Aktion direkt gewählt – denn die tatsächlichen Kosten der Aktion wirken ausgesprochen gut im Vergleich zur Schätzung. Das Planungsverhalten ähnelt hier dem einer Tiefensuche, wobei das Finden des Optimums nicht garantiert ist. Werden die Kosten

der noch bevorstehenden Aktionen dagegen zu niedrig geschätzt, so erscheint jede nächste Aktion zu teuer. Daraufhin werden alle Alternativen untersucht, um die günstigste Aktion zu identifizieren. Das Planungsverhalten gleicht in diesem Fall dem einer Breitensuche und ist entsprechend ineffizient. Für LEGO basiert die Kostenschätzung auf fiktiven Ausführungsdauern derjenigen Tasks, die für alle noch nicht fertiggestellten Legosteine benötigt werden. Leider ist nur sehr schwer abzuschätzen, ob ein Roboter für einen Task eine weite oder kurze Strecke zurückzulegen hat und damit eine lange oder kurze Ausführungsdauer besitzt. Daher ist auch eine realistische Einschätzung der zu erwartenden Kosten sehr schwer. Es wurde ein Trade-off zwischen Über- und Unterabschätzung gewählt, der eine relativ performante Planung ermöglicht, dafür aber anstelle des Optimums lediglich ein relativ gutes Ergebnis liefert.

Alle drei Planer sowie sämtliche Expertenmodule sind in der Programmiersprache Java implementiert. Alle Evaluationen werden in einer JVM auf einem Linux-PC ausgeführt, der über 16 GB RAM und eine Intel(R) Xeon(R) CPU E31230 (3.20GHz) mit 4 Kernen und aktiviertem Hyper-Threading verfügt. Um statistisch aussagekräftige Ergebnisse zu erhalten, wird jedes Szenario 250 mal ausgeführt. Die maximale Programmlaufzeit, also die maximale Planungszeit, ist auf 300 Sekunden (5 Minuten) festgelegt. Planungsläufe, die diese Zeitschranke überschreiten, werden abgebrochen und nicht in den statistischen Aggregationen berücksichtigt.

Die nachfolgend aufgeführten Evaluationsläufe der verschiedenen Szenarien werden jeweils auf mehrere Faktoren untersucht. Neben der Anzahl der insgesamt 250 Läufe, die innerhalb des Zeitlimits von 300 Sekunden mit einer gültigen Lösung terminieren, wird zu jedem Szenario der Durchschnittswert und die Standardabweichung zur Planungszeit sowie zu den untersuchten Zuständen und Transitionen angegeben. Die gefundenen Lösungen werden anhand der Anzahl angewandter Fähigkeiten, der technischen Tasks sowie ihrer Ausführungsdauer untersucht. Auch hierzu sind jeweils der Durchschnittswert und die Standardabweichung angegeben.

Montageplanung des LEGO-Hauses für einen einzelnen Roboter:

Die Hauptherausforderung des LEGO-Hauses an die Planer ist der Umgang mit Deadlocks, die insbesondere nicht in dem Moment identifiziert werden können, in dem sie entstehen (siehe Eckstein-Problem in Abbildung 6.8, Abschnitt 6.2.2). In Tabelle 9.1 zeigt sich bei der Planung für einen einzelnen Roboter bereits der gravierende Nachteil der Tiefensuche, die lediglich in 79 von 250 Läufen innerhalb der maximal vorgegebenen 300 Sekunden eine Lösung findet. Verläuft sich die Suche in einen „toten Ast“, so wird dieser vollständig exploriert, obwohl darin keine Lösung existiert. Dies schlägt sich eindeutig in den Planungszeiten nieder: Mehr als 36 Sekunden werden im Durchschnitt mit einer Standardabweichung von ca. 55 Sekunden benötigt, um die 79 Lösungen zu finden. Dabei werden im Schnitt 9431 Zustände und über 260 000 Transitionen untersucht. Auch hier sind die Standardabweichungen entsprechend hoch.

Anders sieht es dagegen bei A^* und $PaRTs$ aus. Da A^* die Pfade anhand von Fitnesswerten selektiert, wird nicht zwangsläufig jede Option eines „toten Asts“ untersucht. Alle der 250 Läufe terminieren daher in knapp 2 Sekunden mit einer Standardabweichung von ca. 4 Sekunden. Aber auch hier ist das Deadlockproblem klar bemerkbar, da die

Tabelle 9.1. Montage des LEGO-Hauses mit einem Roboter. Evaluationsergebnisse für Tiefensuche, A^* und $PaRTs$.

Lösungsansatz:	Tiefensuche	A^*	$PaRTs$
Performanz der Planung			
Erfolgreiche Läufe:	79 / 250	250 / 250	250 / 250
Planungszeit:	36,57 s ± 54,88 s	2,20 s ± 4,24 s	0,40 s ± 0,31 s
Untersuchte Zustände:	9431 ± 11 279	1323 ± 2254	270 ± 195
Unters. Transitionen:	262 116 ± 313 468	7181 ± 13 145	4251 ± 3473
Ergebnisqualität der Planung			
Angew. Fähigkeiten:	52 ± 0	52 ± 0	52 ± 0
Technische Tasks:	338 ± 0	338 ± 0	338 ± 0
Ausführungsdauer:	230,35 s ± 0,15 s	230,39 s ± 0,17 s	230,38 s ± 0,15 s

Standardabweichung deutlich größer als der Durchschnittswert ist – einige Läufe benötigen beispielsweise ein Dreifaches an Planungszeit im Vergleich zu anderen. $PaRTs$ steht hier deutlich besser da mit einer Planungszeit von unter einer Sekunde und einer noch geringeren Standardabweichung von $0{,}31$ Sekunden. Analog dazu verhält sich die Anzahl der untersuchten Zustände und Transitionen. Betrachten wir die Ergebnisse der Planungsläufe, so fällt auf: Ausnahmslos alle Ergebnisse weisen dieselbe Menge an auszuführenden technischen Tasks auf. Das sind 52 angewandte Fähigkeiten, also das Bereitstellen und ein Pick-and-Place für jeden der 26 Legosteine des Hauses. Jedes Bereitstellen besteht aus einem technischen Task, ein Pick-and-Place wird über 12 kombinierte Einzeltasks realisiert. Daraus ergeben sich für jedes Planungsergebnis 338 technische Tasks. Auch die Ausführungsdauer der gefundenen Ergebnisse ist in allen drei Szenarien ähnlich, was darauf hindeutet, dass die von den Fähigkeitenmodulen erzeugten technischen Tasks nur geringe Variabilität pro Legostein aufweisen.

Montageplanung des LEGO-Hauses für drei Roboter:

Tabelle 9.2 zeigt die statistischen Ergebnisse der Planung des LEGO-Hauses für drei Roboter. Alle Evaluationsläufe, die innerhalb des Zeitlimits von 300 Sekunden terminieren, liefern wie auch bei der Planung für nur einen einzelnen Roboter Ergebnisse mit gleichbleibender Anzahl an angewandten Fähigkeiten und an technischen Tasks. Obwohl durch die Hinzunahme weiterer Fähigkeiten der zu durchsuchende Zustandsraum deutlich komplexer wird, kommt die Tiefensuche damit besser klar. In nun 191 von 250 Fällen findet der Planer innerhalb von knapp 4 Sekunden eine Lösung. Die Planungszeit und die untersuchten Zustände reduzieren sich auf einen Bruchteil im Vergleich zur Planung für einen einzelnen Roboter; für eine Lösung mit 52 Schritten müssen durchschnittlich nur 159 Zustände untersucht werden. Warum aber schneidet die Tiefensuche in einem scheinbar komplexeren Szenario mit drei Robotern sogar besser ab? Vermutlich führt bei den nun drei zur Verfügung stehenden Robotern die Auswahl des Tasks in jedem Schritt, der dort der Beste ist, nicht mehr so oft in die Konstellation eines Deadlocks. Auch wenn die Ergebnisse hier sehr gut sind, so zeigt

Tabelle 9.2. Montage des LEGO-Hauses mit drei Robotern. Evaluationsergebnisse für Tiefensuche, A^* und $PaRTs$.

Lösungsansatz:	Tiefensuche	A^*	$PaRTs$
Performanz der Planung			
Erfolgreiche Läufe:	191 / 250	172 / 250	250 / 250
Planungszeit:	4,32 s ± 11,50 s	20,94 s ± 46,88 s	1,98 s ± 1,62 s
Untersuchte Zustände:	159 ± 463	5792 ± 12 486	676 ± 461
Unters. Transitionen:	13 168 ± 37 724	48 751 ± 102 045	20 078 ± 15 944
Ergebnisqualität der Planung			
Angew. Fähigkeiten:	52 ± 0	52 ± 0	52 ± 0
Technische Tasks:	338 ± 0	338 ± 0	338 ± 0
Ausführungsdauer:	166,30 s ± 17,03 s	174,13 s ± 16,24 s	90,31 s ± 3,44 s

die zusammenhanglose Beziehung zwischen Roboteranzahl und Deadlockproblematik, dass die Performanz der Tiefensuche durch viele Einflüsse bedingt ist und daher schwer vorhersagbar ist. Verschiebt man beispielsweise einen Roboter um wenige Zentimeter in der Roboterzelle, so mag eine erneute Planung eventuell katastrophal schlechte Ergebnisse erzielen.

Im Gegensatz zur Tiefensuche bricht A^* bei der Planung mit drei Robotern hingegen deutlich ein. Der mit den Fähigkeiten exponentiell wachsende Zustandsraum wird bei den fast 5800 untersuchten Zuständen und 4800 Transitionen sichtbar. Dadurch verschlechtert sich die Planungszeit um fast das Zehnfache im Vergleich zur Planung für nur einen einzelnen Roboter. Die im Verhältnis zu den Durchschnittswerten analog mitwachsenden Standardabweichungen bei der Planungszeit und den untersuchten Zuständen und Transitionen weisen darauf hin, dass das Deadlockproblem prozentual gesehen noch dieselbe Auswirkung auf die Planung mit A^* hat. Die Anzahl der Roboter beeinflusst somit die Bewältigung von Deadlocks nicht maßgeblich. Bei einem Vergleich der Ausführungsdauer mit der Tiefensuche fällt das schlechte Ergebnis von A^* auf. Warum ist das so? Das gute Ergebnis der Tiefensuche kann beispielhaft so erklärt werden: Das Bereitstellen von Legosteinen ist in der Regel günstiger als ein Pick-and-Place. Somit werden in den ersten drei Schritten zunächst die drei Bereitsteller aufgefüllt. Das anschließende Pick-and-Place der Legosteine kann mit den drei Robotern weitgehend parallel ausgeführt werden, wobei alle Bereitsteller wieder aufgefüllt werden, sobald ihr Legostein entnommen ist. Für A^* hingegen scheint die Länge von 52 Planungsschritten (angewandte Fähigkeitenmodule) bereits dazu zu führen, dass die Schätzung der erwarteten Kosten entsprechend weit von den realen Kosten abweicht.

Anders sieht es bei $PaRTs$ aus. Alle der 250 Läufe terminieren innerhalb des Zeitlimits in durchschnittlich unter zwei Sekunden. Auch die Standardabweichung ist sehr gering, sodass die Planung deutlich schneller ist als die von Tiefensuche und A^*. Obwohl für die Bestimmung des Fitnesswerts von Plänen dieselbe Berechnung zugrunde liegt wie bei A^*, reduziert sich die erzielte Ausführungsdauer der Ergebnisse bei $PaRTs$ auf fast nur

Tabelle 9.3. Montage der LEGO-Brücke mit drei Robotern. Evaluationsergebnisse für Tiefensuche, A^* und $PaRTs$.

Lösungsansatz:	Tiefensuche	A^*	$PaRTs$
Performanz der Planung			
Erfolgreiche Läufe:	0 / 250	2 / 250	250 / 250
Planungszeit:	–	164,57 s ± 61,62 s	63,33 s ± 48,95 s
Untersuchte Zustände:	–	17 872 ± 12 492	5085 ± 3864
Unters. Transitionen:	–	461 963 ± 320 612	360 403 ± 295 261
Ergebnisqualität der Planung			
Angew. Fähigkeiten:	–	140,5 ± 0,7	141,3 ± 1,1
Technische Tasks:	–	871,0 ± 1,4	873,5 ± 3,2
Ausführungsdauer:	–	487,41 s ± 1,53 s	435,51 s ± 32,36 s

noch die Hälfte. Da hier lediglich innerhalb von Domänentasks auf den A^*-Algorithmus zurückgegriffen wird, ist diese Suche klein genug, sodass die Ungenauigkeit der Kostenschätzung keine allzu großen Auswirkungen hat. Die geringe Standardabweichung der Ausführungsdauer zeigt, dass $PaRTs$ sehr zuverlässig stets ähnlich gute Ergebnisse erzielt. Im Vergleich zur Planung für nur einen einzelnen Roboter ist dank der parallelen Ausführung nun die Erwartung, dass sich die Ausführungszeit mit drei Robotern deutlich verringert – im besten Fall auf ein Drittel der Zeit. Und tatsächlich beträgt die Ausführungsdauer mit drei Robotern nur $40\,\%$ der Ausführungsdauer mit einem einzelnen Roboter. Interessant ist, das $PaRTs$ bei durchschnittlich mehr untersuchten Zuständen und Transitionen deutlich kürzere Planungszeiten aufweisen kann als die Tiefensuche. Berücksichtigt sind hier die von den Fähigkeitenmodulen tatsächlich bereitgestellten Transitionen. Neben diesen werden von den Fähigkeitenmodulen allerdings noch weitere Optionen untersucht, die ggf. schon frühzeitig als ungültig erkannt und somit erst gar nicht als Transitionen angeboten bzw. in der Statistik berücksichtigt werden. Dies ist dann der Fall, wenn beispielsweise für die Fähigkeit Pick-and-Place der Roboter die Aufnahme- oder Ablageposition nicht oder nur mit Kollisionen erreichen kann. Dieser zusätzliche Berechnungsaufwand für derartige Aktionen ist bei der Tiefensuche sehr hoch, da in jedem Zustand sehr viele mögliche Optionen denkbar sind. Um das Optimum für einen Planungsschritt herauszufinden, müssen alle – auch die letztendlich nicht anwendbaren – Aktionen zumindest evaluiert werden. Bei $PaRTs$ hingegen führt der Korridor-geführte Suchansatz dazu, dass pro Planungsschritt nur ein Bruchteil der Optionen überhaupt betrachtet werden muss. So muss etwa eine nie erfüllbare Aktion, die bei der Tiefensuche in jedem Schritt betrachtet wird, bei $PaRTs$ lediglich im Rahmen des betreffenden Domänentasks berücksichtigt werden.

Montageplanung der LEGO-Brücke für drei Roboter:

Für den Bau der LEGO-Brücke sind mindestens drei Roboter erforderlich, da die Fertigung zwischenzeitlich geeignete Unterstützungen der Konstruktion benötigt. Die Ergebnisse der Evaluationsläufe mit drei Robotern sind in Tabelle 9.3 dargestellt. Mit

der Tiefensuche konnte für keine der 250 Durchläufe eine Lösung innerhalb der vorgegebenen Zeit gefunden werden. Zu wenige Pfade des Zustandsraums führen zu einem gültigen Ergebnis. Da die weitere Fertigung oftmals davon abhängt, dass irgendwann zuvor bereits ein geeigneter Support hergestellt wurde, hat die „kurzsichtige“ Tiefensuche bei einem derart komplexen Planungsproblem mit 64 Legosteinen und neun Fähigkeiten keine Chance, in annehmbarer Zeit eine Lösung zu finden.

Die Planung mit A^* nutzt mit ihrer Fitnesswertberechnung zwar einen zielgerichteten Ansatz, doch auch für sie stellt die Brücke eine große Herausforderung dar; sie terminiert in lediglich zwei von den 250 Läufen innerhalb der geforderten Zeit. Als kurzer Einschub sei hier erwähnt, dass auch bei Verdoppelung des Zeitlimits von 300 Sekunden auf zehn Minuten eine ähnlich geringe Erfolgsquote (also um die zwei von 250) beobachtet wurde. Es ist also nicht der Fall, dass viele Planungsläufe das Zeitlimit nur knapp verfehlen. Stattdessen verlieren sich die allermeisten Läufe in nicht zielführenden Pfaden – lediglich sehr wenige sind in der Lage, aufgrund zufällig gut getroffener Planungsentscheidungen rechtzeitig eine Lösung zu finden. Hintergrund für die niedrige Erfolgsquote ist, dass ein Support für den späteren Montageverlauf zwar zwingend erforderlich ist, dieser aber keinen Profit im Fitnesswert bringt. Denn der Support bringt zunächst keinen Fortschritt, kostet aber Zeit. Daher wird dieser – wenn überhaupt – nur nachrangig berücksichtigt. Im Durchschnitt benötigt jeder der zwei Läufe über 2,5 Minuten, um knapp 18 000 Zustände und 462 000 Transitionen zu untersuchen. Dabei zeigt die jeweils hohe Standardabweichung, dass beide Planungsläufe sehr unterschiedlich verlaufen sind. Und trotzdem sind ihre Lösungen nahezu identisch. Die Ausführung des Programms, das durch die Anwendung von rund 140 Fähigkeiten circa 871 technische Tasks enthält, dauert in beiden Fällen an die 5 Minuten. Allerdings ist bei nur zwei vergleichbaren Werten keine statistische Signifikanz gegeben.

Bei $PaRTs$ ist die Standardabweichung der Ausführungsdauer deutlich höher, d. h. die Ergebnisse variieren erkennbar bei mehrfacher Planung. Dennoch sind die erzielten durchschnittlichen 435 Sekunden Ausführungsdauer um rund 50 Sekunden schneller als die der Ergebnisse von A^*. Besonders hervorzuheben ist, dass $PaRTs$ in allen der 250 Läufe eine gültige Lösung gefunden hat. Wie bei A^* hat natürlich auch $PaRTs$ das Problem, dass die zwingend notwendigen Supports keinen positiven Fitnesswert besitzen. Da die Planung bei $PaRTs$ allerdings abgeschlossen für einzelne Domänentasks stattfindet, ist hier der Suchraum derart klein, dass der notwendige Support zum Lösen des Domänentasks schnell identifiziert werden kann. Zwar ist auch bei $PaRTs$ die Standardabweichung der Planungsdauer mit fast 50 Sekunden sehr hoch, doch zeigt der Durchschnittswert von circa einer Minute sehr deutlich, dass $PaRTs$ hervorragend für die Planung von Montageaufgaben in dieser Komplexitätsklasse geeignet ist.

Die zweischichtige Planung in Verbindung mit dem Korridor-geführten und pessimistischen Suchansatz zeigt hier seinen klaren Vorteil gegenüber den klassischen Strategien Tiefensuche und A^*, wobei die Planungsgrundlagen (Montageproblem, Roboterzelle, Expertenmodule) für alle Planer dieselben waren. Für den realen Einsatz von $PaRTs$ in der Fertigung – sofern sämtliche Expertenmodule bereits existieren – steht nun eine Minute Planungszeit gegen sieben Minuten Ausführungszeit. Eine manuelle Programmierung solcher Programme und das Teachen von Robotern würde ein Vielfaches an Zeit in

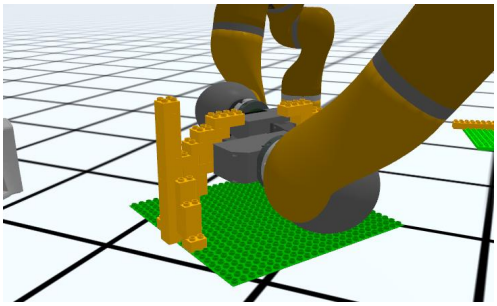
Anspruch nehmen. Damit ist $PaRTs$ ein vielversprechender Ansatz für die zukünftige Roboterprogrammierung.

9.4.3 Roboter-Kooperation

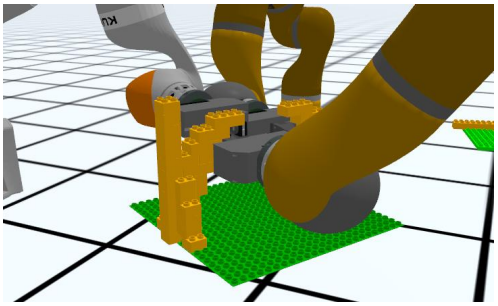
Bei Betrachtung der Planung für Multiroboter-Anwendungen besitzen beide LEGO-Strukturen – das Haus und die Brücke – jeweils einen unterschiedlichen Schwerpunkt. Während das Haus mit seinen zahlreichen eng beieinanderliegenden Legosteinen überwiegend auf Kollisionen der Roboter in einem kleinen gemeinsamen Arbeitsraum abzielt und außerdem die mit dem Eckstein einhergehenden Planungsprobleme mit Deadlocks provoziert, fordert die Brücke vorrangig das Finden von impliziten Kooperationen der Roboter untereinander. Eine Schlüsselstelle bei der Montage der Brücke ist das Absetzen des obersten Verbindungsstücks des Brückenbogens, der beide Bogenhälften miteinander verbindet und gegeneinander abstützt. Erst ab diesem Schritt sind die Bogenhälften stabil und müssen nicht mehr von einem Roboter gestützt werden. Für den weiteren Aufbau der Brücke ist ab dieser Stelle eine besondere Kooperation aller drei Roboter notwendig, deren Verlauf in Abbildung 9.10 dargestellt ist.

In Abbildung 9.10a ist der Montagezustand abgebildet, in dem zwei Roboter (R1 und R2) die Bogenhälften am jeweils zweitobersten Legostein stützen. Dabei hat kurz zuvor Roboter R3 die obersten beiden 4×2-Legosteine montiert. Das Problem der vorliegenden Situation ist nun, dass Roboter R3 den Verbindungsstein nicht montieren kann. Denn die Pins der Legosteine, auf die der Verbindungsstein zu setzen ist, sind in der aktuellen Situation nicht belastbar – die Steine würden dabei vorne überkippen. Für eine prozesssichere Montage müssten anstelle der zweitobersten nun die obersten Legosteine gestützt werden. Um aber diese Zielkonfiguration zu erreichen, ist eine umfassendere Durchwechselaktion der Roboter notwendig, denn einfach „loslassen“ darf keiner von ihnen. Daher übernimmt Roboter R3 den Support der linken Bogenhälfte am obersten Stein (siehe Abbildung 9.10b) und der vordere Roboter R1 kann seinen Support der Bogenhälfte gefahrlos aufheben (siehe Abbildung 9.10c). Roboter R1 kann nun den obersten Stein der anderen, rechten Bogenhälfte stützen (siehe Abbildung 9.10d) und ermöglicht es dadurch Roboter R2 (hinten rechts), seinen Support aufzulösen (siehe Abbildung 9.10e). Beide Bogenhälften werden nun an ihrem obersten Legostein gestützt und Roboter R2 kann den Verbindungsstein holen und an der Zielstelle absetzen (siehe Abbildung 9.10f).

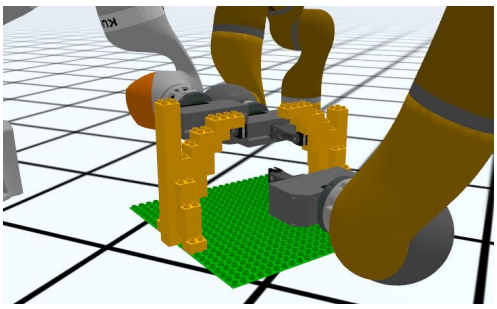
Diese implizite Kooperation ist auf den ersten Blick nicht offensichtlich. Auch wenn die drei Schritte – Übernahme des Supports auf der linken, dann auf der rechten Seite, Setzen des Verbindungssteins – nicht in jedem denkbaren Ablauf unmittelbar hintereinander ausgeführt werden müssen, so sind sie dennoch immer Voraussetzung für einen erfolgreichen Zusammenbau der Brücke. Der Planer mit der Strategie A^* konnte diese Kooperation innerhalb des Zeitlimits lediglich zweimal identifizieren und somit ein funktionierendes Programm erstellen. Der vorgestellte Planungsansatz $PaRTs$ hingegen hat diese komplexe, implizite Kooperation in allen der 250 Evaluationsläufe erfolgreich erkannt. Gerade unter dieser Prämisse ist die Planungszeit von etwa einer Minute beachtlich gering.



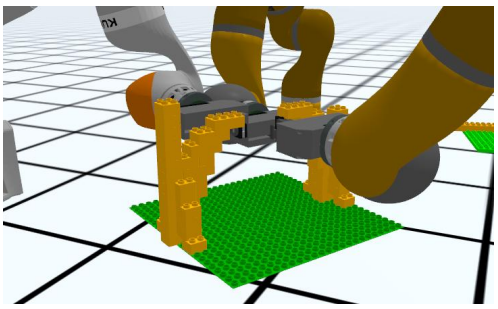
(a) Zwei Roboter R1 (vorne) und R2 (hinten) stützen die instabile Konstruktion.



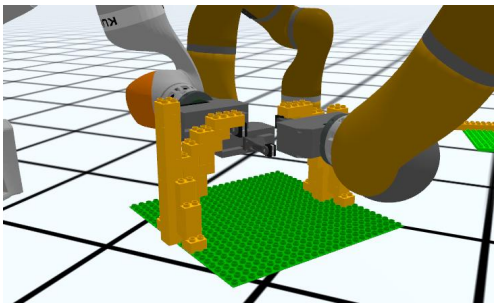
(b) Ein dritter Roboter R3 (links) übernimmt den Support auf der linken Seite.



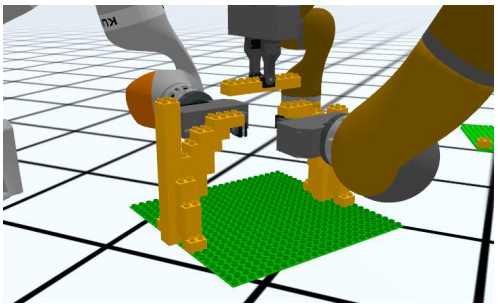
(c) Der Support von Roboter R1 ist damit nicht mehr nötig. R1 kann die Struktur loslassen …



(d) … und den Support von Roboter R2 auf der anderen Seite übernehmen.

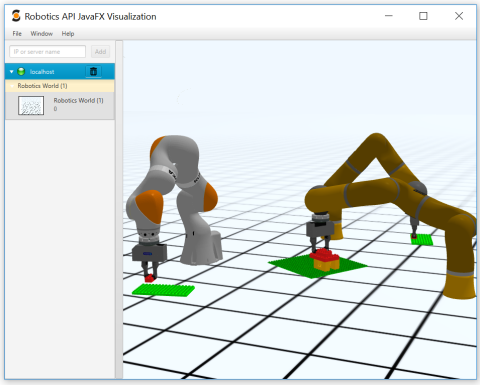


(e) Roboter R2 ist damit frei. Er kann seinen Support nun gefahrlos aufheben und …

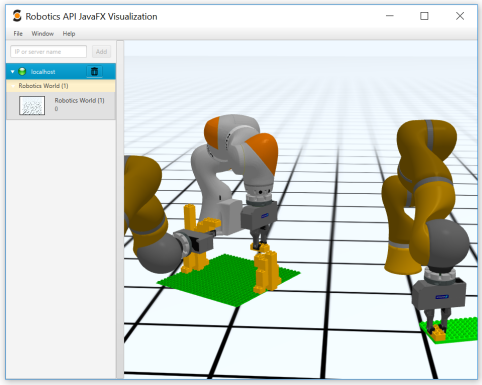


(f) … den kritischen Verbindungsstein der Brücke bringen und absetzen.

Abbildung 9.10. Ein Beispiel der impliziten Roboterkooperation, die für eine erfolgreiche Montage der Brücke vom Planer zu identifizieren ist.



(a) Simulierte Montage des LEGO-Hauses mit drei Robotern



(b) Simulierte Montage der LEGO-Brücke mit drei Robotern

Abbildung 9.11. Simulierte Ausführung von Planungsergebnissen mithilfe der Robotics API und der Visualisierungsschnittstelle.

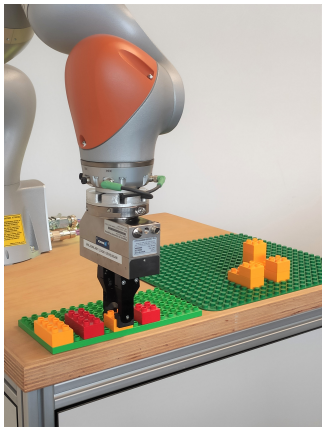
9.4.4 Simulation und reale Ausführung

Dank der eingebauten Visualisierung der Robotics API ist eine Betrachtung der simulativ ausgeführten Planungsergebnisse ohne Weiteres möglich. Abbildung 9.11 zeigt zwei während der Simulation aufgezeichnete Screenshots der Visualisierungsanwendung. In Abbildung 9.11a ist die Montage des LEGO-Hauses mit drei parallel arbeitenden Robotern zu sehen. Auch in Abbildung 9.11b ist das parallele Arbeiten zweier Roboter an einer Hälfte des Brückenbogens dargestellt, wobei der dritte Roboter die andere Brückenhälfte unterdessen stützt. Von den im Rahmen der Evaluation geplanten Ergebnissen für jedes Szenario (siehe Abschnitt 9.4.2, zwei Montageprobleme, zwei Roboterzellen und drei Planer) wurde eine Lösung simulativ ausgeführt und über die Visualisierung der Robotics API betrachtet.

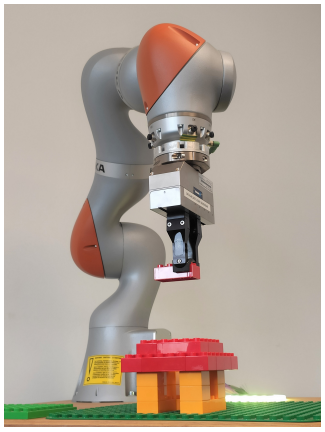
KRC

Für eine reale Ausführung der geplanten Programme kommt der RealtimeRCC der Robotics API zum Einsatz. Dieser wird auf einem Steuerungs-PC ausgeführt, auf dem ein Echtzeit-Linux mit XENOMAI [128] eingerichtet ist. Der Steuerungs-PC fungiert als zentrales Organ, an das die gesamte Peripherie über Echtzeit-Kommunikationsbusse angeschlossen ist. Die KUKA-eigene Steuerung (KUKA Robot Control, **KRC**) des LBR iiwa ist per Ethernet an den Steuerungs-PC angeschlossen. Die Kommunikation findet hier über eine zyklische Positionsvorgabe und Sensorenabfrage statt. Zwischen Greifer (SCHUNK WSG 50) und Steuerungs-PC erfolgt die Kommunikation über einen CAN-Bus. Abbildung 9.12 zeigt das Setup der realen Roboterzelle und Ausschnitte aus der Ausführung des Montageprogramms.

Roboter, LEGO-Grundplatte sowie die Platte für das Bereitstellen von Legosteinen sind auf einem Tisch angebracht und deren Position zueinander wurde nach dem in Abschnitt 9.2.2 angegebenen Verfahren eingeteacht. Das in Abschnitt 9.4.2 beschriebene Szenario (Montageplanung des LEGO-Hauses für einen einzelnen Roboter) wurde mit *PaRTs* auf dieser Roboterzelle geplant und anschließend in dieser real ausgeführt. Da die Fähigkeit zum Bereitstellen von Legosteinen keinen realen Aktuator in der Roboterzelle



(a) Aufnehmen der Legosteine von der Bereitsteller-Platte



(b) Transferbewegung und Vorpositionieren des Greifers

(c) Zielgenaues Absetzen der Steine über der Basisplatte

Abbildung 9.12. Reale Ausführung eines über den Planungsansatz $PaRTs$ erstellten Roboterprogramms zur Montage des LEGO-Hauses mit einem einzelnen Roboter.

besitzt, wurde die dafür vorgesehene Platte von Hand aufgefüllt, sobald ein Legostein für die Fertigung entnommen war. Die Ausführung der von $PaRTs$ geplanten Montage verlief erfolgreich und analog zur Ausführung über die Simulation.

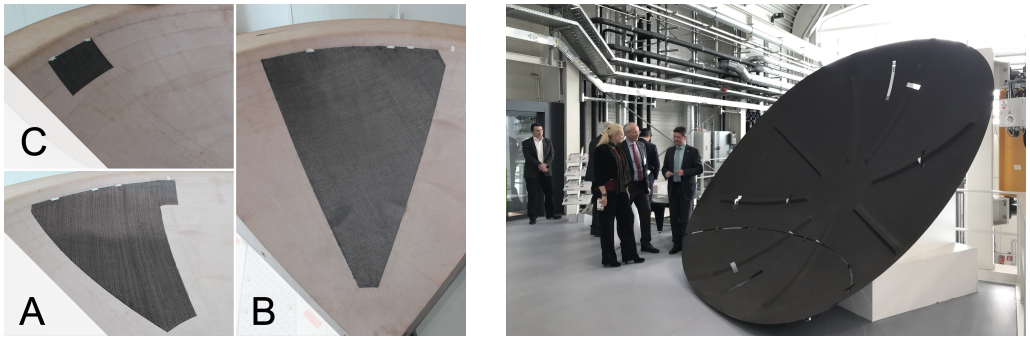
Zusammenfassung. Für eine zweite Evaluation wird der Planungsansatz $PaRTs$ auf ein Planungsproblem im Umfeld der CFK-Verarbeitung angewandt. Neben der zu fertigenden Druckkalotte werden die für die automatische Planung notwendigen Expertenmodule beschrieben. Zur Auswertung werden einerseits die Planungsergebnisse mit nur einem Endeffektor mit denen des AZIMUT-Projekts verglichen, und andererseits die Planungsergebnisse mit zwei parallel einsetzbaren Endeffektoren untersucht.

10

Evaluation der Fallstudie 2: Fertigung von Bauteilen aus carbonfaserverstärktem Kunststoff

Neben der Anwendung des Planungsansatzes $PaRTs$ auf die Erstellung von LEGO-Strukturen wird der Ansatz in Fallstudie 2 auf ein industrienäheres Aufgabenfeld bezogen: Die Fertigung von Bauteilen aus CFK.

Im Rahmen des AZIMUT-Projekts wurde mit der CFK-OPP ein Tool entwickelt, das aus den CAD-Daten des Plybooks automatisiert ein Roboterprogramm für einen einzelnen Roboter generiert. Dabei wurde in einer bottom-up-Herangehensweise ausgehend von der konkreten Problemstellung der CFK-Druckkalotte und den Spezialgreifern ein Ansatz zur Generierung von Steuerungssoftware erarbeitet, der an einigen Stellen eine Variabilität der Planung vorsieht. So funktioniert die Planung für unterschiedliche Zuschnitte gleichermaßen, und es existiert eine gemeinsame Abstraktion der drei Spezialgreifer, die ein einfaches Austauschen des zu verwendenden Greifers erlaubt.



(a) Ausgewählte Zuschnitte

(b) Fertige Druckkalotte aus CFK [26]

Abbildung 10.1. Im VARI-Prozess aus CFK-Textilien gefertigte Druckkalotte des Airbus A350, die den Passagierbereich des Flugzeugs luftdicht gegen das Heck abschließt.

Der Planungsansatz $PaRTs$ ist dagegen so konzipiert, dass er grundsätzlich allgemein und universell auf Planungsprobleme der Montage anwendbar ist und dabei den Einsatz beliebig vieler Roboter unterstützt. In einer top-down-Sicht auf die zu automatisierende Montage können Expertenmodule implementiert werden, um den Planungsansatz für die Anwendung auf eine konkrete Domäne zu spezialisieren. So tragen in dieser Fallstudie Ingenieure und Inbetriebnehmer mit ihrem für die Planung verfügbar gemachten Prozess- und Automatisierungswissen über die CFK-Domäne dazu bei, dass über den allgemeinen Planungsansatz eine konkrete Automatisierung zur Montage der CFK-Druckkalotte erstellt werden kann.

Ein Szenario dieser Fallstudie untersucht das Drapieren der drei Zuschnitte mit nur einem Endeffektor und einem Aufnahmetisch. Die entsprechenden Planungsergebnisse wurden in einer realen Roboterzelle ausgeführt und evaluiert. Im Gegensatz zur CFK-OPP ist es mit $PaRTs$ nun möglich, mehrere Roboter und Greifer auf das Planungsproblem anzuwenden und optimierte Parallelausführungen zu erhalten. In einem zweiten Szenario stehen der Planung zwei unterschiedliche Endeffektoren und zwei Aufnahmetische zur Verfügung. Die Planungsergebnisse hiervon werden über die Simulation ausgewertet.

In Abschnitte 10.1 und 10.2 wird zunächst die Druckkalotte als herzustellendes Produkt und die bei der Fertigung zum Einsatz kommende Roboterzelle beschrieben. Die Definition der Druckkalotte im Format der einheitlichen Produktbeschreibung und sämtliche Erweiterungen, die in Form von roboter- und domänenspezifischen Experten-Modulen den Planungsansatz erweitern, werden in Abschnitt 10.3 vorgestellt. Eine abschließende Evaluation in Abschnitt 10.4 zeigt die erzielten Ergebnisse bei der Anwendung des Planungsansatzes sowie bei der Ausführung der resultierenden Roboterprogramme.

10.1 Zu fertigendes Produkt: Druckkalotte aus CFK

Abbildung 10.1b zeigt eine Druckkalotte des Flugzeugs Airbus A350, die im VARI-Herstellungsverfahren produziert wurde. Der Lagenaufbau besteht aus mehreren übereinanderliegenden Lagen an Zuschnitten, die in ihrer Form variieren. Auf der innen

liegenden Seite der Druckkalotte sind zusätzliche Versteifer (die sogenannten *Stringer*) eingearbeitet, die dem Bauteil zusätzliche Stabilität verleihen. Weitere Erläuterungen zur Funktion des Bauteils im Flugzeug sind in Abschnitt 3.2.4 geben. Aus der Bauteilbeschreibung (Plybook) der Druckkalotte wurden insgesamt drei Zuschnitte A, B und C (siehe Abbildung 10.1a) ausgesucht, die sich in Form und Größe deutlich unterscheiden. Mit den drei Zuschnitten, die im fertigen Bauteil überlappen, ist ein stark reduzierter Lagenaufbau beschrieben, anhand dessen der Planungsansatz $PaRTs$ evaluiert wird.

Für die Fertigung der Druckkalotte sind pro Zuschnitt eine Reihe an Aktionen auszuführen. Bei der robotergestützten Fertigung können die notwendigen Arbeitsschritte pro Zuschnitt in sechs Einzelschritte gegliedert werden:

1. Bereitstellen des Zuschnitts auf dem Aufnahmetisch
2. Transferbewegung des Roboters zum Aufnahmetisch
3. Aufnehmen des Zuschnitts auf den Greifer
4. Transferbewegung vom Aufnahmetisch zur Viertelkalottenform
5. Einbringen der Verformung in den Zuschnitt
6. Ablegen des Zuschnitts in die Form

Für drei zu drapierende Zuschnitte ergeben sich daraus 18 auszuführende Arbeitsschritte. In Kombination mit den zur Verfügung stehenden Robotern und den vielfältigen Möglichkeiten einer konkreten Umsetzung sind hierbei eine Reihe an Freiheitsgraden für die Planung gegeben.

10.2 Aufbau der Roboterzelle

Für die Planung der Fertigungsprozesse und deren reale Ausführung wird die Technologie-Erprobungs-Zelle (**TEZ**) verwendet, die das Augsburger Zentrum für Leichtbauproduktionstechnologie des Deutschen Zentrums für Luft- und Raumfahrt e. V. (DLR-ZLP) für Forschungszwecke einsetzt. Die TEZ verfügt über eine circa 8 Meter lange Linearachse, auf der zwei Roboter unabhängig voneinander bewegt werden können. Seitlich der Linearachse befindet sich ausreichend Platz, um den Versuchsaufbau für die CFK-Fertigung zu realisieren.

TEZ

10.2.1 Objekte und technische Ressourcen

Abbildung 10.2 zeigt den Versuchsaufbau in der TEZ. Zwei in der Roboterzelle aufgestellte Tische dienen als Aufnahmestelle der Zuschnitte, die im untersuchten Szenario dort von einem Werker zur Verfügung gestellt werden. Zur Fertigung der Druckkalotte werden die einzelnen Zuschnitte in eine entsprechende Form drapiert. Statt der Originalen Form (siehe Abbildung 10.3a), die einen Durchmesser von $3{,}5$ bis 4 Meter besitzt und überwiegend von hängend montierten Robotern aufgrund deren Erreichbarkeit vollständig bedient werden kann, ist für die Versuchsreihe in der TEZ lediglich eine Teilform ausreichend. Die in dieser Fallstudie zu drapierenden Zuschnitte befinden sich in einem gemeinsamen Teilbereich der späteren Druckkalotte, sodass die sogenannte Viertelkalottenform (siehe Abbildung 10.3b), die diesen gemeinsam genutzten Ausschnitt

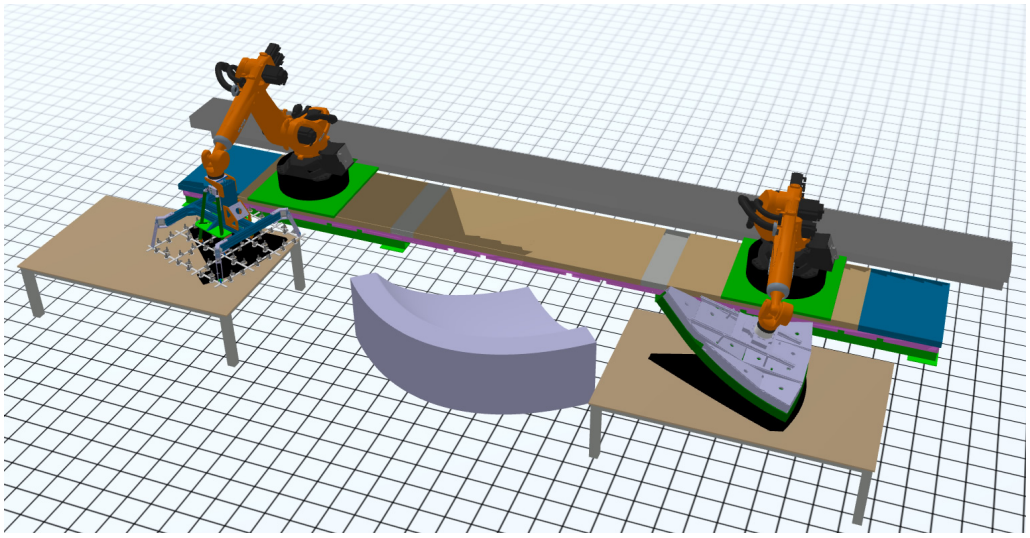


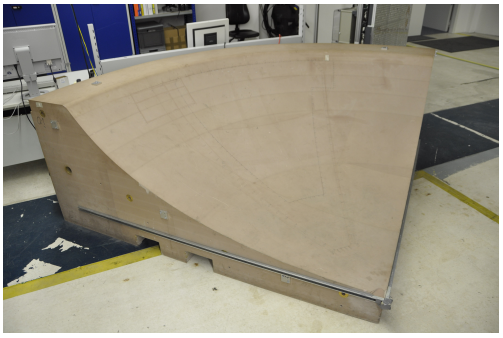
Abbildung 10.2. Computermodell des Versuchsaufbaus in der TEZ.

aus der Kalottenform darstellt, als Drapierform dienen kann. Die Viertelkalottenform ist in der TEZ so zwischen den beiden Aufnahmetischen positioniert, dass sie von den Robotern gut zu erreichen ist. Die TEZ besteht aus zwei KUKA-Robotern der Modellreihe KR QUANTEC ULTRA (KR 210 R3100 ultra), die eine Reichweite von mehr als 4 Metern besitzen und somit auch mit größeren Endeffektoren die verschiedenen Bereiche der Roboterzelle erreichen. Beide Roboter sind jeweils auf einem kleinen Sockel auf einer gemeinsamen Linearachse montiert, die eine unabhängige Bewegung beider Roboter entlang dieser Achse erlaubt.

Weiterhin sind die drei Spezialgreifer des AZIMUT-Projekts, wie in Abbildungen 10.3c bis 10.3e dargestellt, als virtuelle und simulierbare Geräte für die automatische Planung modelliert. Für das Szenario der Fallstudie, das die Auswertung der Drapierung mit einem einzelnen Roboter betrachtet, wird der Schaumstoffgreifer des Fraunhofer IWU-RMV eingesetzt, der an den Flansch des rechts in der TEZ befindlichen Roboters montiert ist. Im zweiten Szenario wird die Roboterzelle um den anderen Roboter der TEZ ergänzt, der mit dem Netzgreifer des DLR als Endeffektor ausgestattet ist.

Beide Greifer haben mehrere Einheiten, über die sie die Verformung in den gegriffenen Zuschnitt einbringen. Beim Schaumstoffgreifer sind das die beiden anklappbaren Seitenflügel, beim Netzgreifer die einzelnen Coanda-Saugmodule, die am Rasternetz angebracht sind. Bei beiden Greifern werden diese Einheiten über ein- und denselben Wertgeber angesteuert, sodass der Verformungsgrad über eine einheitliche Fließkommazahl ausgedrückt werden kann. Besonders beim Netzgreifer ist es von Vorteil, die exakte Position jedes einzelnen Coanda-Saugmoduls zu einem gegebenen Verformungsgrad zu kennen. Über eine Simulation kann während der Planung etwa analysiert werden, wie gut die Abdeckung der Saugmodule über dem Zuschnitt ist oder wie stark einzelne Saugmodule bei der Drapierung in die Form eindrücken. Zur Berechnung der Saugmodulpositionen in Abhängigkeit des Verformungsgrads stützt sich die Planung

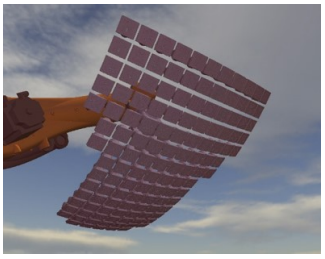
(a) Originalform für die Druckkalotte



(b) Viertelform für die Druckkalotte

(c) Netzgreifer

(d) Schaumstoffgreifer

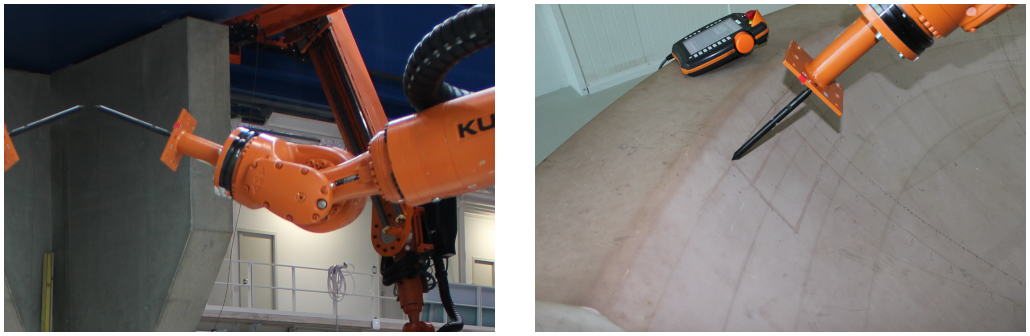


(e) Flächengreifer

Abbildung 10.3. CFK-spezifische Objekte der Roboterzelle, die in digitaler Form für die automatische Fertigungsplanung von Bauteilen aus CFK erforderlich sind.

auf Messdaten, die das Gitternetz des realen Roboters räumlich beschreiben. Zu insgesamt vier unterschiedlichen Verformungsstufen (1: flach, 2: leicht, 3: mittel, 4: maximal) existieren solche Messdaten, sodass für einen dazwischenliegenden Verformungswert eine räumliche Interpolation stattfinden kann. Für ein am Netz montiertes Saugmodul werden jeweils die zwei nächstgelegenen Punkte des Netzes identifiziert, für die solche Messdaten vorliegen, und ebenfalls für eine Interpolation zur Annäherung an die tatsächliche Saugmodulposition herangezogen. Damit kann zu einem beliebig eingestellten Verformungswert die Position eines jeden Saugmoduls berechnet werden.

Aufgrund seiner starren Mittelkonstruktion ist beim Schaumstoffgreifer keine gesonderte Positionsberechnung von Flächenteilen notwendig. Das digitale Modell dieses Greifers ist deutlich einfacher, da auf dem Mittelstreifen des Greifers keine direkte Verformung stattfindet. Dafür erfordert die Greifergeometrie eine komplexe Aufnahmebewegung, die einen Zuschnitt rollend und mit entsprechend nacheinander geschalteten Vakuumkammern von einer ebenen Fläche aufnimmt. Jeweils ein verfahrbarer Schlitten auf der Lineareinheit, der darauf montierte Roboter und der Greifer an dessen Flansch bilden zusammen eine technische Ressource aus Sicht des Planungsansatzes. Dies hat zur Folge, dass unabhängige und parallel ausgeführte Aktionen der drei Aktuatoren nicht möglich sind. Da die Aktuatoren eine kinematische Kette beschreiben, hätte eine beliebig parallele Ausführung z. B. einer Roboterbewegung und einer Verschiebung der Lineareinheit einen unbekannten geometrischen Ablauf zufolge. Daher ist dieses Verhalten explizit nicht erwünscht. Stattdessen können abgestimmte kombinierte Bewegungen



(a) Werkzeugvermessung (Messspitze) **(b)** Objektvermessung

Abbildung 10.4. Zur digitalen Erfassung der Roboterzelle werden sämtliche für die Planung relevanten Objekte mit einem eingelernten Werkzeug (Messspitze) vermessen.

der drei Aktuatoren in Form eines technischen Tasks erfolgen. In Abbildung 10.2 trägt der rechts stehende Roboter mit Schlitten und Schaumstoffgreifer den Namen *Roboter 1*, der links stehende Roboter mit Netzgreifer wird *Roboter 2* genannt.

10.2.2 Einmessen der realen Roboterzelle

Für eine realistische Planung und eine spätere hoch präzise Ausführbarkeit der Planungsprogramme ist eine möglichst wirklichkeitsgetreue digitale Nachbildung der Roboterzelle erforderlich. Um die Objekte der Roboterzelle räumlich zu vermessen, wird eine an die Roboter der TEZ montierte Messspitze über die XYZ 4-Punkt-Methode zur Werkzeugvermessung gemäß Abschnitt 8.2.1 eingelernt (siehe Abbildung 10.4a). Anschließend werden mit der Messspitze am Roboter die Oberflächen der beiden Aufnahmetische über die 3-Punkt-Methode vermessen. Zur Bestimmung der Position der Viertelkalottenform werden markante Punkte auf ihrer Oberfläche angefahren, deren räumliche Beziehung aus den CAD-Daten am Computer auslesbar sind. Mit ihrer relativen Position zueinander und den Messdaten des Roboters kann daraus die Position der Form in der Roboterzelle abgeleitet werden.

Die beiden in dieser Fallstudie eingesetzten Greifer liegen als digitales Modell vor. Eine Bestimmung ihrer absoluten Position in der Roboterzelle ist selbstverständlich nicht notwendig, da sie unmittelbar an den Flansch des Roboters montiert sind. Wie exakt die teilweise recht weit vom Flansch entfernten Teile des Greifers mit den Angaben im Modell übereinstimmen, kann für die Drapiergenauigkeit bei der Fertigung entscheidend sein. Der Schaumstoffgreifer verfügt dank des Schaumstoffs über eine gewisse Fehlertoleranz. Dennoch wurden einige Eck- und Zuschnittsaufnahmepunkte auf dem Schaumstoff mit der Messspitze am anderen Roboter der TEZ angefahren und überprüft. Für die Positionen der Saugmodule am Netzgreifer existieren – wie im vorherigen Abschnitt beschrieben – spezielle Messdaten, die seitens des DLR über ein optisches Lokalisierungsverfahren ermittelt wurden. Somit ist auch für die Greifer eine hohe Wirklichkeitstreue gegeben.

10.3 Erweiterungen zum modularen Ansatz

Für die automatische Fertigungsplanung der CFK-Druckkalotte über den modularen Planungsansatz $PaRTs$ sind von den entsprechenden Experten einige Erweiterungen zur Verfügung zu stellen. Anders als in der Fallstudie mit LEGO ist für die Fertigung der Druckkalotte eine Reihenfolge der Einzelteile bereits über das Plybook vorgegeben, ebenso wie deren Zielposition in der Form. Diese Informationen werden als planungsrelevante Prozessparameter explizit in die einheitliche Produktbeschreibung der Druckkalotte übernommen, und es müssen dafür keine Expertenmodule gesondert zur Verfügung gestellt werden. Dies betrifft:

Analysemodule: Die Zielposition der einzelnen Zuschnitte in der Form ist über das Plybook eindeutig festgelegt. In der einheitlichen Produktbeschreibung wird diese Information in Form eines herzustellenden Attributs des Zuschnitts bereitgestellt. Damit sind die herzustellenden Attribute der Druckkalotte bereits bekannt und es ist keine weitere Analyse zu deren Erkennung erforderlich.

Domänentaskmodul: Über dieses Modul werden zu einem gegebenen Planungszustand gültige Nachfolge-Domänentasks bestimmt. In der CFK-Fallstudie ist die Reihenfolge der zu drapierenden Zuschnitte vollständig vorgegeben, sodass sich hieraus bereits eine eindeutige Abfolge an Domänentasks ergibt.

Domänenprüfmodule: Ungültige Konstellationen des Bauteils während der Planung können über Domänenprüfmodule herausgefiltert werden. Die im Plybook vorgegebene Reihenfolge der zu drapierenden Zuschnitte stellt aus Domänensicht einen gültigen Ablauf dar; daher bedarf es keiner weiteren Untersuchung durch Domänenprüfmodule.

Validatormodule: Produktsituationen, die sich erst in gemeinsamer Betrachtung mit den technischen Ressourcen als ungültig herausstellen, können über Validatormodule herausgefiltert werden. Damit ist z. B. eine Kollisionsprüfung zwischen Bauteil und Roboter realisierbar, wobei die flachen Zuschnitte, die etwa in der Form oder auf dem Aufnahmetisch liegen, tatsächlich keine nennenswerte Veränderung der dynamischen Kollisionsstrukturen darstellen. Statische Kollisionsobjekte hingegen werden unmittelbar von den Fähigkeitenmodulen berücksichtigt. Bei der LEGO-Montage wird über Validatormodule außerdem die Statik der Bauteilstruktur überprüft, die durch Roboter zusätzlich gestützt sein kann. Für die Fertigung der Druckkalotte aus CFK existiert keine dazu vergleichbare zu überprüfende Eigenschaft.

Die für die Planung in der CFK-Domäne relevanten Attribute werden in Abschnitt 10.3.1 eingeführt. Abschnitt 10.3.2 beschreibt anschließend die Modellierung der Druckkalotte über die einheitliche Produktbeschreibung sowie die Einbindung der Zuschnittsreihenfolge und die herzustellenden Attribute als planungsrelevante Prozessparameter. Die für die CFK-Fertigung zur Verfügung stehenden Fähigkeitenmodule sind in Abschnitt 10.3.3 erläutert. Da die Evaluation der Planungsergebnisse am realen Roboter über dessen eigenen Programminterpreter stattfand, werden die entstandenen technischen Tasks über einen eigens entwickelten Code-Generator in die Programmiersprache der Ausführungsplattform übersetzt. Abschnitt 10.3.4 beschreibt die Funktionsweise des Code-Generators für die proprietäre Programmiersprache KRL.

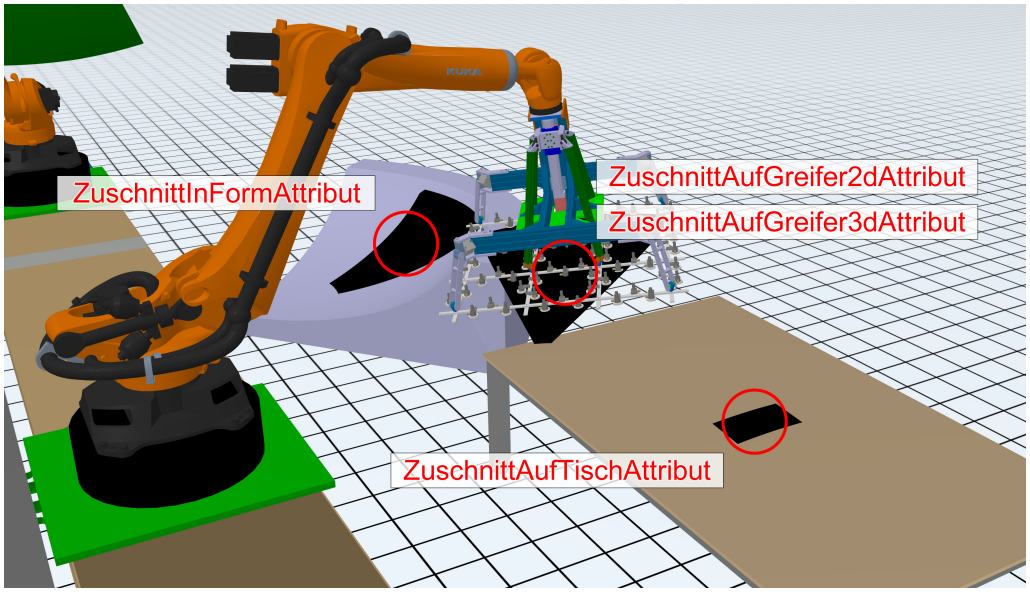


Abbildung 10.5. Für die Fertigung der Druckkalotte dienen vier CFK-spezifische Attribut-Typen als Beschreibung geometrischer und semantischer Zusammenhänge.

10.3.1 CFK-spezifische Attribut-Typen

Für die automatische Planung zur Herstellung der CFK-Druckkalotte existieren vier CFK-spezifische Attribut-Typen (siehe Abbildung 10.5), um die der Planungsansatz für diese Fallstudie erweitert wird:

ZuschnittAufTischAttribut: Wird ein Zuschnitt manuell von einem Werker oder automatisiert über ein Zubringersystem auf dem Aufnahmetisch bereitgestellt, so drückt ein ZuschnittAufTischAttribut eine geometrische Position des Zuschnitts auf dem Tisch aus. Zudem ist implizit die semantische Eigenschaft gegeben, dass der betreffende Zuschnitt vom Aufnahmetisch in planarer Form aufgegriffen werden kann.

ZuschnittAufGreifer2dAttribut: Wird ein Zuschnitt von einem Greifer aufgenommen, so wird zwischen Greifer und Zuschnitt ein ZuschnittAufGreifer2dAttribut unter Angabe des geometrischen Offsets hergestellt. Das ZuschnittAufGreifer2dAttribut gibt zusätzlich an, dass sich der Zuschnitt in einem nicht verformten, planaren Zustand befindet. Dies wird vereinfachend auch für den Schaumstoffgreifer angenommen, der eine gekrümmte Aufnahmefläche besitzt.

ZuschnittAufGreifer3dAttribut: Verformt sich der Greifer und bringt demnach eine Verformung in den gegriffenen Zuschnitt ein, so wird das ZuschnittAufGreifer2dAttribut durch ein ZuschnittAufGreifer3dAttribut ersetzt. Das Attribut gibt nun an, dass sich der Zuschnitt in einem verformten, der Zielform angepassten Zustand befindet. Außerdem kann der Verformprozess Einfluss auf die Position des Zuschnitts auf dem Greifer genommen haben, die jetzt im ZuschnittAufGreifer3dAttribut hinterlegt ist.

ZuschnittInFormAttribut: Liegt ein Zuschnitt in drapierter Form in der Kalottenform, so existiert zwischen ihm und der Kalottenform ein ZuschnittInFormAttribut. Dieses

gibt einerseits ein geometrisches Offset zwischen beiden an, andererseits drückt es aus, dass sich der Zuschnitt im verformten, der Zielform angepassten Zustand befindet.

Für eine vollständige Modellierung eines Fertigungszustands in einer Planungssituation werden für diese Fallstudie zusätzliche drei Attribut-Typen eingesetzt (siehe Abschnitt 4.1.2): Über Positions-Attribute sind beide Aufnahmetische sowie die Lineareinheit, auf denen die zwei Roboter angebracht sind, in der Roboterzelle platziert. Der eingestellte Verformungsgrad des Greifers sowie die Roboterstellung inklusive der Position des Roboters auf der Lineareinheit werden zu jedem Zeitpunkt der Planung in einem Konfigurations-Attribut festgehalten. Für die Beschreibung des zweiten Roboters existiert analog ein zweites Konfigurations-Attribut. Die Viertelkalottenform ist in der Mitte der Roboterzelle über ein Fixpunkt-Attribut positioniert.

Für die Unterscheidung, welche der Attribute für die Prozess- und welche für die Fertigungsplanung relevant sind, ist eine eindeutige Zuordnung zu den Typen Produkt- und Aktuatorattribut gegeben: Sowohl alle Zuschnitte wie auch die Viertelkalottenform gehören zu den Einzelteilen, die am Ende zu einem fertigen Produkt verarbeitet werden. Die Form wird deshalb als Einzelteil des Produkts angesehen, da sie im Anschluss an die Zuschnittsdrapierungen mitsamt dem Lagenaufbau in die Weiterverarbeitung geht und nicht in der Roboterzelle verbleibt. Damit stellen ZuschnittInFormAttribute und das FixpunktAttribut Produktattribute für die Prozessplanung dar. Alle übrigen Attribute (ZuschnittAufTischAttribute, ZuschnittAufGreifer2dAttribute, ZuschnittAufGreifer3dAttribute, Positions- und Konfigurations-Attribute) beschreiben den Zustand der Roboterzelle während der Fertigung und sind deshalb vom Typ Aktuatorattribut.

10.3.2 Modellierung über die einheitliche Produktbeschreibung

Grundlage für die Fertigung der CFK-Druckkalotte ist das Plybook des Ingenieurs, das den Lagenaufbau in der Kalottenform über eine XML-Syntax beschreibt. Zur automatischen Planung der robotergestützten Fertigung über $PaRTs$ wird das Plybook in das Format der einheitlichen Produktbeschreibung überführt. Dafür wurde eigens ein Generator entwickelt, der die im Plybook enthaltenen Zuschnitte in Einzelteilbeschreibungen übersetzt und diese anhand der Struktur des Plybooks, die den Lagenaufbau der Druckkalotte wiedergibt, eins zu eins zusammen mit der Kalottenform in einer Baugruppenbeschreibung zusammenfügt (siehe Listing 10.1). Als planungsrelevanter Prozessparameter (siehe Abbildung 5.4 in Abschnitt 5.1.2) ist in der Baugruppenbeschreibung für jeden Zuschnitt u. a. ein ZuschnittInFormAttribut als herzustellendes Attribut benannt. Dieses referenziert den Zuschnitt und die Form als Bezugselemente und gibt einen *Offset* an, der die Zielposition des Zuschnitts in der Form definiert.

Die in der Baugruppenbeschreibung zusätzlich angegebenen *Tasks* werden bei der Prozessplanung als Domänentasks berücksichtigt. Jeder Task referenziert als *Effekt* der Ausführung exakt ein herzustellendes ZuschnittInFormAttribut, nämlich das Attribut desjenigen Zuschnitts, dessen Drapierung über den Task ausgedrückt wird. Die Tasks *task_b* und *task_c* definieren zusätzlich einen Vorgänger-Task, sodass eine eindeutige Reihenfolge $A \rightarrow B \rightarrow C$ zur Berücksichtigung bei der Prozessplanung gegeben ist.

```xml
<Baugruppenbeschreibung id="druckkalotte" params="">

  <!-- Viertelkalottenform -->
  <Unterelement id="form" ref="viertelkalottenform" params="" />

  <!-- Zuschnitt A, dessen Attribut und Task -->
  <Unterelement id="zuschnitt_a" ref="zuschnitt_a" params="" />

  <Attribut id="attr_a" typ="ZuschnittInFormAttribut" offset="...">
    <Beziehung unterelement="form" />
    <Beziehung unterelement="zuschnitt_a" />
  </Attribut>

  <Task id="task_a" prozess="drapieren">
    <Effekt attribut="attr_a" />
  </Task>

  <!-- Zuschnitt B, dessen Attribut und Task -->
  <Unterelement id="zuschnitt_b" ref="zuschnitt_b" params="" />

  <Attribut id="attr_b" typ="ZuschnittInFormAttribut" offset="...">
    <Beziehung unterelement="form" />
    <Beziehung unterelement="zuschnitt_b" />
  </Attribut>

  <Task id="task_b" prozess="drapieren">
    <Effekt attribut="attr_b" />
    <Nach task="task_a" />
  </Task>

  <!-- Zuschnitt C, dessen Attribut und Task -->
  <Unterelement id="zuschnitt_c" ref="zuschnitt_c" params="" />

  <Attribut id="attr_c" typ="ZuschnittInFormAttribut" offset="...">
    <Beziehung unterelement="form" />
    <Beziehung unterelement="zuschnitt_c" />
  </Attribut>

  <Task id="task_c" prozess="drapieren">
    <Effekt attribut="attr_c" />
    <Nach task="task_b" />
  </Task>

  <!-- Weitere Zuschnitte -->
  ...

</Baugruppenbeschreibung>
```

Listing 10.1. Ausschnitt aus der Baugruppenbeschreibung im XML-Format für die Druckkalotte aus CFK. Für die Fallstudie sind die Zuschnitte A bis C relevant.

10.3.3 Bereitgestellte Fähigkeitenmodule

Für eine automatisierte Fertigung der CFK-Druckkalotte beschreiben Fähigkeitenmodule die teilweise komplexen Prozesse der Roboter und Greifer. Dabei sind die sechs in Abschnitt 10.1 genannten Arbeitsschritte pro Zuschnitt notwendig, um diesen in die Form zu drapieren. Beim Herstellen des Lagenaufbaus ist diese Abfolge für jeden Zuschnitt immer einzuhalten. Liegen z. B. jeweils zwei Zuschnitte so weit in der Form auseinander, dass sie mit den Robotern parallel drapiert werden können, so dürfen die Ausführungen beider Arbeitsschritt-Abfolgen allerdings überlappen (interleaven). Zur Konkretisierung der sechs Arbeitsschritte im Rahmen der Planung werden nachfolgende drei Fähigkeitenmodule zur Verfügung gestellt. Verändert ein vom Fähigkeitenmodul erzeugter Task die Einstellung eines Aktuators, so wird ein vorher geltendes Konfigurations-Attribut $a_{konf} \in \mathcal{A}$ für Roboter 1 oder Roboter 2 in ein nachher geltendes Konfigurations-Attribut $a'_{konf} \in \mathcal{A}$ überführt.

Fähigkeitenmodul „Bereitstellen“: Für Arbeitsschritt 1, in dem ein Zuschnitt auf dem Aufnahmetisch bereitgestellt wird, berechnet das Fähigkeitenmodul „Bereitstellen“ einen entsprechenden technischen Task. Dieser stellt mit seiner Ausführung ein ZuschnittAufTischAttribut $a_{tisch} \in \mathcal{A}$ zwischen dem Tisch als technische Ressource und dem Zuschnitt als Einzelteil her. Da der Aufnahmetisch keine sich bewegende Aktuatorik besitzt, ist der durch den Task beschriebene Situationsübergang vollständig durch Gleichung 10.1 beschrieben:

$$\begin{aligned} \mathit{Effekt}(t, s) &= (\{a_{tisch}\}, \{\}) \\ nach(t, s) &= s \cup \{a_{tisch}\} \end{aligned} \tag{10.1}$$

In der Simulation führt ein derartiger Task außer der Herstellung des Attributs keine tatsächliche Aktion aus. Bei der realen Ausführung dieser Fallstudie erfolgt das Bereitstellen des Zuschnitts auf dem Aufnahmetisch manuell durch einen Werker. Allerdings darf hierbei der Roboter nicht mit dem Aufnahmevorgang starten, solange der Werker noch mit dem Auslegen des Zuschnitts beschäftigt ist. Bei der realen Ausführung wartet der Bereitstell-Task daher vor seiner Beendigung auf ein Input-Signal, das über einen an die Steuerung angeschlossenen Knopf gegeben wird. Damit kann der Werker bestätigen, dass er mit der Auslage des Zuschnitts fertig ist. Aufgrund der Ausführungssemantik der technischen Tasks warten die nachfolgenden, abhängigen Tasks automatisch solange mit ihrer Ausführung.

Für das Anbieten des Zuschnitts benötigt das Fähigkeitenmodul „Bereitstellen“ die exakte Position des Zuschnitts auf dem Aufnahmetisch. Hier ist eine innerhalb des Aufnahme-Tasks ausgeführte automatische Erkennung – etwa durch eine am Greifer montierte Kamera – denkbar. Für diese Fallstudie wurde jedoch eine interaktive Variante während der Planung gewählt. So fordert das Fähigkeitenmodul diese Information über eine externe Schnittstelle an, die daraufhin einen grafischen Dialog, wie in Abbildung 10.6 dargestellt, anzeigt. Auf diesem ist die Tischoberfläche und die Kontur des Zuschnitts dargestellt, die der Experte mit der Computermaus nun verschieben und um beliebige Punkte drehen kann. Dabei werden spitze Winkel der Zuschnittskontur als rote Punkte dargestellt. Zu einelnen Eckpunkten des Zuschnitts kann der Werker

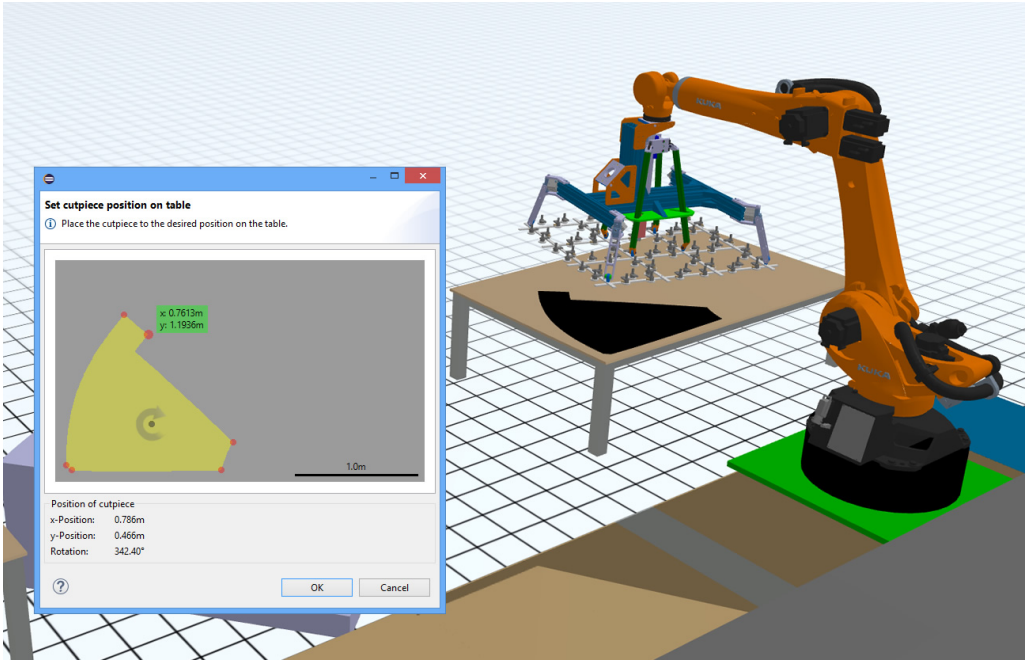


Abbildung 10.6. Interaktive Festlegung der Zuschnitts-Position auf dem Aufnahmetisch.

exakte Positionswerte ablesen, die er bei der späteren Zuschnittsauslegung während der realen Fertigung berücksichtigen kann.

Das Fähigkeitenmodul „Bereitstellen“ ist in zweifacher Instanz den beiden Aufnahmetischen als jeweils eigene Fähigkeit zugewiesen.

Fähigkeitenmodul „Aufnehmen“: Für das Greifen eines Zuschnitts auf dem Aufnahmetisch stellt das Fähigkeitenmodul „Aufnehmen“ einen ausführbaren Task zur Verfügung. Dieser beinhaltet auch das Anfahren einer geeigneten Vorposition durch den Roboter und setzt damit die Arbeitsschritte 2 und 3 um. Die Ausführung des Tasks ändert einerseits die Konfiguration des Roboters, andererseits wird das bestehende ZuschnittAufTischAttribut $a_{tisch} \in \mathcal{A}$ durch ein ZuschnittAufGreifer2dAttribut $a_{greif2d} \in \mathcal{A}$ ersetzt. Folglich gilt der in Gleichung 10.2 angegebene Situationsübergang:

$$\begin{aligned} \mathit{Effekt}(t,s) &= \left(\{a'_{konf}, a_{greif2d}\}, \{a_{konf}, a_{tisch}\}\right) \\ \mathit{nach}(t,s) &= (s \setminus \{a_{konf}, a_{tisch}\}) \cup \{a'_{konf}, a_{greif2d}\} \end{aligned} \tag{10.2}$$

Die Position des Zuschnitts auf dem Aufnahmetisch wird bei der Erstellung des Aufnahme-Tasks dem abzubauenden ZuschnittAufTischAttribut entnommen. Die Entscheidung, wie der Greifer den Zuschnitt aufnimmt und an welcher Stelle sich dieser auf der Greiffläche anschließend befindet, stellt einen Freiheitsgrad bei der Planung dar. Tatsächlich gibt es viele Möglichkeiten, wie die Position und auch die Drehung des Zuschnitts gewählt werden kann. Allerdings stellt sich bei der Drapierung heraus, ob die Oberfläche des verformten Greifers zu der komplexen Krümmung der Viertelkallotenform an der Stelle passt, an der der Zuschnitt zielgenau abzulegen ist. In der Tat

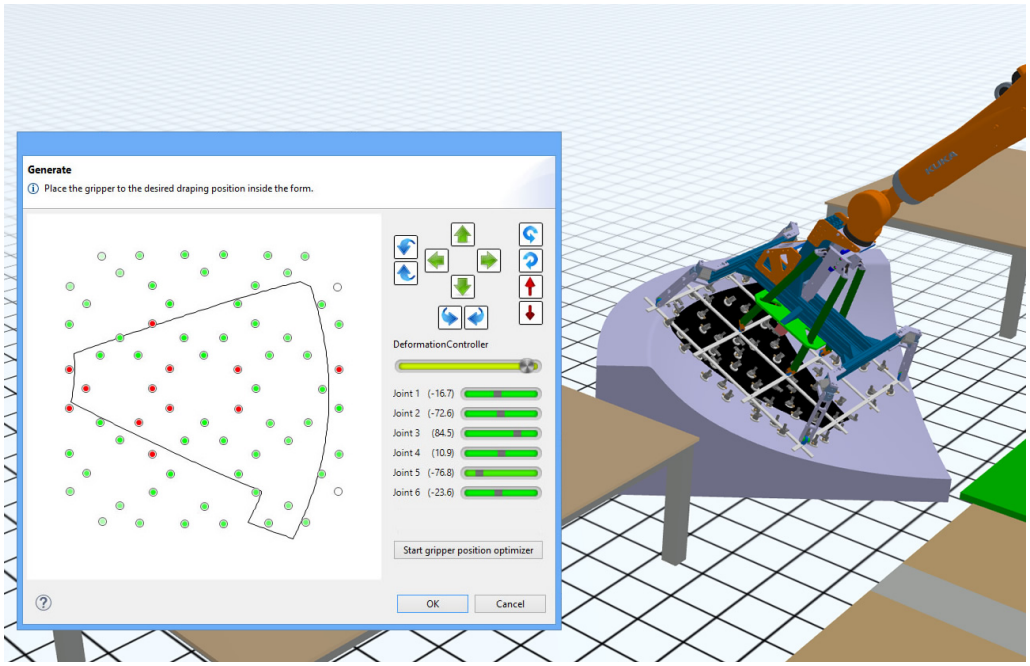


Abbildung 10.7. Interaktive Bestimmung der optimalen Zuschnitts-Position auf dem Greifer für eine passgenaue Drapierung des Zuschnitts in der Form durch die Greifergeometrie.

fallen die Freiheitsgrade bei der Wahl der Zuschnittsposition auf dem Greifer im Fall der dreidimensionalen Drapierung deutlich geringer aus als bei der ebenen Zuschnittsaufnahme über dem Tisch. Daher verfolgt das Fähigkeitenmodul „Aufnehmen“ bei der Task-Erzeugung einen rückwärtsorientierten Ansatz. Demnach soll ein Experte die geeignete Position des Greifers bei der Drapierung des Zuschnitts in der Form bestimmen, damit daraus die Position des Zuschnitts auf dem Greifer und somit der Aufnahme-Task berechnet werden kann. Analog zum Fähigkeitenmodul „Bereitstellen“ greift auch das Fähigkeitenmodul „Aufnehmen“ dafür auf eine externe Schnittstelle zurück, die dem Experten einen grafischen Eingabedialog anzeigt. Gleichzeitig wird in der Visualisierung der Roboterzelle der betreffende Zuschnitt an seiner Zielposition in der Form dargestellt und der Roboter nimmt eine vorgeschlagene Stellung ein, die den Greifer in einer ungefähr passenden Position über dem Zuschnitt positioniert. Abbildung 10.7 zeigt ein solches Szenario in der Visualisierung sowie den Dialog, der dem Experten präsentiert wird. Im rechten, oberen Teil des Dialogs befinden sich Knöpfe, über die der Experte den Greifer intuitiv bewegen (*handverfahren*) kann. Die kartesischen Bewegungen finden im Koordinatensystem des Greifers statt, sodass eine Umorientierung stets die Position des Greifers in der Mitte der Greiffläche hält. Die dafür notwendigen Bewegungen werden vom Roboter simultan ausgeführt. Die sechs länglichen, grünen Anzeigefelder teilen dem Experten mit, wie weit der Roboter in seinen Einzelachsen von den Achsgrenzen entfernt ist. Dies ist oft hilfreich für den Experten, zu erkennen, warum sich der Roboter ggf. nicht mehr kartesisch in eine bestimmte Richtung bewegen lässt. Unterhalb der Knöpfe für das Handverfahren befindet sich ein Schieberegler, über

den der Verformungsgrad des Greifers passend gewählt werden kann. Auf der linken Seite des Dialogs sind über Kreise die einzelnen Greifpunkte des Greifers – in diesem Fall die Coanda-Saugmodule des Netzgreifers – dargestellt. Zusätzlich ist die Kontur des Zuschnitts aus der derzeitigen Sicht des Greifers abgebildet, die eine Analyse seiner Überdeckung mit den Greifpunkten zulässt. Die Einfärbung der Kreise gibt an, wie weit ein Greifpunkt von der Viertelkalottenform entfernt ist. Während ein grüner Kreis einen optimalen Abstand angibt, bedeutet die Farbe weiß einen zu großen Abstand und rot ein Eindrücken des Greifpunkts in die Form. Setzt der Experte die Knöpfe für das Jogging ein, so verschiebt sich die Zuschnittskontur im Dialog gegengleich zum Greifer in der Visualisierung. Das Jogging kann – über den Dialog betrachtet – somit als intuitives Verschieben und Drehen des Zuschnitts genutzt werden.

Über einen Knopf im unteren, rechten Bereich des Dialogs kann der Experte ein automatisches Finden einer geeigneten Position anstoßen. Dazu wird im Hintergrund ein Partikelschwarmoptimierer gestartet, der verschiedene Greiferpositionen und -verformungen ausprobiert und dabei nach einer bestmöglichen Greifpunktüberdeckung und optimalen Greifpunktabständen zur Form sucht. Abbildung 10.7 zeigt diese Daten für den Netzgreifer. Für den Schaumstoffgreifer sind einige solcher Greifpunkte auf der Schaumstoffoberfläche definiert, sodass auch für ihn die grafische Bestimmung der Zuschnittsposition angewandt werden kann. Für ihn ergibt sich der zusätzliche Vorteil, dass der Dialog ein „Durchschauen“ durch den Greifer auf die Zuschnittskontur ermöglicht, wohingegen in der Visualisierung der Greifer den Zuschnitt verdeckt.

Nach der Bestätigung durch den Experten (Schließen des Dialogs) ist die Position des Zuschnitts auf dem Greifer – zumindest im verformten Zustand – über die Berechnung der Inverskinematik des Roboters bekannt. Um daraus die Zuschnittsposition im unverformten Zustand abzuleiten, wird die doppelt gekrümmte Fläche des drapierten Zuschnitts auf die Fläche des unverformten Greifers projiziert. Die Projektion wird als Grundlage für die anschließende Erstellung des Aufnahme-Tasks herangezogen. Da sich die beiden in der Fallstudie eingesetzten Greifer in ihren Aufnahmebewegungen unterscheiden, existiert das Fähigkeitenmodul „Aufnehmen“ in zwei Varianten: Das Fähigkeitenmodul für den Netzgreifer erzeugt für die Aufnahme eine senkrecht zum Aufnahmetisch erfolgende „Stempel“-Bewegung mit anschließender Aktivierung der Saugmodule. Für den Schaumstoffgreifer wird aufgrund seiner Geometrie eine rollende Aufnahmebewegung erzeugt, die zu entsprechenden Zeitpunkten die einzelnen Vakuumkammern aktiviert.

Fähigkeitenmodul „Drapieren“: Für die verbleibenden drei Arbeitsschritte 4 bis 6 (Transferbewegung zur Form, Verformung des Zuschnitts und Ablage in der Form) erstellt das Fähigkeitenmodul „Drapieren“ einen entsprechenden technischen Task. Dabei wird die Konfiguration des Roboters $a_{konf} \in \mathcal{A}$ in mehreren Schritten in eine resultierende Konfiguration $a'_{konf} \in \mathcal{A}$ überführt. Das zuvor bestehende ZuschnittAufGreifer2dAttribut $a_{greif2d} \in \mathcal{A}$ wird mit der Umformung des Greifers, und damit der Einbringung der Verformung in den Zuschnitt, zunächst in ein ZuschnittAufGreifer3dAttribut $a_{greif3d} \in \mathcal{A}$ umgewandelt. Über das anschließende Ablegen des Zuschnitts in der Form wird das ZuschnittAufGreifer3dAttribut wieder abgebaut und ein ZuschnittInFormAttribut $a_{form} \in \mathcal{A}$ hergestellt. Die insgesamt stattfindende Situationsänderung ist in Gleichung 10.3 beschrieben, wobei $a_{greif3d}$ als innerhalb des Tasks auf- und

abzubauendes Attribut für den übergeordneten Drapier-Task nicht erscheint:

$$\begin{aligned} \mathit{Effekt}(t,s) &= \left(\{a'_{konf}, a_{form}\}, \{a_{konf}, a_{greif2d}\}\right) \\ nach(t,s) &= (s \setminus \{a_{konf}, a_{greif2d}\}) \cup \{a'_{konf}, a_{form}\} \end{aligned} \tag{10.3}$$

Da die Informationen zur Drapierung des Zuschnitts aufgrund des rückwärtsorientierten Ansatzes bereits über das Fähigkeitenmodul „Aufnehmen“ ermittelt wurden und diese im hergestellten ZuschnittAufGreifer2dAttribut für die nachfolgende Planung gespeichert sind, ist vom Fähigkeitenmodul „Drapieren“ lediglich noch die Erzeugung des entsprechenden Drapier-Tasks zu erfüllen. Auch dieses Modul wird jedem Greifer in einer eigenen Variante zugewiesen, da sich in der Art, wie die Verformung in den Zuschnitt eingebracht wird, und die Art der Ablage für beide Greifer unterscheidet. Während der Netzgreifer, wie auch bei der Aufnahme, eine direkte und senkrechte Bewegung hin zur Ablageposition ausführt, führt dies beim Schaumstoffgreifer aufgrund der dabei verdrängten Luft zwischen der Form und dem großflächigen Schaumstoff zu einer negativen Beeinflussung des Zuschnitts. Für den Schaumstoffgreifer wird deshalb eine Ablagebewegung nach dem „Stempel-Abroll-Prinzip“ gewählt, wonach zunächst die obere Kante des Schaumstoffs auf Kontakt mit der Form gebracht wird und sich der Greifer anschließend in die Form eindreht.

10.3.4 Code-Generierung für eine externe Robotersteuerung

Das Szenario, in dem das als Roboter 1 bennante Trio an Lineareinheit, Roboterarm und Schaumstoffgreifer die Zuschnitte A bis C in einer realen Ausführung drapiert, wird in der TEZ auf der KUKA-eigenen Robotersteuerung ausgeführt. Dazu ist das von der Planung vorliegende Ergebnis in Form eines technischen Tasks in die herstellereigene Programmiersprache KRL zu übersetzen (siehe Abschnitt 8.2.3).

In KRL werden die Befehle *PTP* und *LIN* zur Erzeugung von Punkt-zu-Punkt- sowie Linearbewegungen des Roboters sowohl im kartesischen als auch im Achsraum eingesetzt. Die Ansteuerung der Linearachse, auf der der Roboter montiert ist, erfolgt in KRL über eine Zusatzachse *E1*. Insgesamt sieben analoge Ausgänge (Analog Outputs, AO) *AO1* bis *AO6* ermöglichen die Ansteuerung der einzelnen Vakuumkammern des Greifers hinter der Schaumstoffoberfläche. Da die Ausführung des KRL-Codes über eine einzelne KUKA-Steuerung (KRC) stattfindet, muss eine Synchronisation mehrerer Steuerungen in dieser Fallstudie nicht berücksichtigt werden.

Betrachtet wird im Folgenden das Aufgreifen eines Zuschnitts vom Aufnahmetisch. Der technische Task, der vom Fähigkeitenmodul „Aufnehmen“ erzeugt wird, enthält eine Sequenz von vier untergeordneten technischen Tasks:

1. Anfahren der Vorposition über dem Aufnahmetisch
2. Senkrechtes Anfahren des Aufnahmetisches mit der oberen Kante des Schaumstoffgreifers
3. Rollbewegung des Greifers mit zum entsprechenden Zeitpunkt stattfindender Aktivierung derjenigen Vakuumsektoren, die Kontakt zum Zuschnitt haben
4. Senkrechtes Abheben des Greifers vom Tisch

```
SIGNAL AO1 $OUT[249] TO $OUT[264]                ;Definition der Analogausgänge zur
...                                              ;Ansteuerung der Vakuumkammern
SIGNAL AO7 $OUT[345] TO $OUT[360]

;Daten aus Vermessung und CAD
DECL FRAME tisch                                 ;Vermessung des Aufnahmetischs
DECL FRAME greifflaeche_oben                     ;TCP der Schaumstoffoberkante
DECL FRAME greifflaeche_unten                    ;TCP der Schaumstoffunterkante
DECL FRAME greifer_mittelpunkt                   ;TCP für rollende Bewegung
tisch = {X .., Y .., Z .., A .., B .., C ..}
greifflaeche_oben = {X .., Y .., Z .., A .., B .., C ..}
greifflaeche_unten = {X .., Y .., Z .., A .., B .., C ..}
greiffer_mittelpunkt = {X .., Y .., Z .., A .., B .., C ..}


;================================================;
; (1) ANFAHREN DER VORPOSITION                   ;
;================================================;
PTP {E6AXIS: A1 .., A2 .., ..., A6 .., E1 ...}   ;Bewegung von Linearachse und
                                                 ;Roboter im Achsraum

;================================================;
; (2) SENKRECHTES ANFAHREN DES AUFNAHMETISCHES   ;
;================================================;
$BASE = tisch                                    ;Basis-Bezugskoordinatensystem
$TOOL = greifflaeche_oben                        ;Tool-Koordinatensystem
LIN {E6POS: X .., Y .., Z .., A .., B .., C ..}  ;Kartesische Bewegung des Roboters


;================================================;
; (3) ROLLENDE AUFNAHMEBEWEGUNG                  ;
;================================================;
;Erste Aktivierung an Vakuumkammern
TRIGGER WHEN DISTANCE=0.0 DELAY=0 DO AO6=30000   ;Oberer Sektor mitte
TRIGGER WHEN DISTANCE=0.0 DELAY=0 DO AO5=30000   ;Oberer Sektor rechts
TRIGGER WHEN DISTANCE=0.0 DELAY=0 DO AO2=30000   ;Oberer Sektor links
;Zweite Aktivierung an Vakuumkammern
TRIGGER WHEN DISTANCE=0.4 DELAY=0 DO AO7=30000   ;Mittlerer Sektor mitte
TRIGGER WHEN DISTANCE=0.4 DELAY=0 DO AO4=30000   ;Unterer Sektor rechts
TRIGGER WHEN DISTANCE=0.4 DELAY=0 DO AO3=30000   ;Unterer Sektor links
;Dritte Aktivierung an Vakuumkammern
TRIGGER WHEN DISTANCE=0.8 DELAY=0 DO AO1=30000   ;Unterer Sektor mitte
$BASE = tisch                                    ;Basis-Bezugskoordinatensystem
$TOOL = greiffer_mittelpunkt                     ;Tool-Koordinatensystem
LIN {E6POS: X .., Y .., Z .., A .., B .., C ..}  ;Rollende Bewegung, auf die
                                                 ;sich die Trigger beziehen

;================================================;
; (4) SENKRECHTES ABHEBEN DES GREIFERS           ;
;================================================;
$BASE = tisch                                    ;Basis-Bezugskoordinatensystem
$TOOL = greifflaeche_unten                       ;Tool-Koordinatensystem
LIN {E6POS: X .., Y .., Z .., A .., B .., C ..}  ;Kartesische Bewegung des Roboters
```

Listing 10.2. Nach KRL übersetztes Roboterprogramm für die Aufnahme eines Zuschnitts mit dem Schaumstoffgreifer.

Listing 10.2 zeigt ein Beispiel für die vier technischen Tasks, die nach KRL übersetzt wurden. Die Analogausgänge zur Schaltung der Vakuumkammern sind in den Zeilen 1 bis 3 als jeweils 16-Bit-Werte über eine Reihe an OUTs definiert. Zeilen 6 bis 13 definieren besondere Koordinatensysteme, die bei den folgenden Bewegungen eine Rolle spielen und aus der Vermessung der Roboterzelle bzw. über CAD-Modelle bekannt sind. So ist die für den Aufnahmetisch angefertigte Vermessung (Zeile 10) sowie drei relevante Koordinatensysteme des Greifers (Zeile 11 bis 13) angegeben. Ziel-Koordinatensysteme oder Ziel-Achsstellungen ab Zeile 19, die mit Punkte-Platzhaltern dargestellt sind, stellen die über geometrische Berechnungen ermittelten Werte dar. In Zeile 19 erfolgt das Anfahren der Vorposition über dem Aufnahmetisch (1) mit einer gleichzeitigen Bewegung von Linearachse und Roboter im Achsraum, da keine kartesische Bahn eines Tool-Centerpoints (TCP) erforderlich ist. Für den Beginn der rollenden Zuchnittsaufnahme (2) wird die obere Schaumstoffkante des Greifers in einer Linearbewegung auf den Zuschnitt bewegt. Dazu werden sowohl das Basiskoordinatensystem, auf das sich der anzufahrende Zielpunkt bezieht, als auch das Toolkorrdinatensystem spezifiziert, das den „Teil“ des Greifers als TCP angibt, mit dem der Zielpunkt erreicht werden soll. Die rollende Aufnahmebewegung (3) ist über eine Linearbewegung realisiert, die den theoretischen Kreismittelpunkt der runden Schaumstoffoberfläche als TCP parallel zum Aufnahmetisch bewegt und dabei gleichzeitig dessen Orientierung ändert. Während dieser rollenden Bewegung sind in den richtigen Momenten die einzelnen Vakuumkammern des Greifers zu aktivieren, um den Zuschnitt an die Schaumstoffoberfläche anzuheften. Insgesamt sieben Trigger setzen zu einem bestimmten prozentualen Fortschritt der rollenden Bewegung ihren jeweiligen Analogausgang zur Aktivierung der Vakuumkammer. Das abschließende Abheben des Greifers vom Aufnahmetisch (4) geschieht analog zu (2) mit einer Linearbewegung nach oben.

Die Übersetzung von technischen Tasks, die vom Fähigkeitenmodul „Drapieren“ erzeugt werden, erfolgt auf analoge Weise. Tasks des Fähigkeitenmoduls „Bereitstellen“ resultieren in einer KRL-Anweisung, die auf das Setzen eines Inputs zur Bestätigung der Zuschnittsauslegung durch den Werker wartet. Das gesamte KRL-Programm kann auf die Robotersteuerung (KRC) geladen und dort zur Ausführung gebracht werden.

Die Ansteuerungen der Roboter sind auf Basis der Roboterzellenvermessung und der geometrischen Daten der Greifer präzise geplant. Ergibt sich eine Änderung in der Roboterzelle, z. B. wird der Aufnahmetisch verrückt, ist das Roboterprogramm zur Fertigung nicht mehr gültig. Damit die Planung nicht neu erfolgen muss, die Eingabe des Experten nicht erneut abgefragt und auch die Übersetzung nicht nochmals ausgeführt werden muss, werden Roboterbewegungen nie in absoluten Koordinatensystemen, sondern immer relativ zu Objekten der Roboterzelle angegeben. Auch bei der Überführung in KRL werden die relativen Angaben beibehalten und bei Bewegungen in Form einer relativen Basis *$BASE* und einem zu bewegenden TCP *$TOOL* eingesetzt. Bei einer Veränderung der Roboterzelle sind nun lediglich die Vermessungen der Roboterzelle erneut durchzuführen und die dabei erlangten Messdaten im KRL-Programm (im Beispiel Zeile 10 bis 13) auszutauschen. Nach einer Überprüfung im Testlauf, ob alle Bewegungen weiterhin von den Robotern ausgeführt und Zielpositionen erreicht werden können, ist das KRL-Programm mit den selben Präzisionsgarantien wieder einsatzbereit.

10.4 Evaluation

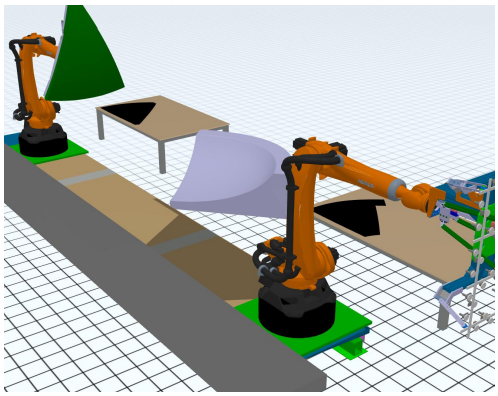
Mit der Einbeziehung der oben genannten Expertenmodule ist der Planungsansatz $PaRTs$ für die Planung von Fertigungsprozessen mit CFK einsetzbar. Dass der Planungsansatz auch hier seine Stärken hat, wird mittels zweier unterschiedlicher Szenarien gezeigt: Die Planung mit zwei unterschiedlichen Greifern, die unabhängig voneinander und parallel arbeiten können, wurde mit dem Netz- und Schaumstoffgreifer durchgeführt und über die Simulation ausgewertet. Abschnitt 10.4.1 erläutert die Ergebnisse hierzu. Um die Tauglichkeit der von $PaRTs$ erzielten Planungsergebnisse für einen realen Einsatz zu zeigen, beschreibt Abschnitt 10.4.2 die Planung und Ausführung von Drapierprozessen mit einem einzeln betrachteten Greifer. Eine abschließende Auswertung erfolgt in Abschnitt 10.4.3.

10.4.1 Ergebnisse des Ansatzes in der Simulation

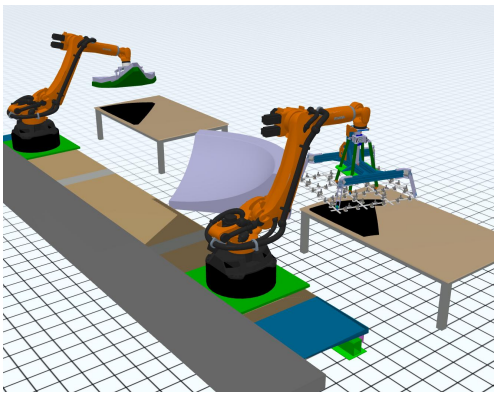
Die Planung der Drapierprozesse für die Zuschnitte A, B und C über den Planungsansatz $PaRTs$ äußert sich gegenüber dem Experten als geführter Modus zur Eingabe der noch erforderlichen Daten. Wann immer ein Fähigkeitenmodul die Position eines Zuschnitts auf dem Tisch oder die Greiferposition bei der Ablage zur Berechnung der Zuschnittsaufnahme benötigt, öffnet sich ein Dialog zur interaktiven Eingabe (siehe Abbildungen 10.6 und 10.7 in Abschnitt 10.3.3). Für jeden Zuschnitt wird seine initiale Position auf dem Aufnahmetisch abgefragt, wobei im vorliegenden Szenario beide Aufnahmetische identisch sind und eine Eingabe für beide gilt. Dank der intuitiven Verschiebung und Drehung des Zuschnitts mit der Maus ist eine sekundenschnelle, exakte Positionsangabe möglich. Anders als mit den Aufnahmetischen verhält es sich mit den Greifern: Deren optimale Ablagepositionen in der Form können nicht gleichermaßen für beide Greifer herangezogen werden. Da Netz- und Schaumstoffgreifer unterschiedliche Wirkprinzipien besitzen (kontaktgenaues Anfahren der Coanda-Saugmodule, Eindrücken mit dem Schaumstoff), ist deren optimale Ablageposition auch für denselben Zuschnitt in der Regel nicht identisch. Für alle drei Zuschnitte ist mit jedem Greifer dessen optimale Ablageposition separat zu ermitteln. Über den Dialog ist dies aus der Perspektive des Greifers durchführbar, auf dem der Zuschnitt positioniert wird. Die weniger intuitiven Bewegungen des Roboters werden vom Computer daraus automatisch errechnet. In Kombination mit der Abstandsanalyse zwischen den Greifpunkten und der Form stellt dies eine ungemeine Vereinfachung für den Experten dar.

Insgesamt sind es damit neun Eingaben, die beim Einsatz zweier Greifer zur Drapierung von drei Zuschnitten vom Experten zu machen sind. Die zwischenzeitliche Planungs- und Berechnungszeit, in der kein Dialog geöffnet ist, nimmt nur wenige Sekunden in Anspruch, sodass die gesamte Planung – abhängig vom Greifer und von der Routine des Experten bei der Eingabe – innerhalb weniger Minuten abgeschlossen werden kann. Das Resultat der Planung – ein verschachtelter technischer Task – kann direkt über die Simulation wiedergegeben werden. Abbildungen 10.8a bis 10.8l stellen den Ablauf der Ausführung chronologisch dar. Zwei Kernaspekte sind besonders herauszustellen:

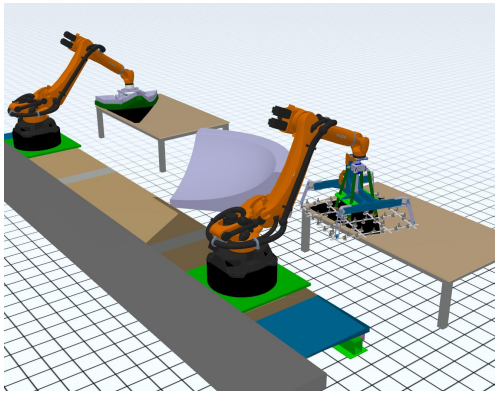
1. Die Gesamtaufgabe (das Drapieren dreier Zuschnitte) wird so auf beide Greifer aufgeteilt, dass die Gesamtausführungsdauer möglichst gering ist. Der Netzgreifer



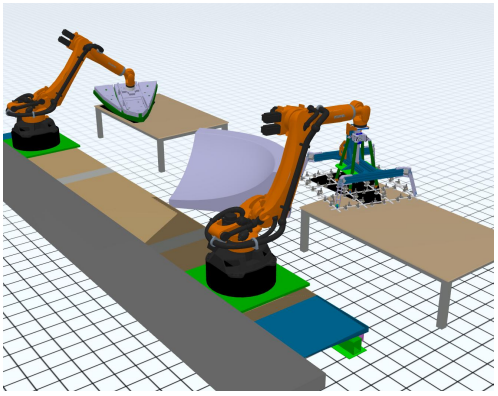
(a) Zu Beginn werden Zuschnitte A und B auf den Aufnahmetischen bereitgestellt.



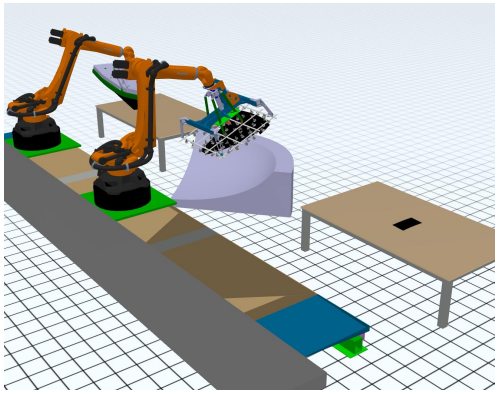
(b) Beide Greifer positionieren sich parallel über dem jeweiligen Zuschnitt.



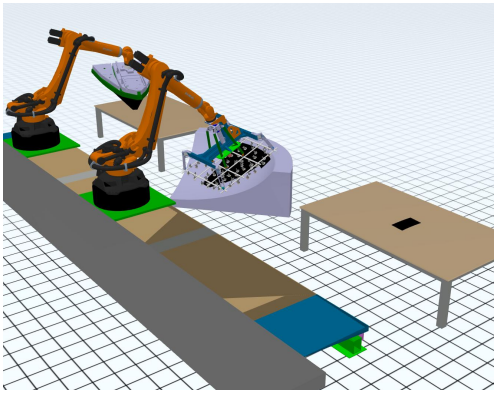
(c) Die Aufnahme durch den Netzgreifer erfolgt über eine senkrechte Bewegung, …



(d) … wohingegen der Schaumstoffgreifer eine rollende Aufnahme durchführt.

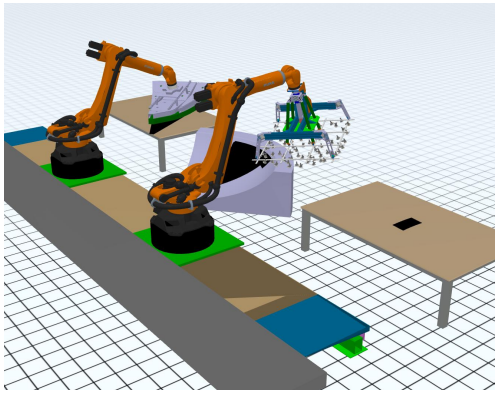


(e) Der Schaumstoffgreifer wartet, bis Zuschnitt A durch den Netzgreifer verformt …

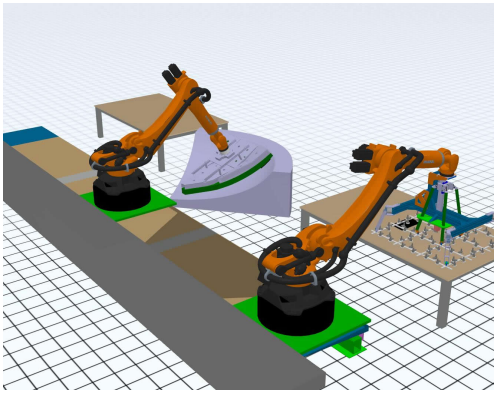


(f) … und an seiner Zielposition in der Viertelkalottenform abgelegt ist.

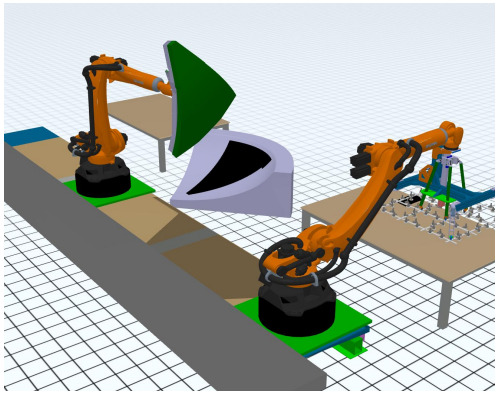
Abbildung 10.8. Von *PaRTs* generierter Ablauf zur Drapierung der Zuschnitte A, B und C mit zwei Robotern. Zum Einsatz kommen der Schaumstoffgreifer und der Netzgreifer.



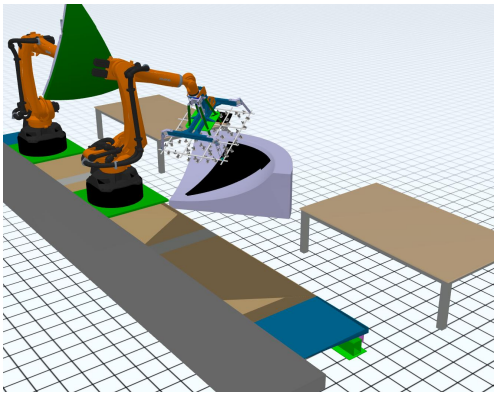
(g) Im Wechsel kann sich nun der Schaumstoffgreifer zur Form bewegen, …



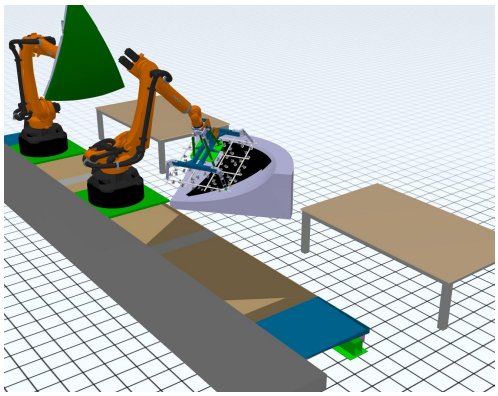
(h) …um Zuschnitt B zu drapieren. Derweil nimmt der Netzgreifer Zuschnitt C auf.



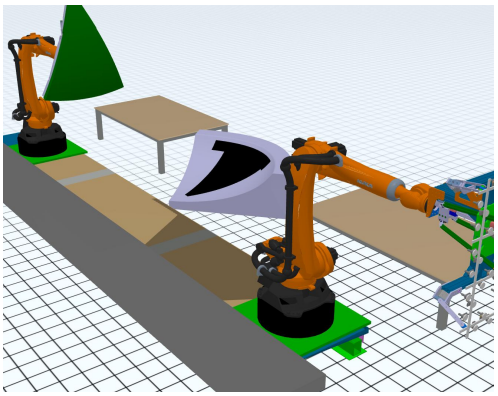
(i) Während der Schaumstoffgreifer seine Initialposition einnimmt, …



(j) …kann sich der Netzgreifer wieder zur Form bewegen und dort …



(k) …die Verformung in Zuschnitt C einbringen und ihn in der Form ablegen.



(l) Nach der Ausführung befinden sich beide Roboter wieder in ihrer Ausgangsposition.

Abbildung 10.8. Von *PaRTs* generierter Ablauf zur Drapierung der Zuschnitte A, B und C mit zwei Robotern. Zum Einsatz kommen der Schaumstoffgreifer und der Netzgreifer.

widmet sich Zuschnitt A (siehe Abbildung 10.8f), anschließend wird Zuschnitt B vom Schaumstoffgreifer drapiert (siehe Abbildung 10.8h). Zuletzt ist der Netzgreifer derjenige von beiden, der mit der Drapierung von Zuschnitt C eine schnellere Fertigstellung erzielt (siehe Abbildung 10.8k).

2. Obwohl keine explizite Kollisionserkennung zum Einsatz kommt, ist bei der parallelen Ausführung stets garantiert, dass sich immer nur ein Roboter gleichzeitig vor der Form zur Drapierung des Zuschnitts befindet. Dabei handelt es sich um einen Effekt der Parallelisierung von Ausführungsplänen (siehe Abschnitt 7.3), wonach eine strikte Nacheinanderausführung all derjenigen Tasks erzwungen wird, die die Form als Ressource allokieren.

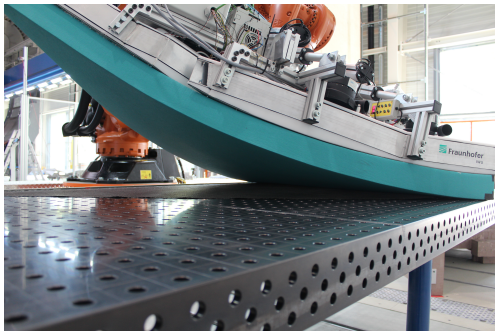
Sowohl die Planung als auch die Ausführung des Ergebnisses über die Simulation war erfolgreich. Die Gesamtausführungsdauer der Simulation, um die drei Zuschnitte aufzunehmen, die Verformung einzubringen und sie in die Viertelkalottenform zu applizieren, betrug $66{,}62$ Sekunden.

10.4.2 Ablage von CFK-Textilien in einer realen Roboterzelle

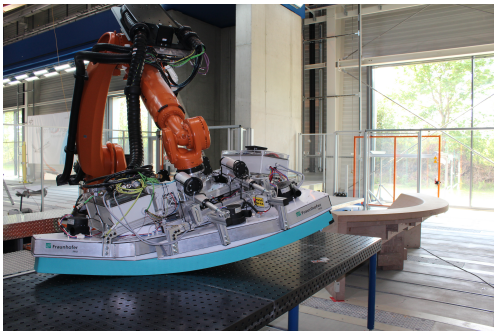
Im Rahmen des Projekts AZIMUT wurden alle drei Zuschnitte von den unterschiedlichen Greifern in einer realen Ausführung drapiert. Die über den Planungsansatz $PaRTs$ berechneten Abläufe zur Drapierung der Zuschnitte wurden in KRL-Programme übersetzt und mit denen des AZIMUT-Projekts verglichen. Aufgrund der exakten Nachbildung der TEZ als Modell für die Planung und die Eingabe der selben Werte (z. B. Zuschnittsposition auf dem Aufnahmetisch) konnten für den Schaumstoffgreifer und die drei Zuschnitte identische KRL-Programme erzeugt werden. Die Ergebnisse des Planungsansatzes $PaRTs$ werden daher anhand der im Rahmen des AZUMUT-Projekts stattgefundenen Versuchsreihen diskutiert.

Zu Beginn wird der zu drapierende Zuschnitt manuell auf dem Aufnahmetisch ausgelegt. Dabei wird exakt die Position nachgebildet, wie sie bei der Planung über den Dialog angegeben wurde. Abbildung 10.9 zeigt eine chronologische Bilderfolge über den anschließend ausgeführten Drapiervorgang mit dem Schaumstoffgreifer für Zuschnitt B, den größten der drei Zuschnitte. Dargestellt ist die rollende Aufnahmebewegung (siehe Abbildungen 10.9a bis 10.9c), die zu bestimmten Zeitpunkten die unterhalb der Schaumstoffschicht angebrachten Vakuumkammern aktivieren, um den Zuschnitt aufzunehmen. Nach der Transferfahrt zur Viertelkalottenform bringt der Greifer die Verformung in den Zuschnitt ein, indem er die zwei Flügel anklappt (siehe Abbildung 10.9d). Die Ablage in der Form erfolgt über ein Stempel-Abroll-Prinzip, wonach zunächst die obere Kante des Schaumstoffs in die Form gedrückt und anschließend der Greifer passgenau eingedreht wird. Dadurch wird eine unkontrollierte Verdrängung der Luft zwischen Schaumstoff und Form vermieden, die die Qualität der Drapierung beeinflussen würde. Nach dem Abschalten der Vakuumkammern wird der Greifer aus der Form bewegt (siehe Abbildung 10.9f); der Zuschnitt ist drapiert.

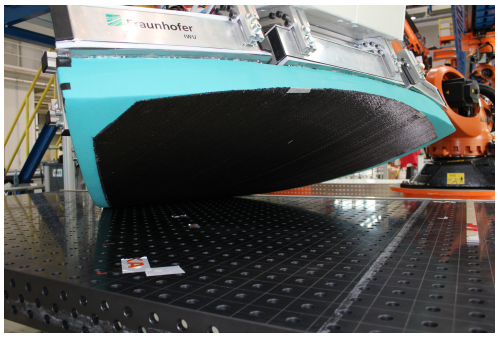
Auch mit dem Netzgreifer wurde eine Versuchsreihe zur Drapierung der Zuschnitte A, B und C durchgeführt. Das Programm wurde in analoger Weise erstellt und in ein KRL-Programm überführt. Abbildung 10.10a zeigt den Netzgreifer nach dem Aufgreifen des



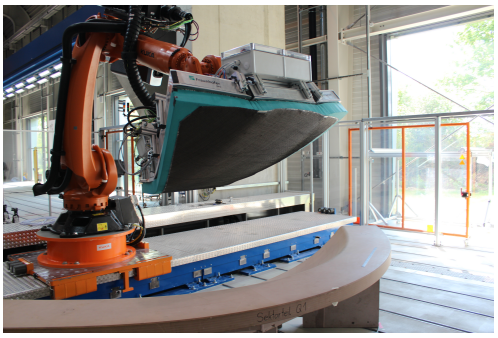
(a) Die obere Kante des Schaumstoffgreifers wird für die Aufnahme auf dem Aufnahmetisch positioniert.



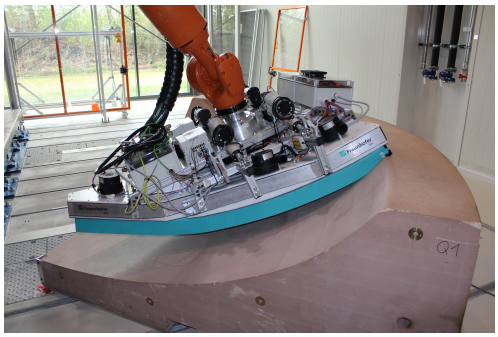
(b) Anschließend erfolgt die rollende Aufnahmebewegung und die zeitgleiche Aktivierung der Vakuumkammern.



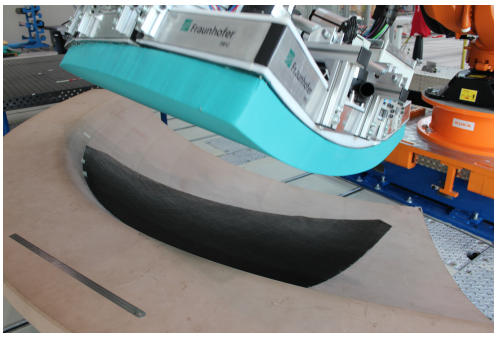
(c) Nach Beendigung der rollenden Bewegung haftet der Zuschnitt auf der Schaumstoffoberfläche des Greifers



(d) Es erfolgt die Transferfahrt zur Form, wo über den Greifer die Verformung in den Zuschnitt eingebracht wird.

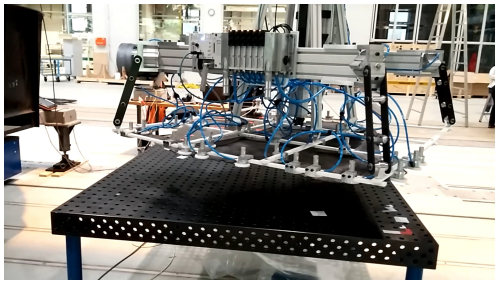


(e) Für die eindrehende Ablage wird auch hier zunächst die obere Schaumstoffkante des Greifers positioniert, …



(f) …bevor die finale Ablage und die Deaktivierung der Vakuumkammern erfolgen. Der Zuschnitt liegt in der Form.

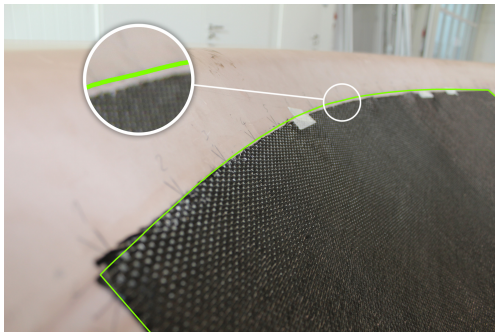
Abbildung 10.9. In der TEZ stattfindende, reale Ausführung des geplanten und nach KRL übersetzten Ablaufs zur Drapierung von Zuschnitt B mit dem Schaumstoffgreifer.



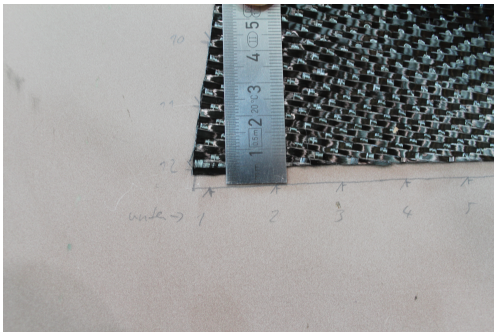
(a) Planare Aufnahme des Zuschnitts

(b) Zuschnitt nach der Drapierung

Abbildung 10.10. Durchführung der Versuchsreihe mit dem Netzgreifer.



(a) Ziel- und Ist-Position von Zuschnitt B



(b) Untere Kante von Zuschnitt C

Abbildung 10.11. Genauigkeit bei der Drapierung der Zuschnitte mit dem Schaumstoffgreifer.

Zuschnitts über dem Aufnahmetisch. Die Drapierung über den Netzgreifer erfolgte in die originale, schwarze Kalottenform, wie in Abbildung 10.10b abgebildet. Zu sehen ist ebenso der noch verformte Greifer, nachdem er den Zuschnitt abgelegt hat und ein Stück aus der Form gefahren ist.

Erwähnenswert ist die Tatsache, dass bei den ersten Testläufen der drapierte Zuschnitt einige Zentimeter versetzt zur eingezeichneten Zielkontur in der Form lag. Über Nachmessungen konnte ermittelt werden, dass die Positionen der geplanten Programme richtig waren, allerdings bei der Einzeichnung der Zielkonturen ein Fehler unterlaufen ist. Im klassischen Teaching-Verfahren hätte der Inbetriebnehmer den Greifer so einprogrammiert, dass er die Zuschnitte an die falsch eingezeichnete Position bringt. Der Fehler wäre hierbei wohl nicht aufgefallen. Dabei zeigt sich ein beträchtlicher Vorteil der automatischen Programmierung: Aufgrund der Berechnungen mit idealen Werten führen eventuell selbst kleine Unstimmigkeiten schnell zu größeren Problemen und fallen auf. Solche Fehler gehen nicht unter und eine fehlerhafte Fertigung unterbleibt; die Unstimmigkeiten können identifiziert und entsprechend berücksichtigt werden.

Bei den späteren Versuchsläufen wurde die Ablagegenauigkeit der Zuschnitte untersucht. Großteils werden die Zielkonturen in der Form präzise eingehalten. An einigen Stellen treten allerdings Abweichungen in der Größenordnung weniger Millimeter auf. Dabei sind die Abweichungen besonders entlang der horizontalen Kanten im oberen

sowie unteren Teil der Zuschnittskonturen festzustellen. Abbildung 10.11a zeigt den Verlauf der oberen Kante von Zuschnitt B, der vom Schaumstoffgreifer bei einer der Ausführungen in die Viertelkalottenform drapiert wurde. Während der Zuschnitt in den Eckbereichen mit der eingezeichneten Kontur übereinstimmt, „sackt" der Zuschnitt in der Mitte erkennbar um einige Millimeter nach unten ab. Dagegen weist Zuschnitt C, wie in Abbildung 10.11b dargestellt, einen durchgängigen Versatz von 4 Millimetern nach oben entlang der horizontalen Konturlinie auf. Im Allgemeinen kann bei den Ablageergebnissen kein systematisch bedingter Versatz festgestellt werden. Bei mehreren Durchläufen wurden leicht abweichende Prozessdaten im Dialog zur Positionierung des Zuschnitts auf dem Greifer ausprobiert, die jeweils zu unterschiedlichen Einflüssen auf das Ablageergebnis führten. Um den Zuschnitt manuell in Einklang mit der Zielkontur zu bringen, konnte nicht etwa der gesamte Zuschnitt verschoben werden. Stattdessen war ein teilweises „Verziehen" des Zuschnitts über dessen Verscherungseigenschaft notwendig. Vermutlich rühren die Ungenauigkeiten überwiegend vom Einbringen der Verformung in den Zuschnitt durch den Greifer her. Tatsächlich scheint es so, dass es nicht für jeden Zuschnitt möglich ist, die komplexe 3D-Verformung über die Wirkflächen des Schaumstoffgreifers physikalisch exakt einzubringen. Untermauert wird diese Annahme dadurch, dass auch bei einem manuellen Teachen des Roboterprogramms durch einen Ingenieur (siehe Abschnitt 3.2.2) keine deutlich besseren Ablagegenauigkeiten erzielt werden konnten [44]. Ungeachtet dessen bewegen sich die beobachteten Abweichungen im Rahmen der erlaubten Toleranz von $[+5mm, -7.5mm]$ [44].

Zusammenfassend war die Planung über den vorgestellten Planungsansatz $PaRTs$ durchwegs erfolgreich. Die Erstellung der Roboterprogramme zur Drapierung der einzelnen Zuschnitte verlief für den Experten mit sehr wenig Aufwand und in kurzer Zeit. Mit den präzise erfassten Vermessungsdaten der TEZ war der nach KRL übersetzte Programmcode sofort ausführbar und bewegte den Greifer präzise an die vorgegebenen Punkte relativ zum Aufnahmetisch und zur Viertelkalottenform.

10.4.3 Auswertung

Im Zusammenhang mit der CFK-Fallstudie wurden fünf Problemstellungen postuliert, die gleichermaßen Anforderungen an den Planungsansatz $PaRTs$ sind. Mit der Anwendung des Planungsansatzes auf das CFK-Fallbeispiel und den erfolgten Evaluationen können die Problemstellungen aufgegriffen und schlussendlich bewertet werden.

1. **Auswertung des Plybooks:** Problemstellung 1 adressiert die Art und Weise, wie die Daten des Plybooks zu interpretieren sind, sowie die Frage, wie diese für den Planungsansatz aufbereitet und verfügbar gemacht werden können. Es hat sich gezeigt, dass das Format der einheitlichen Produktbeschreibung sich nicht nur für die Definition des fertigen Bauteils eignet (siehe LEGO-Fallstudie), sondern mit den zusätzlichen planungsrelevanten Prozessparametern vollständig die in einem Plybook angegebenen Prozessdaten beschreiben kann. Für die CFK-Fallstudie übernimmt ein Generator die Übersetzung des Plybooks in das Format der einheitlichen Produktbeschreibung.

2. **Bereitstellen von Prozessparametern:** Problemstellung 2 formuliert die Frage, wie Inbetriebnehmer und Ingenieur ihr Wissen zur Planung beisteuern können, damit die korrekte Roboter- und Greiferansteuerung (technische Sicht) und die Wahl der richtigen Prozessparameter bei der Verarbeitung von CFK-Zuschnitten (Domänensicht) automatisiert erfolgen kann. Der Planungsansatz $PaRTs$ ermöglicht es, sämtliche automatisch ableitbaren Berechnungen und Bewegungsplanungen über Fähigkeitenmodule abzubilden und so dem Planungsansatz verfügbar zu machen. Der Inbetriebnehmer formuliert auf diese Weise sein Wissen zur Roboteransteuerung und zum Greiferhandling, auf dessen Basis z. B. die rollende Aufnahmebewegung des Schaumstoffgreifers berechnet werden kann. Auch wenn eine vollständig automatische Planung angestrebt wird, ist es dem Experten dennoch möglich, bestimmte Prozessdaten erst zur Planungszeit beizusteuern. In der CFK-Fallstudie werden zwei Zuschnitts-abhängige Prozessdaten – die Position des Zuschnitts auf dem Aufnahmetisch und die Position des Greifers bei der Drapierung – über ein interaktives Dialogsystem im Zuge der Planung ermittelt. Dieses steht den Fähigkeitenmodulen zur bedafsgemäßen Ermittlung der Prozessdaten zur Verfügung. Es ist aber auch denkbar, die Planung völlig automatisch ohne die Interaktion mit dem Experten zu gestalten. So wäre die Zuschnittsposition auf dem Aufnahmetisch durch ein Zubringersystem bestimmt, und die optimale Greiferposition bei der Drapierung kann über einen angebundenen Optimierungsalgorithmus ermittelt werden.
3. **Austauschbarkeit des Endeffektors:** Problemstellung 3 fordert eine Untersuchung, inwieweit der konkret einzusetzende Endeffektor für die Planung austauschbar bzw. durch weitere Endeffektoren ergänzbar ist, und mit welchem Aufwand ein solcher Wechsel (in Kombination mit Problemstellung 2) verbunden ist. Die Evaluation hat gezeigt, dass lediglich die für einen Greifer spezifischen Fähigkeitenmodule zu entwickeln und dem Planungansatz hinzuzufügen sind, um die Planung um einen weiteren Greifer zu erweitern. Für den Netz- und den Schaumstoffgreifer sind dies die Fähigkeitenmodule „Aufnehmen“ und „Drapieren“, die sich in ihrer Art der Zuschnittsverarbeitung (rollende oder stempelnde Bewegung) unterscheiden. Werden diese für einen zusätzlichen Greifer bereitgestellt, erfolgt die Planung unter Berücksichtigung der neuen Möglichkeiten an Bearbeitungsschritten.
4. **Übergang zur realen Fertigung:** Problemstellung 4 benennt die Herausforderungen, die bei der Online-Ausführung von offline geplanten Roboterprogrammen zu lösen sind, um ein CFK-Bauteil real zu fertigen. Im Rahmen dieser Arbeit wurden in der Praxis gängige Verfahren zur Vermessung der Roboterzelle vorgestellt. Die Planung über $PaRTs$ berücksichtigt diese Vermessungsdaten und bildet die reale Roboterzelle in der Simulation ab, auf die sich die Planung für geometrische Berechnungen stützt. Die reale Ausführung der geplanten Prozesse ist über eine Übersetzung nach KRL realisiert, die Informationen zu geometrischen Bezugsobjekten beibehält und eine spätere Anpassung der Roboterzelle erlaubt.
5. **Präzision:** Problemstellung 5 adressiert die Frage, ob bei einer Offline-Planung die vorgegebene Genauigkeit bei der Herstellung von CFK-Bauteilen erlangt

werden kann. Tatsächlich kam es bei der Drapierung der Zuschnitte zu Abweichungen, diese sind allerdings überwiegend auf die komplexen Verformprozesse und die Wirkprinzipien der Greifer zurückzuführen. Grundsätzlich kann über den Planungsansatz $PaRTs$ eine beliebig hohe Präzision erlangt werden, wenn die verwendeten Modelle und Berechnungen der Fähigkeitenmodule bzw. die Vermessungsdaten der Roboterzelle entsprechend genau sind.

$PaRTs$ kann allen Anforderungen genügen und dabei durch die automatische Programmierung eine immense Zeitersparnis erreichen. Ist nach einem initialen Aufwand die virtuelle Roboterzelle modelliert und sind die Fähigkeitenmodule implementiert, geht die Planung auch für eine Vielzahl von zu verarbeitenden Zuschnitten schnell vonstatten. Tabelle 10.1 gibt einen Überblick über den zeitlichen Aufwand, wobei für jeden Zuschnitt zum einen das konventionelle manuelle Teachen und zum anderen die automatische Planung über $PaRTs$ betrachtet wird. Für die großen Zuschnitte A und B wurden sechs bis sieben Minuten Planungszeit mit $PaRTs$ benötigt, wobei wenige Sekunden davon auf die eigentliche Planung und der größte Teil auf die Eingabe von Prozessdaten durch den Experten entfallen. Gerade bei den großen Zuschnitten stellte sich die Positionierung des Greifers in der Form schwieriger dar, da einerseits eine geeignete Überdeckung mit Coanda-Saugmodulen erzielt werden soll und andererseits die Abstände der Greifpunkte zur Form zu beachten sind. Für den Zuschnitt C wurden lediglich zwei Minuten dafür benötigt. Zu allen drei Zuschnitten wurde ein Drapierprogramm ebenfalls über die Methode des manuellen Teachens erstellt. Der zeitliche Aufwand war mit teilweise einer Stunde deutlich höher, wobei das Teachen für Zuschnitt C vergleichsweise zügig erfolgte. Die Zuschnitte wurden mit dem Schaumstoffgreifer so aufgegriffen, dass jeweils die Oberkante des Zuschnitts über die Greiffläche hinausragte. Das half, um die Ablageposition per Augenmaß an die eingezeichnete Zielkontur anzupassen. Für den kleinen Zuschnitt C funktionierte dies recht gut, bei den größeren Zuschnitten A und B kam es wiederholt zu Ungenauigkeiten im unteren, vom Greifer verdeckten Bereich. Ein mehrmaliges Wiederholen des Teach-Vorgangs war notwendig, bevor eine zum Ergebnis von $PaRTs$ vergleichbare Drapierqualität erreicht wurde.

Erfahrungen anderer Anwender zufolge dauert das manuelle Teachen des Roboterprogramms üblicherweise sogar zwischen zwei Stunden und einem Tag für einen einzelnen Zuschnitt [44]. Hochgerechnet auf die Anzahl an Zuschnitten eines gesamten CFK-Bauteils stellt dies einen enormen Zeitaufwand von mehreren Wochen bis hin zu Monaten für die Programmierung dar. In diesem Fall ist eine Automatisierung der Fertigung

Tabelle 10.1. Zeiten, die von einem Inbetriebnehmer zum Teachen bzw. vom Planungsansatz zur Planung des Drapierprozesses für den jeweiligen Zuschnitt benötigt werden.

Zuschnitt	Konventionelles Teachen	Interaktive Planung über $PaRTs$
A	0:52 h	0:06 h
B	1:06 h	0:07 h
C	0:31 h	0:02 h

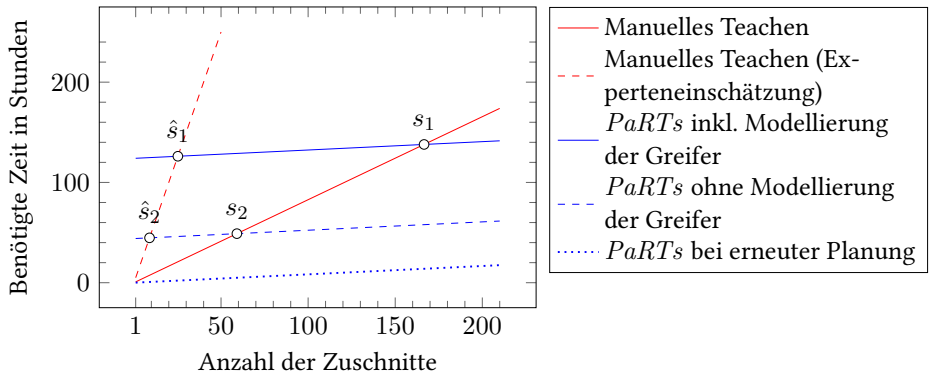


Abbildung 10.12. Benötigte Zeit für das manuelle Teachen von Zuschnittsdrapierungen (rot) und für die Planung über $PaRTs$ (blau) in Abhängigkeit der Anzahl an Zuschnitten, aus denen das zu fertigende CFK-Bauteil besteht.

mit entsprechend kleinen Stückzahlen nicht ökonomisch. Der Planungsansatz $PaRTs$ dagegen stellt für die Planung einer derartigen Fertigung in Kombination mit dem interaktiven Dialogsystem zur Einbeziehung von Experten während der Planung eine wesentlich schnellere Alternative dar. Die Implementierung der Expertenmodule war für einen geübten Entwickler in weniger als einer Arbeitswoche realisierbar, wobei das anschließende Testen und Beheben von Programmierfehlern bereits einberechnet ist. Für das Einmessen der Roboterzelle ist etwa ein halber Arbeitstag anzusetzen. Die Modellierung der Greifer war mit circa zwei Wochen etwas aufwendiger, wobei diese auch für spätere Produktionen ähnlicher Bauteile ohne Mehraufwand einsetzbar ist.

Nimmt man einen Arbeitstag mit 8 Stunden und eine Arbeitswoche mit 40 Stunden an, so ergeben sich für das konventionelle Teachen eines Zuschnitts gemäß Tabelle 10.1 durchschnittlich etwa 50 Minuten. Für die Planung über $PaRTs$ wurden einmalig 124 Stunden für die Entwicklung der Expertenmodule und die Greifermodellierung sowie durchschnittlich 5 Minuten für die Interaktion bei der Planung je Zuschnitt benötigt. Abbildung 10.12 skizziert mit den durchgezogenen Geraden jeweils die Zeit, die für eine gegebene Anzahl von Zuschnitten hierfür benötigt würde. Daraus ist abzuleiten, dass sich ab 167 Zuschnitten ein Zeitvorteil bei der Verwendung von $PaRTs$ ergibt (Schnittpunkt s_1). Stünden die Werkzeuge bereits zuvor als Modelle zur Verfügung, dann entfielen die zwei Wochen Modellierungsaufwand (blau gestrichelte Gerade). In diesem Fall würde sich ein Zeitvorteil bereits bei einem Bauteil mit 60 zu drapierenden Zuschnitten einstellen (Schnittpunkt s_2).

Geht man bei den benötigten Zeiten für das manuelle Teachen jedoch – wie von den erfahrenen Anwendern angegeben [44] – von den 2 bis 8 Stunden ($= \varnothing 5h$) aus, so bringt der Einsatz von $PaRTs$ bereits ab 26 Zuschnitten (bei zu modellierenden Greifern) bzw. ab 9 Zuschnitten (bei bereits existierenden Greifermodellen) eine Zeitersparnis. Dabei nimmt die Zeitersparnis mit jedem zusätzlichen Zuschnitt des zu fertigenden Bauteils stetig zu. Die grafische Darstellung der zeitlichen Überschneidungen sind in Abbildung 10.12 mit den Schnittpunkten $\hat{s}_1$ und $\hat{s}_2$ angegeben. In dieser Gesamtbetrachtung des zeitlichen Aufwands lohnt sich der Einsatz von $PaRTs$ zur Fertigung

der Druckkalotte bereits dann, wenn nur ein einzelnes Exemplar angefertigt wird. Ist das Fertigungsprogramm der Druckkalotte – etwa aufgrund einer Änderung im Roboterzellenlayout – erneut zu erstellen, oder soll die Montage für eine Produktvariante der Druckkalotte automatisiert werden, so kann bereits ab dem ersten Zuschnitt von einer signifikanten Zeitersparnis profitiert werden. In Abbildung 10.12 ist der zeitliche Aufwand hierfür über die blau gepunktete Gerade angegeben. Hochgerechnet auf die Gesamtanzahl der Zuschnitte im Bauteil stellt dies einen beträchtlichen Vorteil von $PaRTs$ gegenüber der klassischen Methode dar. Hinzu kommt, dass bei $PaRTs$ mehrere Entwickler gleichzeitig eingesetzt werden können, um die Expertenmodule und die Modelle der Endeffektoren in wesentlich kürzerer Zeit umzusetzen. Beim manuellen Teachen hingegen ist diese zeitliche Optimierbarkeit nicht gegeben; die Roboterzelle ist den gesamten Programmierzeitraum über belegt. Die Fallstudie zeigt deutlich, dass dank $PaRTs$ auch von einer Parallelisierung mit mehreren Robotern profitiert und die Produktion dadurch enorm beschleunigt werden kann.

Zusammenfassung. In diesem Kapitel werden die Ergebnisse, die im Rahmen dieser Dissertation erzielt wurden, nochmals dargelegt und bewertet. Die Arbeit schließt mit einem Ausblick auf mögliche weitere Forschungsfragen, die sich insbesondere mit dem Debugging der Montageplanung zur Analyse des Planungsgeschehens auseinandersetzen.

11 Fazit und Ausblick

Die anfänglich gestellte Frage, wie die Programmierung von Montageaufgaben in flexiblen Roboterzellen auf effiziente und ökonomische Weise auch für die Produktion mit hoher Variabilität und geringen Stückzahlen ermöglicht werden kann, wurde in dieser Arbeit eingehend beleuchtet und mit der Vorstellung des allgemeinen Planungsansatzes $PaRTs$ beantwortet. Die dabei erzielten Ergebnisse werden in Abschnitt 11.1 zusammengefasst und bewertet. Ein Ausblick auf weiterführende Forschungsmöglichkeiten findet sich in Abschnitt 11.2.

11.1 Bewertung der erzielten Ergebnisse

Die Freiheitsgrade intelligenter und flexibler Roboteranlagen ermöglichen die automatisierte Fertigung von komplexen Bauteilen, jedoch stellen sie den Programmierer vor die Herausforderung einer nur schwer zu bewältigenden Komplexität bei der Entwicklung adäquater Steuerungssoftware. Hier bietet sich der Einsatz automatischer Programmierung an, um die Steuerungssoftware für solche Systeme zur Fertigung der Produkte automatisch über einen Computer zu berechnen. Doch bedarf es für eine erfolgreiche Fertigungsplanung zum einen detaillierter Informationen über die Domäne und den Prozess, die nur ein jeweiliger Fachexperte mit entsprechendem Know-How beisteuern kann, und zum anderen einer tragfähigen Strategie, die das Komplexitätsproblem der Planung beherrschbar macht.

Die vorliegende Dissertation leistet mit dem vorgestellten Planungsansatz $PaRTs$ und seinen Konzepten zur Einbeziehung von Expertenwissen bzw. zur Reduktion der Planungskomplexität einen elementaren Beitrag, die Automatisierung von Kleinserienproduktionen in flexiblen Fertigungsanlagen zu ermöglichen und deren Freiheitsgrade nutzbar zu machen. Ermöglicht wird dies durch ein geeignetes Zusammenwirken

mehrerer eigens entwickelter Techniken und Konzepte, die jeweils einzelne der zugrundeliegenden Problemstellungen im Gesamtkontext der Fertigungsplanung erfolgreich lösen. Kapitel 3 stellte zwei Fallstudien vor, die diese Problemstellungen adressieren und die auf üblichem Wege nur mit sehr hohem Aufwand in eine automatisierte Fertigung überführt werden können.

Bei der Umsetzung einer automatisierten Produktfertigung existieren meist mehrere Handlungsfelder, in denen verschiedene Fachexperten ihren jeweiligen Beitrag zur Prozessentwicklung leisten. Kapitel 4 gab einen Überblick über den Aufbau des Planungsansatzes $PaRTs$ und stellte dessen mehrstufiges Phasenmodell vor, das eine Untergliederung der Planung in drei wesentliche Abstraktionsebenen mit jeweils eigenen Verantwortlichkeiten vorschlägt. Diese Aufteilung macht es möglich, die dadurch kleineren und gekapselten Teilprobleme unter dem Einsatz wiederverwendbarer Expertenmodule gezielt und in weiten Teilen unabhängig voneinander lösen zu können. Erst dadurch wird die automatische Planung auch von komplexeren Fertigungsprozessen in einer flexiblen Fertigungszelle beherrschbar.

Einen wichtigen Kernaspekt bei der automatischen Planung stellt das Verständnis über das zu fertigende Bauteil dar. In Kapitel 5 wurde ein allgemeines Format für die einheitliche Beschreibung der zu fertigenden Produkte vorgestellt und zeigte auf, wie daraus planungsrelevante Prozess- und Bauteilinformationen über ein automatisches und erweiterbares Verfahren extrahiert werden können. Damit werden alle Daten, die für die Montage des Produkts relevant sind und aus dem Wissen über das Produkt abgeleitet werden können, in Form eines digitalen Zwillings des Produkts dargestellt; dies bildet die Grundlage für die Planung.

Anhand der Informationen über das Bauteil erfolgt die Planung zunächst aus Sicht der Domäne und den abstrakten Prozessen, die für die Fertigung erforderlich sind. Kapitel 6 zeigte auf, wie die Prozessplanung auf dieser Ebene zunächst ohne Einbeziehung konkreter Aktuatorik funktionieren kann und welche vorgestellten Konzepte essentiell für eine erfolgreiche und effiziente Planung sind. Eine besondere Bedeutung kommt hierbei der automatischen Extraktion von Kollateraleffekten bei, die das Konzept der Strukturanalyse aus Kapitel 5 wiederverwendet, um damit auch während der Planung den tatsächlichen und vollständigen Effekt einzelner Prozessschritte auf das Bauteil und auf die Roboterzelle automatisch zu ermitteln. Dadurch wird eine realistische und aufbauende Planung ermöglicht. Als weiterer, essentieller Bestandteil der Prozessplanung wurde der pessimistische Suchansatz als erfolgversprechende Strategie der Planung vorgestellt, der insbesondere bei komplex zu fertigenden Bauteilen das effiziente und zielgerichtete Finden eines Plans maßgeblich begünstigt.

Anhand eines Plans abstrakter Prozesse werden in einem nachfolgenden Schritt auf Basis eines digitalen Zwillings der Roboterzelle real ausführbare Aktionen identifiziert und in ein konsistentes Gesamt-Programm überführt. Kapitel 7 stellte hierzu die Fertigungsplanung über Fähigkeitenmodule in einem Korridor-geführten Suchansatz vor, die eine fokussierte Planungsstrategie verfolgt und redundante Planungsverläufe, die lediglich Umsortierungen von Einzelaktionen beschreiben, erkennt und verwirft. Die Strategie leistet daher – ohne dabei den Lösungsraum nennenswert einzuschränken – einen substantiellen Beitrag zur Reduktion der Planungskomplexität. Selbst nicht-

triviale Kooperationen von Robotern, die für eine erfolgreiche Montage notwendig sind, können damit identifiziert und angewandt werden. Als weiteres Ergebnis wurde ein in die Planung verwobener ressourcenbasierter Ansatz zur Parallelisierung von Plänen vorgestellt, der eine nochmals deutliche Zeitersparnis und Effizienzsteigerung bei der Ausführung des Montageplans in einer flexiblen Fertigungsanlage erzielt.

Ein zentraler Baustein bei der Offline-Programmierung von Roboterapplikationen ist der Transfer von einem idealen Plan zu einer präzisen Ausführung in einer realen Roboterzelle. Kapitel 8 beleuchtete hierzu verschiedene Optionen sowie deren Herausforderungen und stellte die in $PaRTs$ angewandten Konzepte zur Abstimmung des Zellenmodells sowohl zur Planungsphase als auch zur Ausführungsphase heraus. Die Möglichkeit zur unmittelbaren Simulation von Plänen erlaubt eine detaillierte und frühzeitige Diagnose des Planungsergebnisses und stellt somit ein wichtiges Werkzeug bei der Automatisierung von Fertigungsprozessen dar. Darüber hinaus wurden zwei Möglichkeiten der realen Ausführung von Planungsergebnissen aufgezeigt: Wenig Aufwand erfordert die Ausführung über die an $PaRTs$ direkt angebundene Robotics API zur echtzeitkonformen Ansteuerung der Roboterzelle. Der in der Industrie oft geforderte Einsatz etablierter Steuerungsrechner wird über ein Generatorenkonzept zur Ableitung von Roboterprogrammen für proprietäre Zielplattformen bedient.

Die in Kapitel 3 eingeführten Fallstudien wurden in Kapitel 9 und 10 wieder aufgegriffen, um die Ergebnisse dieser Arbeit in zwei unterschiedlichen Anwendungsfällen einzusetzen und zu evaluieren. Kapitel 9 beschrieb die Anwendung auf die Domäne LEGO zur Errichtung einer Brücke mit drei Robotern. Das Szenario stellt wegen der Größe und Stabilitätsproblematik der Brücke, der zahlreich zur Verfügung stehenden Fähigkeiten der Roboter sowie der erforderlichen Kooperation bei der Montage eine Planungsaufgabe enormer Komplexität dar. Dank des mehrstufigen Vorgehens, der Deadlockfindungsstrategie und der auf abstrakte Weise beigetragenen Expertenmodule ist sie über $PaRTs$ mit ökonomisch sinnvollem Zeitaufwand lösbar; das Resultat ist ein Ausführungsplan, der sowohl jederzeit die Statik des Bauwerks als auch die dafür benötigten Roboterkooperationen berücksichtigt. In einer zweiten Fallstudie beschrieb Kapitel 10 das robotergestützte Herstellen eines Lagenaufbaus für eine Druckkalotte des Airbus A350 XWB aus carbonfaserverstärktem Kunststoff. Der Hauptfokus dieser Evaluation lag auf der Automatisierung einer Fertigung mit vielen, sich ähnelnden komplexen Prozessschritten, die zudem mit mehreren zur Verfügung stehenden Spezial-Endeffektoren mit unterschiedlichen Wirkprinzipien umsetzbar sind. Aufgrund der abstrakt formulierbaren Fähigkeitenmodule, die für jeden Spezialfall wiederholt und individuell ausgewertet werden können, erzielt $PaRTs$ auch hier universell für unterschiedliche Endeffektoren adäquate Ergebnisse, die bezüglich der Qualität denen einer manuellen Programmierung in Nichts nachstehen. Die Fähigkeitenmodule können auch für die Fertigungsplanung einer abgewandelten oder neuen Version der Druckkalotte wiederverwendet werden, was im Vergleich zur manuellen Programmierung nochmals eine deutliche Zeitersparnis einbringt. Überdies wird von $PaRTs$ der koordinierte Einsatz mehrerer, selbst unterschiedlicher Endeffektoren betrachtet, der zu einer optimalen und, wo möglich, parallelen Ausführung der Prozesse führt.

Insgesamt zeigt die vorliegende Dissertation, dass sich der Einsatz von $PaRTs$ zur Automatisierung verschiedener Fertigungsprozesse und der damit verbundene Initialaufwand für die Bereitstellung der benötigten Expertenmodule lohnt. Das Resultat ist ein Planungssystem, das einerseits die Domäne des zu fertigenden Produkts und andererseits das Modell und die Fähigkeiten der verfügbaren Roboterzelle kennt. Anschließend können Fertigungspläne zur automatisierten Herstellung des Produkts bestimmt werden. Darüber hinaus ist es ebenso möglich, für neue Produkte, die zur selben Domäne gehören, das bestehende Planungssystem wiederzuverwenden, und mit nur minimalem Aufwand auch für diese neuen Produkte eine Fertigungsplanung durchzuführen. Die vorliegende Arbeit zeigt mit $PaRTs$ erstmalig einen allgemeinen Ansatz zur automatischen Planung von Fertigungsprozessen beginnend beim Computermodell des Produkts bis hin zur Ausführung in einer flexiblen Roboterzelle, der sich auf ein Ökosystem unabhängig voneinander konzipierter und universell einsetzbarer Expertenmodule stützt. Im Vergleich zu üblichen Vorgehensweisen bei der Automatisierung von Fertigungsprozessen erlaubt $PaRTs$ eine deutliche Verkürzung der benötigten Entwicklungszeit und bezieht zudem die Parallelisierung von Fertigungsschritten mit mehreren Robotern mit ein. $PaRTs$ ermöglicht dadurch eine effiziente und ökonomische Automatisierung gerade von Produktionen mit hoher Variabilität und geringen Stückzahlen und stellt somit einen vielversprechenden Ansatz für die zukünftige Roboterprogrammierung dar.

11.2 Ausblick

Das Gebiet der automatischen Programmierung ist umfangreich und es existieren zahlreiche noch offene Forschungsfragen in diesem Themenfeld. Der vorgestellte Planungsansatz $PaRTs$ beschreibt zwar ein ganzheitliches und in sich abgeschlossenes Vorgehen zur Automatisierung von Montageprozessen, lässt aber dennoch in vielerlei Hinsicht Potential für weiterführende Forschung.

Um gültige und zerstörungsfreie Ausführungspläne zu erhalten, stellt $PaRTs$ in allen Zwischenschritten der Planung die Kollisionsfreiheit aller Roboter und Bauteile sicher. Für die Kollisionsfreiheit während der Ausführung eines Prozesses ist das Fähigkeitenmodul bislang selbst verantwortlich. Anwender der automatischen Planung würden enorm von einem universell im Planungsansatz eingebetteten Verfahren profitieren, das die Verantwortung der Überprüfung von den einzelnen Fähigkeitenmodulen nimmt und die Kollisionsfreiheit in Gesamtbetrachtung eines Plans insbesondere bei parallelen Abläufen der Montage sicherstellt. Erreichbar wäre dies über einen Algorithmus, der potentiell nebenläufige Prozessausführungen identifiziert und zu diesen jeweils eine Kollisionserkennung auf Basis räumlicher Bewegungshüllen vornimmt. Werden hierbei Kollisionen erkannt, ist der Ausführungsplan entsprechend auszubessern. Hierfür wäre eine neue Art technischer Task (*rekonfigurierbarer Task*) als Baustein für Fähigkeitenmodule vorstellbar, der lediglich das Erreichen eines konkreten Zielpunkts beschreibt (z. B. Transferbewegung), und dessen konkrete Trajektorie generisch und kollisionsfrei über den Ansatz festgelegt werden kann.

Die Planung der Fertigung auf Basis eines Modells der Roboterzelle kann aus vielen Gründen fehlschlagen. Fehlt eine entsprechende Fähigkeit, die für die vorliegende Mon-

tage benötigt wird, oder liegt das zu montierende Einzelteil außerhalb der Erreichbarkeit aller verfügbaren Roboter, kann der Planer keine gültige Lösung finden. Auch weniger offensichtliche Konstellationen können dazu führen, dass die Montageplanung nicht möglich ist: Versetzt man die in Fallstudie 1 eingesetzten Roboter zur Montage der LEGO-Brücke um nur wenige Zentimeter, so ergibt sich sehr schnell die Problematik, dass beim Setzen des oberen Verbindungssteins sowohl die zu unterstützende Bogenhälfte als auch der zu setzende Verbindungsstein im Arbeitsbereich von nur noch einem Roboter liegen. Da beide Aktionen jedoch zeitgleich erfolgen müssen, ist eine Montage im beschriebenen Setting mit den drei Robotern technisch nicht möglich; die Planung terminiert ohne Ergebnis. Die Kombinatorik aller verfügbaren Planungsoptionen führt nicht zu der angestrebten Ziel-Produktsituation. Woran dies liegt, etwa weil eine benötigte Fähigkeit fehlt, der Arbeitsbereich eines Roboters nicht ausreicht oder eine Kooperation unter den gegebenen Umständen nicht umsetzbar ist, bleibt zunächst allerdings unbekannt. Während der Planung kann der Verlauf zwar über die Visualisierung betrachtet werden, doch ist der Grund, warum keine Lösung existiert, auch hieraus nicht trivial zu erkennen, zumal oft immer noch sehr viele planbare Optionen bestehen und untersucht werden. Diese Unwissenheit kann also eine nennenswerte Herausforderung bei der Entwicklung von Expertenmodulen und der Anwendung des Planungsansatzes darstellen. Ein wichtiges und essentielles Werkzeug kann daher die Möglichkeit zum Debuggen von Planungsverläufen sein. Ein möglicher Ansatz wäre, die Anwendbarkeit aller Fähigkeitenmodule in jedem Planungsschritt zu protokollieren und Gründe zu sammeln, weshalb eine Fähigkeit in einer gegebenen Situation nicht eingesetzt werden kann. Terminiert ein Planungslauf ohne Ergebnis, können aus einem solchen Protokoll unter Verwendung geeigneter Auswertungstools ggf. Rückschlüsse auf die vorliegende Ursache gezogen und die Planungsmodalitäten durch den Anwender entsprechend angepasst werden.

Um das Potential flexibler Roboterzellen in Kombination mit dem Planungsansatz $PaRTs$ zu demonstrieren, wäre ein Anwendungsszenario zur Online-Bestellung individueller Produkte über einen Online-Konfigurator denkbar. Ein Kunde hat – z. B. über eine grafische Oberfläche – die Möglichkeit, ein Produkt zu konfigurieren und individuell nach seinen Vorstellungen zu gestalten. Schließt er den Kauf ab, wird die Bestellung digital an die intelligente Fabrik übermittelt. Dort wird die Fertigung des Unikats automatisiert über $PaRTs$ auf Basis der Expertenmodule geplant und in der flexiblen Roboterzelle ausgeführt. In einer weiterführenden Arbeit könnte ein solcher Demonstrator mit einer geeigneten Artikelpalette erarbeitet werden, der zur Machbarkeitsanalyse der individuellen „On-Demand“-Produktion variabler Erzeugnisse herangezogen werden kann. In Zukunft mag eine solche Art der individuellen Produktkonfiguration dank innovativer Produktionstechnologien durchaus eine gängige Einkaufsmöglichkeit im personalisierten Onlinehandel darstellen.

Literaturverzeichnis

[1] E. Abele, K. Haddadian, F. Hähn, D. Andrecht, B. Luckas, and B. Schmidt. Roboterzelle zur feinbearbeitung großer werkzeuge. *wt Werkstattstechnik online*, 06 2015.

[2] A. Angerer. *Object-oriented Software for Industrial Robots*. Dissertation, University of Augsburg, 03 2014.

[3] A. Angerer, C. Ehinger, A. Hoffmann, W. Reif, G. Reinhart, and G. Strasser. Automated cutting and handling of carbon fiber fabrics in aerospace industries. In *Proc. 6th IEEE Conf. on Autom. Science and Engineering, CASE 2010*, pages 861–866. IEEE, 08 2010.

[4] A. Angerer, C. Ehinger, A. Hoffmann, W. Reif, and G. Reinhart. Design of an automation system for preforming processes in aerospace industries. In *2011 IEEE International Conference on Automation Science and Engineering*, pages 557–562, Aug 2011. doi: 10.1109/CASE.2011.6042411.

[5] A. Angerer, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif. Robotics API: Object-oriented software development for industrial robots. *J. of Softw. Eng. for Robotics*, 4(1):1–22, 2013.

[6] A. Angerer., M. Vistein., A. Hoffmann., W. Reif., F. Krebs., and M. Schönheits. Towards multi-functional robot-based automation systems. In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,*, pages 438–443. INSTICC, SciTePress, 2015. ISBN 978-989-758-123-6. doi: 10.5220/0005573804380443.

[7] A. Angerer, A. Hoffmann, L. Larsen, M. V. anf J. Kim, M. Kupke, and W. Reif. Planning and execution of collision-free multi-robot trajectories in industrial applications. In *Proc. 47th Intl. Symp. on Robotics, ISR 2016*. VDE Verlag, 2016.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL `http://www.sciencedirect.com/science/article/pii/S0921889008001772`.

[9] Aus: Paolo Friz. *Ein Weiser, ein Kaiser und viel Reis*. Atlantis, ein Imprint von Orell Füssli Verlag, `www.ofv.ch` ©2017 Orell Füssli Sicherheitsdruck AG, Zürich, 2017. ISBN 978-3-7152-0724-7.

[10] AVK – Industrievereinigung Verstärkte Kunststoffe e.V. *Handbuch Faserverbundkunststoffe/Composites: Grundlagen, Verarbeitung, Anwendungen*. Springer Fachmedien Wiesbaden, 01 2014. ISBN 978-3-658-02754-4. doi: 10.1007/978-3-658-02755-1. URL `https://books.google.de/books?id=mnskBAAAQBAJ`.

[11] R. Awad and M. Naumann. Roboter auch für anspruchsvolle Prozesse. Fachmedium Computer & Automation (WEKA FACHMEDIEN GmbH), 09 2017. URL `https://www.computer-automation.de/feldebene/robotik/roboter-auch-fuer-anspruchsvolle-prozesse.145178.html`. Abgerufen: 23. Mai 2020.

[12] A. A. Baker, S. Dutton, and D. Kelly. *Composite Materials for Aircraft Structures*. Amer. Inst. of Aeronautics & Astronautics, 2004.

[13] J. Banks, editor. *Handbook of Simulation*. John Wiley & Sons, Inc., 2007. ISBN 9780470172445.

[14] Bayerische Motoren Werke Aktiengesellschaft. Wie ein BMW i entsteht. https://www.bmwgroup-werke.com/de/produktion/bmw-i.html. Abgerufen: 06. Mai 2020.

[15] Prof. Dr. E. Behrends. Schachbrett Reiskorn Exponentielles Wachstum. https://www.youtube.com/watch?v=jWXLNPrVhfw, 2015. Abgerufen: 29. Mai 2020.

[16] Benedikt Müller. Gigaset: Smartphones aus der deutschen Provinz. Süddeutsche Zeitung Digital, 07 2018. URL https://www.sueddeutsche.de/digital/gigaset-smartphone-deutschland-1.4041591. Abgerufen: 23. Mai 2020.

[17] J. Benton, A. Coles, and A. Coles. Temporal planning with preferences and time-dependent continuous costs. In *Proc. 22nd Intl. Conf. on Autom. Planning & Scheduling, ICAPS*, 2012.

[18] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi. Motion planning with complex goals. *IEEE Robotics Automation Magazine*, 18(3):55–64, 9 2011. ISSN 1070-9932. doi: 10.1109/MRA.2011.942115.

[19] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot Programming by Demonstration. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 59, pages 1371–1394. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-23957-4. doi: 10.1007/978-3-540-30301-5_60. URL https://doi.org/10.1007/978-3-540-30301-5_60.

[20] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues. Knowledge for intelligent industrial robots. In *AAAI Spring Symposium: Designing Intelligent Robots*, 2012.

[21] D. Bourne, H. Choset, H. Hu, G. Kantor, C. Niessl, Z. Rubinstein, R. Simmons, and S. Smith. Mobile manufacturing of large structures. In *Proc. 2015 IEEE Intl. Conf on Robotics & Automation, ICRA*, pages 1565–1572, 2015.

[22] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405. IEEE, 2007.

[23] DIN 6789:2013-10. Dokumentationssystematik - Verfälschungssicherheit und Qualitätskriterien für die Freigabe digitaler Produktdaten, 10 2013. URL https://dx.doi.org/10.31030/2060741.

[24] DIN EN ISO 10209:2012-11. Technische Produktdokumentation - Vokabular - Begriffe für technische Zeichnungen, Produktdefinition und verwandte Dokumentation, 11 2012. URL https://dx.doi.org/10.31030/1868609.

[25] F. Dirschmid. Die cfk-karosserie des bmw i8 und deren auslegung. In G. Tecklenburg, editor, *Karosseriebautage Hamburg*, pages 217–231, Wiesbaden, 2014. Springer Fachmedien Wiesbaden. ISBN 978-3-658-05980-4.

[26] DLR_de, Deutsches Zentrum für Luft- und Raumfahrt (DLR). Druckkalotte aus CFK vom A350. https://twitter.com/DLR_de/status/971388828345884683/photo/2, 2018. Abgerufen: 09. Mai 2020.

[27] M. Dogar, A. Spielberg, S. Baker, and D. Rus. Multi-robot grasp planning for sequential assembly operations. In *Proc. 2015 IEEE Intl. Conf on Robotics and Automation (ICRA), Seattle, WA, USA*, pages 193–200, 2015. doi: 10.1109/ICRA.2015.7138999.

[28] Z. Dogmus, E. Erdem, and V. Patoglu. React!: An interactive educational tool for ai planning for robotics. *IEEE Trans on Education*, 58(1):15–24, 2 2015. ISSN 0018-9359. doi: 10.1109/TE.2014.2318678.

[29] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59 – 69, 2014. ISSN 1877-7058. doi: https://doi.org/10.1016/j.proeng.2014.12.098. URL http://www.sciencedirect.com/science/article/pii/S187770581403149X. Modelling of Mechanical and Mechatronic Systems.

[30] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular open robots simulation engine: Morse. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 46–51. IEEE, 2011.

[31] EtherCAT Technology Group. EtherCAT – The Ethernet Fieldbus, 11 2018.

[32] K. Etschberger. *Controller Area Network: Basics, Protocols, Chips and Applications*. IXXAT Press, 2001. ISBN 9783000073762. URL https://books.google.de/books?id=n1TJAAAACAAJ.

[33] Europäische Union. Smerobotics: The european robotics initiative for strengthening the competitiveness of smes in manufacturing by integrating aspects of cognitive systems. CORDIS, cordis.europa.eu, 2012-2016. Programm FP7-ICT.

[34] Y. Fan, J. Luo, and M. Tomizuka. A learning framework for high precision industrial assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 811–817, 2019.

[35] M. Feiler. VARI - Kostengünstiges Verfahren zur Herstellung von großflächigen Luftfahrtbauteilen. In *DGLR Jahrestagung (Fachsitzung Produktionsverfahren), 17. - 20. September 2001, Hamburg*, 2001. URL https://elib.dlr.de/14827/. LIDO-Berichtsjahr=2001,.

[36] S. Feldmann, K. Kernschmidt, and B. Vogel-Heuser. Combining a sysml-based modeling approach and semantic technologies for analyzing change influences in manufacturing plant models. *Procedia Cirp*, 17:451–456, 2014.

[37] R. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971.

[38] FRANKA EMIKA GmbH. Introducing the Franka Emika Robot. https://www.franka.de. Abgerufen: 23. Mai 2020.

[39] R. Frericks and U. Hägele. *Robotik in der Praxis - Kompendium für berufliche Schulen*. Ludwigsburg, Württ, Berlin, Heidelberg, 2014. ISBN 978-3-00-046344-0. URL http://d-nb.info/1052691986.

[40] Y. Gan, X. Dai, and D. Li. Off-line programming techniques for multirobot cooperation system. In *International Journal of Advanced Robotic Systems*, volume 10. InTech Europe, 2013. doi: 10.5772/56506. URL https://doi.org/10.5772/56506.

[41] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 978-0-7167-1045-5.

[42] A. Gaschler, R. P. A. Petrick, M. Giuliani, M. Rickert, and A. Knoll. Kvp: A knowledge of volumes approach to robot task planning. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 202–208, 11 2013. doi: 10.1109/IROS.2013.6696354.

[43] I. Georgievski and M. Aiello. HTN planning: Overview, comparison, and beyond. *Artificial Intelligence*, 222:124–156, 05 2015. doi: 10.1016/j.artint.2015.02.002.

[44] T. Gerngross and D. Nieberl. Automated manufacturing of large, three-dimensional cfrp parts from dry textiles. *CEAS Aeronautical Journal*, 7:241–257, 2 2016. doi: 10.1007/s13272-016-0184-5. URL https://doi.org/10.1007/s13272-016-0184-5.

[45] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 05 2004. ISBN 978-1558608566.

[46] M. Ghallab, D. Nau, and P. Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[47] R. Glück, A. Hoffmann, L. Nägele, A. Schierl, W. Reif, and H. Voggenreiter. Towards a tool-based methodology for developing software for dynamic robot teams. In K. Madani and O. Gusikhin, editors, *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics*, number Volume 2. 2018. ISBN 978-989-758-321-6. doi: 10.5220/0006884806050612. URL `http://www.scitepress.org/PublicationsDetail.aspx?ID=Mwf8m9u6pSI=&t=1`.

[48] M. Gombolay, R. Wilcox, and J. Shah. Fast scheduling of multi-robot teams with temporospatial constraints. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, 6 2013.

[49] J. Haag, T. Mertens, L. Kotte, and S. Kaskel. Investigation on atmospheric plasma surface treatment for structural bonding on titanium and cfrp. In *2015 IEEE International Conference on Plasma Sciences (ICOPS)*, pages 1–1, 2015.

[50] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6251. IEEE, 2018.

[51] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi. Towards manipulation planning with temporal logic specifications. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 346–352, 5 2015. doi: 10.1109/ICRA.2015.7139022.

[52] A. Hoffmann. *Serviceorientierte Automatisierung von Roboterzellen: Modularität und Wiederverwendbarkeit von Software in der Robotik*. Dissertation, Universität Augsburg, 12 2015.

[53] A. Hoffmann, L. Nägele, A. Angerer, A. Schierl, and W. Reif. Using object-oriented development for planning and controlling industrial robot systems. In *Workshop on Recent Advances in Planning & Manipulation for Industrial Robots, Robotics: Science and Systems Conference, Ann Arbor, MI, USA*, 06 2016. URL `https://sites.google.com/site/rss16irt/`.

[54] A. Hoffmann, L. Nägele, and W. Reif. How to find assembly plans (fast) – hierarchical state space partitioning for efficient multi-robot assembly planning. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. 2020.

[55] D. Höller, G. Behnke, P. Bercher, S. Biundo-Stephan, H. Fiorino, D. Pellier, and R. Alford. Hddl: An extension to pddl for expressing hierarchical planning problems. In *AAAI*, 2020.

[56] D. Höller, P. Bercher, G. Behnke, and S. Biundo. Htn planning as heuristic progression search. *Journal of Artificial Intelligence Research*, 67:835–880, 2020.

[57] L. S. Homem de Mello and A. C. Sanderson. And/or graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, 1990.

[58] C. Hopmann and W. Michaeli. *Einführung in die Kunststoffverarbeitung*. Carl Hanser Verlag GmbH & Co KG, M, 2017. ISBN 978-3-446-45356-2.

[59] J. Huckaby, S. Vassos, and H. I. Christensen. Planning with a task modeling framework in manufacturing robotics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 5787–5794. IEEE, 2013.

[60] IEC 61158-6-10:2019. Industrial communication networks - Fieldbus specifications - Part 6-10: Application layer protocol specification - Type 10 elements, 06 2019.

[61] IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015). IEEE Standard for Ethernet, 08 2018. URL https://doi.org/10.1109/IEEESTD.2018.8457469.

[62] IFR Pressemitteilung. Roboter-Investitionen steigen auf Rekord von 16,5 Mrd USD – IFR stellt neuen World Robotics Report vor. International Federation of Robotics (IFR), 09 2019. URL https://ifr.org/downloads/press2018/2019-09-18_Pressemeldung_IFR_World_Robotics_2019_Industrial_Robots_deutsch.pdf.

[63] F. Imeson and S. L. Smith. A language for robot path planning in discrete environments: The tsp with boolean satisfiability constraints. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5772–5777, 5 2014. doi: 10.1109/ICRA.2014.6907707.

[64] F. Imeson and S. L. Smith. Multi-robot task planning and sequencing using the sat-tsp language. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5397–5402, 5 2015. doi: 10.1109/ICRA.2015.7139953.

[65] Y. Jiang, S. Zhang, P. Khandelwal, and P. Stone. Task planning in robotics: an empirical comparison of pddl-based and asp-based systems, 2018.

[66] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1470–1477, 5 2011. doi: 10.1109/ICRA.2011.5980391.

[67] B. Kast, S. Albrecht, W. Feiten, and J. Zhang. Bridging the gap between semantics and control for industry 4.0 and autonomous production. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 780–787, 2019.

[68] B. Kast, V. Dietrich, S. Albrecht, W. Feiten, and J. Zhang. A hierarchical planner based on set-theoretic models: Towards automating the automation for autonomous systems. In O. Gusikhin, K. Madani, and J. Zaytoon, editors, *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2019 - Volume 1, Prague, Czech Republic, July 29-31, 2019*, pages 249–260. SciTePress, 2019. doi: 10.5220/0007840702490260. URL https://doi.org/10.5220/0007840702490260.

[69] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Proc. 2013 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 855–862, 2013. doi: 10.1109/ICRA.2013.6630673.

[70] C. A. Knoblock. Learning abstraction hierarchies for problem solving. In *Nat. Conf. on Artif. Intell.*, pages 923–928, 1990. ISBN 0-262-51057-X.

[71] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[72] KUKA AG. LBR iiwa. https://www.kuka.com/de-de/produkte-leistungen/robotersysteme/industrieroboter/lbr-iiwa, . Abgerufen: 23. Mai 2020.

[73] KUKA AG. Montieren: die Technologie. https://www.kuka.com/de-de/produkte-leistungen/verfahrenstechnologien/2016/07/montieren, . Abgerufen: 23. Mai 2020.

[74] *KUKA.RoboTeam 2.0*. KUKA System Technology, Apr 2013. Version: KST RoboTeam 2.0 V2 de.

[75] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006. ISBN 9780511546877. doi: 10.1017/CBO9780511546877. URL https://doi.org/10.1017/CBO9780511546877.

[76] M. Levihn, L. P. Kaelbling, T. Lozano-Pérez, and M. Stilman. Foresight and reconsideration in hierarchical planning and execution. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 224–231, 11 2013. doi: 10.1109/IROS.2013.6696357.

[77] D. Lu and X. Chen. Interpreting and extracting open knowledge for human-robot interaction. *IEEE/CAA Journal of Automatica Sinica*, 4(4):686–695, 2017.

[78] M. Macho, L. Nägele, A. Hoffmann, A. Angerer, and W. Reif. A flexible architecture for automatically generating robot applications based on expert knowledge. In *Proc. 47th Intl. Symp. on Robotics, ISR 2016, Munich, Germany*. VDE Verlag, 2016.

[79] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. L. McGuinness, E. Sirin, and N. Srinivasan. Bringing semantics to web services with OWL-S. *World Wide Web*, 10(3):243–277, 2007. ISSN 1573-1413. doi: 10.1007/s11280-007-0033-x.

[80] Mathieu Bélanger-Barrette, Robotiq Inc. What is an Average Price for a Collaborative Robot? `https://blog.robotiq.com/what-is-the-price-of-collaborative-robots`, 02 2020. Abgerufen: 28. Mai 2020.

[81] Matthias Beyrle, Deutsches Zentrum für Luft- und Raumfahrt (DLR). AZIMUT – Automatisierung zukunftsweisender industrieller Methoden und Technologien für CFK-Rümpfe. `https://www.dlr.de/bt/en/desktopdefault.aspx/tabid-8381/14345_read-36200/`, 2020. Abgerufen: 08. Mai 2020.

[82] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - the planning domain definition language. Technical Report TR 98 003/DCS TR 1165, Yale Center for Computational Vision and Control, 10 1998.

[83] W. Meeussen, J. De Schutter, E. Staffetti, J. Xiao, and H. Bruyninckx. Integration of planning and execution in force controlled compliant motion. *Robotics and Autonomous Systems*, 5 2008.

[84] M. Merdan, E. List, and W. Lepuschitz. Knowledge-driven industrial robotics for flexible production. In *2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES)*, pages 000225–000230, 2017.

[85] n-tv.de. Lego-Alarm überführt Dieb: Oma und Enkel auf Zack. n-tv Nachrichtenfernsehen GmbH, 08 2002. URL `https://www.n-tv.de/archiv/Oma-und-Enkel-auf-Zack-article122872.html`. Abgerufen: 07. Januar 2020.

[86] L. Nägele, M. Macho, A. Angerer, A. Hoffmann, M. Vistein, M. Schönheits, and W. Reif. A backward-oriented approach for offline programming of complex manufacturing tasks. In *Proc. 6th Intl. Conf. on Automation, Robotics and Applications, ICARA 2015, Queenstown, New Zealand*, pages 124–130. IEEE, 2015.

[87] L. Nägele, A. Schierl, A. Hoffmann, and W. Reif. Automatic planning of manufacturing processes using spatial construction plan analysis and extensible heuristic search. In K. Madani and O. Gusikhin, editors, *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics*, number Volume 2. 2018. ISBN 978-989-758-321-6. doi: 10.5220/0006861705860593. URL `http://www.scitepress.org/PublicationsDetail.aspx?ID=XmYAzaKOtMk=&t=1`.

[88] L. Nägele, A. Schierl, A. Hoffmann, and W. Reif. Modular and domain-guided multi-robot planning for assembly processes. In O. Gusikhin, K. Madani, and J. Zaytoon, editors, *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, July 29-31, 2019, in Prague, Czech Republic*, number Volume 2, pages 595–604. SciTePress, 2019. ISBN 978-989-758-380-3. doi: 10.5220/0007977205950604.

[89] L. Nägele, A. Hoffmann, A. Schierl, and W. Reif. LegoBot: Automated planning for coordinated multi-robot assembly of LEGO structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[90] L. Nägele, A. Schierl, A. Hoffmann, and W. Reif. Multi-Robot Cooperation for Assembly: Automated Planning and Optimization. In O. Gusikhin, K. Madani, and J. Zaytoon, editors, *Informatics in Control, Automation and Robotics - 16th International Conference, ICINCO 2019, Prague, Czech Republic, July 29-31, 2019, Revised Selected Papers*, Lecture Notes in Electrical Engineering (LNEE). Springer, 2021. ISBN 978-3-030-63192-5. doi: 10.1007/978-3-030-63193-2.

[91] M. Neitzel, P. Mitschang, and U. Breuer. *Handbuch Verbundwerkstoffe - Werkstoffe, Verarbeitung, Anwendung*. Carl Hanser Verlag GmbH & Co KG, M, 2014. ISBN 978-3-446-43697-8.

[92] K. Nottensteiner, T. Bodenmueller, M. Kassecker, M. A. Roa, A. Stemmer, T. Stouraitis, D. Seidel, and U. Thomas. A complete automated chain for flexible assembly using recognition, planning and sensor-based execution. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–8, 2016.

[93] A. Perzylo, N. Somani, M. Rickert, and A. Knoll. An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4197–4203, 2015.

[94] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll. Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2293–2300, 2016.

[95] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. Rath Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. Gestegard Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kessler, and M. Danzer. Smerobotics: Smart robots for flexible manufacturing. *IEEE Robotics Automation Magazine*, 26(1):78–90, 2019.

[96] J. Pfrommer, M. Schleipen, and J. Beyerer. PPRS: production skills and their relation to product, process, and resource. In *Proc. 2013 IEEE 18th Conf. on Emerging Technologies & Factory Automation, ETFA 2013, Cagliari, Italy*. IEEE, 2013.

[97] J. Pfrommer, D. Stogl, K. Aleksandrov, S. E. Navarro, B. Hein, and J. Beyerer. Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. *Automatisierungstechnik*, 63(10):790–800, 2015.

[98] S. Profanter, A. Breitkreuz, M. Rickert, and A. Knoll. A hardware-agnostic opc ua skill model for robot manipulators and tools. In *Proceedings of the IEEE International Conference on Emerging Technologies And Factory Automation (ETFA)*, Zaragoza, Spain, Sept. 2019.

[99] M. F. F. Rashid, W. Hutabarat, and A. Tiwari. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *International Journal of Advanced Manufacturing Technology*, 59:335–349, 2011.

[100] Rethink Robotics GmbH. Sawyer, the high performance collaborative robot. `https://www.rethinkrobotics.com`. Abgerufen: 23. Mai 2020.

[101] I. Rodrıguez, K. Nottensteiner, D. Leidner, M. Kaßecker, F. Stulp, and A. Albu-Schäffer. Iteratively refined feasibility checks in robotic assembly sequence planning. *IEEE Robotics and Automation Letters*, 4(2):1416–1423, 2019.

[102] E. Rohmer, S. P. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework.

[103] E. Rohmer, S. P. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1321–1326. IEEE, 2013. doi: 10.1109/IROS.2013.6696520.

[104] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2010. ISBN 978-0136042594.

[105] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. *ai*, 5(2):115–135, 1974.

[106] A. Schierl, A. Hoffmann, L. Nägele, and W. Reif. Integrating reactive behavior and planning: optimizing execution time through predictive preparation of state machine tasks. In E. Y. T. Chuo, D. Brugali, T. Fukuda, J.-C. Latombe, P. C.-Y. Sheu, and J. J. P. Tsai, editors, *2018 Second IEEE International Conference on Robotic Computing (IRC), 31 January - 2 February 2018, Laguna Hills, CA, USA*. 2018. ISBN 9781538646526. doi: 10.1109/irc.2018.00022.

[107] A. Schierl, A. Hoffmann, L. Nägele, and W. Reif. Integrating planning and reactive behavior by using semantically annotated robot tasks. *Encyclopedia with Semantic Computing and Robotic Intelligence*, 2(1):1850005, 2018. doi: 10.1142/s2529737618500053.

[108] M. Schleipen, B. Hein, J. Pfrommer, K. Aleksandrov, D. Štogl, S. Navarro, and J. Beyerer. Automationml to describe skills of production plants based on the ppr concept. 10 2014.

[109] T. R. Schoen and D. Rus. Decentralized robotic assembly with physical ordering and timing constraints. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5764–5771, 2013.

[110] A. Skoglund, B. Iliev, B. Kadmiry, and R. Palm. Programming by demonstration of pick-and-place tasks for industrial manipulators using task primitives. In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, pages 368–373, June 2007. doi: 10.1109/CIRA.2007.382863.

[111] N. Somani, M. Rickert, and A. Knoll. An exact solver for geometric constraints with inequalities. *IEEE Robotics and Automation Letters*, 2(2):1148–1155, 2017.

[112] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 5 2014. doi: 10.1109/ICRA.2014.6906922.

[113] F. Steinmetz, A. Wollschläger, and R. Weitschat. Razer—a hri for visual task-level programming and intuitive skill parameterization. *IEEE Robotics and Automation Letters*, 3(3): 1362–1369, 2018.

[114] M. Stenmark and J. Malec. Describing constraint-based assembly tasks in unstructured natural language. In P. Korondi, editor, *Proc. IFAC 2014 World Congress, Capetown, South Africa*, 2014.

[115] M. Stenmark and J. Malec. Knowledge-based instruction of manipulation tasks for industrial robotics. *Robotics and Computer-Integrated Manufacturing*, 33:56–67, 2015. ISSN 0736-5845.

[116] M. Stenmark, M. Haaae, and E. A. Topp. Simplified programming of re-usable skills on a safe industrial robot - prototype and evaluation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 463–472, 2017.

[117] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham. Can robots assemble an ikea chair? *Science Robotics*, 3(17), 2018. doi: 10.1126/scirobotics.aat6385. URL https://robotics.sciencemag.org/content/3/17/eaat6385.

[118] Technische Universität München. Schachbrett und Reiskörner. http://www-hm.ma.tum.de/ws1213/lba1/erg/erg07.pdf, 2012. Abgerufen: 28. Mai 2020.

[119] U. Thomas, T. Stouraitis, and M. A. Roa. Flexible assembly through integrated assembly sequence planning and grasp planning. In *Proc. 2015 IEEE Intl. Conf. on Automation Science & Engineering (CASE)*, pages 586–592, 2015. doi: 10.1109/CoASE.2015.7294142.

[120] F. H. Tseng, T. T. Liang, C. H. Lee, L. D. Chou, and H. C. Chao. A star search algorithm for civil uav path planning with 3g communication. In *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 942–945, 2014.

[121] VDI/VDE 3682 Blatt 2:2015-05. Formalisierte Prozessbeschreibungen - Informationsmodell, 05 2015.

[122] M. Vistein. *Embedding Real-Time Critical Robotics Applications in an Object-Oriented Language*. Dissertation, Universität Augsburg, 05 2015.

[123] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Gálvez-López, K. Häussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schießle, M. Tenorth, O. Zweigle, and R. V. De Molengraft. Roboearth. *IEEE Robotics Automation Magazine*, 18(2):69–82, 2011.

[124] C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, M. Cao, X. Ou, X. Su, W. Li, Z. Lu, M. Li, Y. Wang, J. Long, M. Huang, Y. Li, and Q. Wang. Path planning of automated guided vehicles based on improved a-star algorithm. In *2015 IEEE International Conference on Information and Automation*, pages 2071–2076, 2015.

[125] M. Weser, D. Off, and J. Zhang. Htn robot planning in partially observable dynamic environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 1505–1510, 2010.

[126] F. Wildgrube, A. Perzylo, M. Rickert, and A. Knoll. Semantic mates: Intuitive geometric constraints for efficient assembly specifications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macao, China, Nov. 2019. URL https://youtu.be/o5EiAut3N2c.

[127] J. Wolfe, B. Marthi, and S. Russell. Combined task and motion planning for mobile manipulation. In *20th Intl. Conf. on Automated Planning and Scheduling*, pages 254–257. AAAI Press, 2010. URL http://dl.acm.org/citation.cfm?id=3037334.3037373.

[128] xenomai.org. Xenomai. https://xenomai.org. Abgerufen: 20. April 2020.

[129] J. Xiao and X. Ji. Automatic generation of high-level contact state space. *Intl. J. of Rob. Research*, 20(7):584–606, 2001.

[130] Z. Yan, N. Jouandeau, and A. A. Cherif. A survey and analysis of multi-robot coordination. 10(12):399, 2013. doi: 10.5772/57313.

[131] S. Yun and D. Rus. Adaptive coordinating construction of truss structures using distributed equal-mass partitioning. *IEEE Trans. on Robotics*, 30(1):188–202, 2 2014. ISSN 1552-3098. doi: 10.1109/TRO.2013.2279643.

[132] Z. Zhang and Z. Zhao. A multiple mobile robots path planning algorithm based on a-star and dijkstra algorithm. *International Journal of Smart Home*, 8:75–86, 2014.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Stichwortverzeichnis

Betreute Studien- und Abschlussarbeiten

- Thomas Eibel. „Interpretation und Bearbeitung von Roboteraufgaben in der Offline-Programmierung“. Bachelorarbeit. Universität Augsburg, 2013
- Adrian Sezeleanu. „A Concept for UAV Obstacle Avoidance Based on Sensor Fusion“. Masterarbeit. Universität Augsburg, 2013
- Benjamin Schwartz. „A visual language for specifying quadrotor missions“. Masterarbeit. Universität Augsburg, 2014
- Peter Wagner. „Konzeption und Umsetzung einer Navigationsschnittstelle für eine 3D-Robotersimulation“. Bachelorarbeit. Universität Augsburg, 2016
- Lev Sorokin. „Algorithmen zur automatisierten Handhabung von biegeschlaffen Faser-Halbzeugen in 2D“. Masterarbeit. Universität Augsburg, 09. Aug. 2016
- Stefan Lindner. „Konzepte zur Steuerung von Robotern und Roboterteams mit Force-Feedback-Steuerungen“. Masterarbeit. Universität Augsburg, 2016
- Markus Pöschl. „Erstellung einer Android Bridge zur automatischen Übernahme von Änderungen im Robotics API Quellcode“. Projektmodul. Universität Augsburg, 2015
- Matthias Stüben. „Robust and Real-Time Matching and Tracking of Flowers for an Agricultural Robot Platform“. Masterarbeit. Universität Augsburg, 2017
- Lukas Vogel. „Integrating Robotic Software Frameworks: A Case Study Using the Robot Manager and the Robotics API“. Masterarbeit. Universität Augsburg, 2018
- Johannes Tintenherr. „Automatische Planung von robotergestützten Montageprozessen mit PDDL“. Bachelorarbeit. Universität Augsburg, 2019
- Luitpold Reiser. „Swept Volume Approximation for Multi-Robot Collision Estimation“. Masterarbeit. Universität Augsburg, 2020
- Jonas Geschke. „Kollisionsvermeidung und Ausweichstrategien in der Industrierobotik“. Projektmodul. Universität Augsburg, 2020

Eigene Publikationen

1. Ludwig Nägele, Andreas Schierl, Alwin Hoffmann and Wolfgang Reif (2021). „Multi-Robot Cooperation for Assembly: Automated Planning and Optimization“. Informatics in Control, Automation and Robotics, 2019 Revised Selected Papers, ser. Lecture Notes in Electrical Engineering: LNEE. Springer Verlag. ISBN 978-3-030-63192-5. DOI: 10.1007/978-3-030-63193-2.

2. Ludwig Nägele, Alwin Hoffmann, Andreas Schierl and Wolfgang Reif (2020). „LegoBot: Automated Planning for Coordinated Multi-Robot Assembly of LEGO structures“. IEEE/RSJ International Conference on Intelligent Robots and Systems: IROS.

3. „Intelligente Roboter sollen Produktmontage vereinfachen“ (2020, Juli 16). Augsburger Allgemeine, in der Beilage Wissenschaft und Forschung in Augsburg, Ausgabe 15, Seite 2. Auflage: 325.000.

4. Alwin Hoffmann, Ludwig Nägele and Wolfgang Reif (2020). „How to find assembly plans (fast) – Hierarchical state space partitioning for efficient multi-robot assembly planning“. IEEE International Conference on Robotic Computing: IRC.

5. Ludwig Nägele, Andreas Schierl, Alwin Hoffmann and Wolfgang Reif (2019). „Modular and Domain-Guided Multi-Robot Planning for Assembly Processes“. In Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics: ICINCO. DOI: 10.5220/0007977205950604.

6. Ludwig Nägele, Andreas Schierl, Alwin Hoffmann and Wolfgang Reif (2018). „Automatic Planning of Manufacturing Processes using Spatial Construction Plan Analysis and Extensible Heuristic Search“. In Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO, pages 576-583. ISBN 978-989-758-321-6. DOI: 10.5220/0006861705760583.

7. Roland Glück, Alwin Hoffmann, Ludwig Nägele, Andreas Schierl, Wolfgang Reif and Heinz Voggenreiter (2018). „Towards a Tool-Based Methodology for Developing Software for Dynamic Robot Teams“. In Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO, pages 605-612. ISBN 978-989-758-321-6. DOI: 10.5220/0006884806050612.

8. Andreas Schierl, Alwin Hoffmann, Ludwig Nägele and Wolfgang Reif (2018). „Integrating planning and reactive behavior by using semantically annotated robot tasks“. Encyclopedia with Semantic Computing and Robotic Intelligence - Volume 2. DOI: 10.1142/s2529737618500053.

9. Andreas Schierl, Alwin Hoffmann, Ludwig Nägele and Wolfgang Reif (2018). „Integrating Reactive Behavior and Planning: Optimizing Execution Time through Predictive Preparation of State Machine Tasks“. Proceedings of 2nd IEEE International Conference on Robotic Computing: IRC, pages 95-101. DOI: 10.1109/irc.2018.00022.

10. Miroslav Macho, Ludwig Nägele, Dr. Alwin Hoffmann, Dr. Andreas Angerer and Prof. Dr. Wolfgang Reif (2016). „A Flexible Architecture for Automatically Generating Robot Applications based on Expert Knowledge“. Proceedings of 47st International Symposium on Robotics: ISR. VDE-Verlag Berlin.
11. Alwin Hoffmann, Ludwig Nägele, Andreas Angerer, Andreas Schierl, Wolfgang Reif (2016). „Using Object-Oriented Development for Planning and Controlling Industrial Robot Systems“. Robotics: Science and Systems 2016 Workshop on Recent Advances in Planning and Manipulation for Industrial Robots.
12. Ludwig Nägele, Miroslav Macho, Andreas Angerer, Alwin Hoffmann, Michael Vistein, Manfred Schönheits and Prof. Dr. Wolfgang Reif (2015). „A backward-oriented approach for offline programming of complex manufacturing tasks“. Proceedings of 6th International Conference on Automation, Robotics and Applications: ICARA, pages 124-130. DOI: 10.1109/icara.2015.7081135.
13. Benjamin Schwartz, Ludwig Nägele, Andreas Angerer and Bruce A. MacDonald (2014). „Towards a graphical language for quadrotor missions“. Proceedings of the Fifth International Workshop on Domain-Specific Languages and Models for Robotic Systems: DSLRob.
14. Ludwig Nägele, Andreas Angerer and Bruce A. MacDonald (2013). „Graphical formalization and automated computing of safety constraints in robotics“. Eighth full-day Workshop on Software Development and Integration in Robotics: SDIR VIII.