Semantische Modellierung automatisierter Produktionssysteme zur Verbesserung der IT-Integration zwischen Anlagen-Engineering und Steuerungsebene

Der Technischen Fakultät der
Friedrich-Alexander-Universität Erlangen-Nürnberg
zur
Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von

Jochen Merhof
aus Erlangen

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

Tag der mündlichen Prüfung:	12.11.2015
Vorsitzender des Promotionsorgans:	Prof. Dr. Peter Greil
Gutachter:	Prof. Dr.-Ing. Jörg Franke
	Prof. Dr.-Ing. Dietmar Fey

Vorwort und Danksagung

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik (FAPS) der Friedrich-Alexander-Universität Erlangen-Nürnberg.

Ganz herzlich möchte ich Prof. Dr.-Ing. Jörg Franke sowie dem ehemaligen Lehrstuhlinhaber Prof. Dr.-Ing. Klaus Feldmann für die stete Förderung, konstruktive Anregungen sowie dem mir gewährten wissenschaftlichen Freiraum danken.

Ebenfalls möchte ich mich bei Prof. Dr.-Ing. Dietmar Fey, Leiter des Lehrstuhls für Informatik 3 (Rechnerarchitektur) der Friedrich-Alexander-Universität Erlangen-Nürnberg für die Übernahme des Korreferats als auch bei Prof. Dr.-Ing. Lothar Pfitzner, Lehrstuhl für Elektronische Bauelemente der Friedrich-Alexander-Universität Erlangen-Nürnberg als weiteres Mitglied des Prüfungskollegiums bedanken.

Da ein wesentlicher Teil meiner wissenschaftlichen Arbeit in enger Zusammenarbeit mit der Vorfeldentwicklung „Advanced Technologies and Standards (ATS)“ von Siemens entstand, möchte ich ebenfalls Herrn Dr.-Ing. Anton Friedl (ehem. Direktor ATS 1, System Architecture & Platforms) sowie den Mitarbeitern der ATS für die sehr gute Zusammenarbeit meinen Dank aussprechen.

Für die prompte Unterstützung bei administrativen, technischen und kaufmännischen Aufgaben sowie für die stets ausgezeichnete Zusammenarbeit in verschiedenen Projekten möchte ich allen Mitarbeiterinnen und Mitarbeitern des Lehrstuhl FAPS herzlich danken! Das gute Arbeitsklima war immer motivierend.

Mein persönlicher Dank gilt meinen Eltern, meiner Schwester, meiner Frau und meinem Sohn, die mich stets unterstützt und den erforderlichen Rückhalt für die Durchführung des Promotionsvorhabens geboten haben.

Erlangen, im November 2015 Jochen Merhof

Inhaltsverzeichnis

1 Einleitung

Kontinuierliche Veränderungen im Bereich der industriellen Produktion stellen Unternehmen vor große Herausforderungen. Die zunehmende Globalisierung führt zu einem verstärkten internationalen Wettbewerb, bei dem sich Unternehmen in den Hochlohnländern, insbesondere durch komplexe Produkte in hoher Qualität bei großer Variantenvielfalt, vergleichsweise geringen Stückzahlen und kurzen Lieferzeiten, behaupten können. Auch im industriellen Umfeld verkürzen sich die Produkt-Lebenszyklen stetig, was zu einer weiteren Verstärkung des Wettbewerbsdrucks führt. [1]

Um die genannten Produktanforderungen erfüllen zu können, sind innovative Produktionsprozesse erforderlich. Diese müssen einerseits flexibel für verschiedene Produktvarianten einsetzbar sein, andererseits hohe Anforderungen bezüglich Qualität und Reproduzierbarkeit erfüllen. Des Weiteren sollen die Ausfallzeiten der Maschinen aufgrund von Störungen und Wartungsmaßnahmen möglichst gering gehalten werden.

Dies hat unmittelbare Auswirkungen auf die erforderlichen automatisierten Produktionssysteme. Um die zunehmend komplexeren Prozesse zu überwachen und zu steuern, ist in den Produktionssystemen ein vermehrter Einsatz von Sensorik notwendig, um den Prozesszustand zu erfassen. Dies bildet die Datengrundlage für Prozessanpassungen mittels Steuerungstechnik unter Verwendung der eingesetzten Aktuatorik, um den Prozess in einem bestimmten Prozessfenster bezüglich bestimmter Messwerte zu halten. Auf Steuerungsebene zeichnen sich durch diesen Trend steigende Aufwände in den Bereichen Kommunikation und Datenverarbeitung sowie bei der Projektierung der Steuerungstechnik ab, da die zunehmend komplexeren Automatisierungsszenarien in einem Steuerungsprogramm zum Beispiel einer Speicherprogrammierbaren Steuerung abgebildet werden müssen.

Aus organisatorischer Sicht nimmt die Komplexität ebenfalls zu. Die erfolgreiche Bearbeitung von Projekten im Bereich des Anlagenbaus automatisierter Produktionssysteme wird zunehmend schwieriger, da die Integration von verschiedenen Fachkompetenzen aus unterschiedlichen Domänen unter Verwendung von geeigneten Software-Werkzeugen erforderlich ist. Dennoch werden die vorgesehenen Bearbeitungszeiten der einzelnen Projektphasen immer kürzer, wodurch immer weniger Zeit zur Absicherung der Ergebnisse zur Verfügung steht. Dadurch zeigt sich oftmals erst während der Inbetriebnahme-Phase, welcher Aufwand zur Korrektur von Fehlern vorangegangener Engineering-Phasen anfällt. Neben einem zusätzlichen monetären Aufwand ist auch der zusätzliche Zeitbedarf ein erhebliches Problem, da dies zu Produktionsausfällen und einem Image-Schaden führen kann. Da hier erhebliche ungeplante Kosten entstehen können, ist die Inbetriebnahme-Phase ein schwer vorhersagbarer Kostenfaktor, der über eine gewinnbringende oder verlustbehaftete Projektabwicklung entscheiden kann.

Ein grundlegendes Ziel ist die Reduktion des erforderlichen Aufwands, um automatisierte Produktionssysteme produktionsbereit zur Verfügung zu stellen. Dies gilt sowohl für Neuentwicklungen als auch für Umbau- bzw. Umrüstungsmaßnahmen von Produktionssystemen während des Anlagen-Life-Cycles. Insbesondere bei neuen, innovativen Produkten ist eine möglichst kurze Zeit bis zur Marktverfügbarkeit (time to market) ein wichtiger Erfolgsfaktor als Differenzierungsmerkmal gegenüber konkurrierenden Unternehmen in einem internationalen Wettbewerbsumfeld. Aus diesem Grund müssen die Automatisierungshersteller Hard- und Softwarelösungen zur Verfügung stellen, die die zunehmende Komplexität der Produktionssysteme während der Planung, der Inbetriebnahme und dem Betrieb besser unterstützen [2]. Um dabei ein reibungsloses Zusammenspiel gewährleisten zu können, ist eine enge Verzahnung der Softwarewerkzeuge untereinander sowie ein stufenloser Übergang zwischen den Software-Produkten der digitalen Fabrik und dem realen Produktionssystem erforderlich [3]. Diese beiden Punkte sind wichtige Faktoren zur erfolgreichen Realisierung der zunehmend geforderten Flexibilität [4; 5] und Wandlungsfähigkeit [6] von Produktionssystemen, um Produkt- bzw. Variantenwechsel schneller und effizienter realisieren zu können.

Nach heutigem Stand der Technik besteht oftmals eine starke Diskrepanz zwischen dem aktuellen Zustand eines realen Produktionssystems bezüglich Mechanik, Elektrik und Software und dem letzten vorhandenen digitalen Planungsstand, was ein Hemmnis für eine Wandlungsfähigkeit während des Lebenszyklus der Anlage darstellt. Heutige automatisierte Produktionssysteme fokussieren sich bezüglich Wandlungsfähigkeit überwiegend auf mechanische Aspekte, um zum Beispiel durch einen modularen mechanischen Aufbau Umbauarbeiten an einer Anlage schneller realisieren zu können. Weitere Aspekte wie zum Beispiel Elektrik und Software werden oftmals zu spät berücksichtigt, obwohl diese einen erheblichen Einfluss auf die erzielbare Wandlungsfähigkeit haben. Es ist daher erforderlich, dass die verschiedenen Domänen der Mechatronik während der Projektierung eines automatisierten Produktionssystems differenziert betrachtet und beschrieben sowie zu einer Gesamtlösung integriert werden können. Außerdem ist eine Verbesserung des Informationsaustauschs zwischen Engineering-Applikationen erforderlich, um eine reibungslose Zusammenarbeit verschiedener Engineering-Werkzeuge (horizontale IT-Integration) während des Entwicklungsprozesses zu gewährleisten. Des Weiteren ist eine Verbesserung der Interaktion zwischen Engineering-Werkzeugen und der Steuerungsebene des realen Produktionssystems (vertikale IT-Integration) notwendig, um die Effizienz bei Inbetriebnahme, Betrieb und Wartung zu steigern. Sowohl bei der horizontalen als auch vertikalen IT-Integration spielen semantische Beschreibungen eine wesentliche Rolle, um einen Informationsaustausch zwischen verschiedenen Engineering-Werkzeugen als auch zwischen Engineering-Werkzeugen und Steuerungsebene vornehmen zu können.

Aufbau und Struktur der Arbeit

Die Inhalte dieser Arbeit gliedern sich in sechs Kapitel, die für einen besseren Überblick nachfolgend kurz zusammengefasst werden.

- Kapitel 1: In diesem Kapitel wird einleitend die Ausgangssituation bei produzierenden Unternehmen erläutert sowie zum Handlungsbedarf übergeleitet.
- Kapitel 2: Im Umfeld dieser Arbeit existieren Standards und Normen aus verschiedenen Bereichen, die zur Orientierung sowie für eine erfolgreiche Realisierung hilfreich sind. Dabei werden zum Beispiel generelle Konzepte und Methoden zur Entwicklung von Lösungsansätzen insbesondere auch mit Bezug zur Lösungsfindung beim Entwickeln und Konstruieren erläutert. Darüber hinaus werden allgemeine Grundlagen zur Modellbildung dargestellt, um Systeme bezüglich ausgewählter Aspekte zu beschreiben. Daran anknüpfend werden verschiedene Beschreibungsstandards vorgestellt, die zur Beschreibung bzw. Modellbildung im Kontext automatisierter Produktionssysteme eingesetzt werden können. Nachfolgend werden ausgewählte Verhaltensmodelle zur Beschreibung des dynamischen Systemverhaltens beschrieben. Daran anknüpfend sind Standards zur Projektierung von speicherprogrammierbaren Steuerungen exzerpiert. Des Weiteren werden die vorgestellten Standards diskutiert, eine Abgrenzung zu anderen Forschungsarbeiten sowie eine Einordnung in den Kontext dieser Arbeit vorgenommen. Der Handlungsbedarf zur Verbesserung der IT-Integration zwischen dem Engineering von Produktionssystemen und der Steuerungsebene, die Zieldefinition dieser Arbeit sowie der abgeleitete Lösungsansatz sind anschließend dargestellt.
- Kapitel 3: In diesem Kapitel wird ein Konzept zur semantischen Beschreibung eines Produktionssystems vorgestellt. Es wird ein adaptiertes Konzept erläutert, das sich für eine prototypische Umsetzung eignet. Dieses umfasst die Beschreibung von Struktur und Verhalten eines modularen, komponentenbasierten Automatisierungssystems mit Hilfe von semantisch definierten Elementen und Referenzen. Dieses Konzept wird ebenfalls mit Hilfe von UML-Klassendiagrammen der verschiedenen Domänen beispielhaft dargestellt. Dies spannt die Komplexität für eine semantische Beschreibung von Struktur und Verhalten einer Automatisierungslösung aus informationstechnischer Sicht auf.
- Kapitel 4: Die prototypische Umsetzung ist in diesem Kapitel beschrieben und untergliedert sich in die folgenden drei Hauptbereiche.

  1. Referenzimplementierung als PlugIn der CAD-Software NX

     Ausgehend von einer bestehenden Konstruktion wird die mechanische Struktur erfasst, um diese in ein semantisches Informationsmodell auszuleiten. Darüber hinaus besitzt der Konstrukteur implizites Wissen über den Pro-

zessablauf, den die von ihm entwickelte Maschine erfüllen soll. Durch Verwendung des Mechatronics Concept Designers, einem Erweiterungsmodul von NX, kann das Kinematikmodell und das Ablaufverhalten einer Maschine beschrieben und simuliert werden. Auch diese Informationen werden in ein semantisches Informationsmodell überführt.

2. Referenzimplementierung eines semantischen Informationsmodells

   Aus technischer Sicht muss ermittelt werden, wie ein semantisches Informationsmodell sinnvollerweise realisiert werden kann. Für den ausgewählten Standard OPC UA wurden umfangreiche Erweiterungen im Typsystem vorgenommen, um die Informationen bzgl. Struktur und Verhalten eines automatisierten Produktionssystems semantisch abbilden zu können. Diese Erweiterungen werden in diesem Abschnitt spezifiziert.

3. Referenzimplementierung einer Ablaufumgebung

   Ziel der semantischen Beschreibung eines automatisierten Produktionssystems ist die Unterstützung der Projektierung auf Steuerungsebene. Um dies bestmöglich darzustellen, wurde eine Ablaufumgebung realisiert, die durch Verknüpfung von elementaren Projektierungselementen (Servicebausteine) projektiert werden kann.

Auf Basis der Referenzimplementierungen des semantischen Informationsmodells und der Ablaufumgebung kann eine Modelltransformation der Struktur- und Verhaltensbeschreibung aus dem semantischen Informationsmodell in die Ablaufumgebung dargestellt werden. Auf Steuerungsebene ist darauf aufbauend eine Verwendung dieser Informationen zur automatisierten Generierung einer webbasierten Visualisierung automatisierter Produktionssysteme möglich.

- Kapitel 5: Das Potential des vorgestellten Konzepts und dessen Realisierung wird in diesem Kapitel beschrieben. Dabei wird insbesondere darauf eingegangen, welche neuen Nutzungs- und Verwendungsmöglichkeiten die zusätzlichen Informationen aus dem Engineering auf Steuerungsebene eröffnen.
- Kapitel 6: Abschließend werden die Inhalte dieser Arbeit zusammengefasst und die erzielten Forschungsergebnisse dargestellt. Darüber hinaus sind weiterführende Gedanken erläutert.

2 Semantische Beschreibung von Automatisierungssystemen

In diesem Kapitel soll ein Überblick über die verschiedenen Standards und Normen sowie Konzepte und Methoden gegeben werden, die sowohl zur abstrakten als auch zur umsetzungsnahen Beschreibung, Modellierung und Programmierung verwendet werden. Daran anschließend wird der Handlungsbedarf abgeleitet und der Lösungsansatz dieser Arbeit motiviert.

2.1 Stand der Technik, Standards und Normen im Kontext automatisierter Produktionssysteme

Im Themenkomplex automatisierter Produktionssysteme gibt es eine Vielzahl relevanter Normen, Standards und Beschreibungsmittel (siehe Anhang 11.1). Einige werden in den nachfolgenden Abschnitten im Überblick dargestellt, um diese Arbeit besser in den Gesamtkontext einordnen zu können. In Abbildung 1 ist ein Überblick über die Struktur dieses Unterkapitels dargestellt.

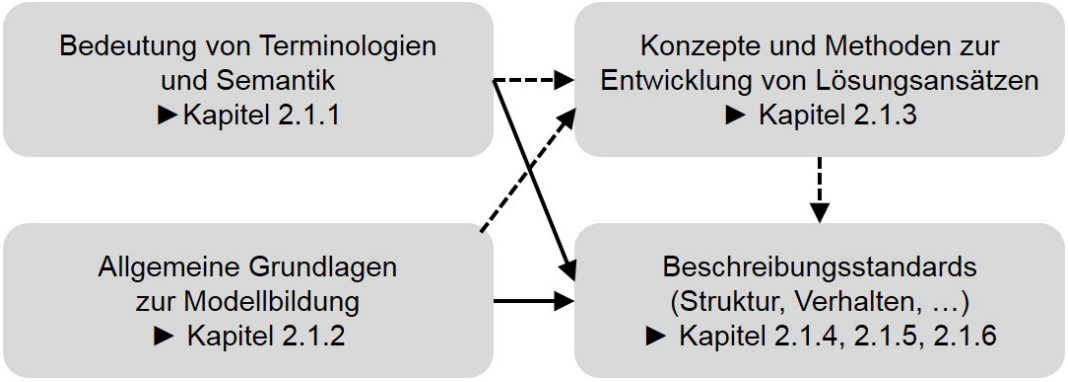


Abbildung 1: Darstellung der Strukturierung der beschriebenen Standards und Normen

Beginnend mit der Bedeutung von Terminologien und Semantik sowie zur Modellbildung werden allgemeine Grundlagen für das Verständnis und die Verwendung von Beschreibungsstandards geschaffen, die wiederum als Werkzeuge zur Verwendung bei der Spezifikation von Konzepten oder Methoden eingesetzt werden können. Hierbei ist ebenfalls ein Verständnis bzgl. Modellbildung, Terminologien und Semantik erforderlich.

2.1.1 Darstellung der grundlegenden Bedeutung von Terminologien und Semantik

Eine besondere Fähigkeit des Menschen ist es, im Laufe seines Lebens ein Begriffssystem aufzubauen, das materiellen bzw. nicht-materiellen Gegenständen zugeordnet ist. Darüber hinaus ist der Mensch in der Lage, durch sein gesammeltes

Wissen und seine Erfahrung kommunizierte Benennungen auf sein Begriffssystem abzubilden. [7] Dies stellt einen wesentlichen Unterschied zu Begriffssystemen und deren Verwendung bei M2M-Kommunikation dar, da hier eine eindeutige, deterministische Zuordnung ohne Interpretationsspielraum erfolgen muss, um einen sinnvollen Informationsaustausch realisieren zu können.

Aufbauend auf diesen Grundlagen von Begriffen und Benennungen werden in der Norm DIN 2342 [8] verschiedene Begriffe der Terminologielehre definiert.

Begriffssysteme können auf verschiedene Art aufgebaut und dargestellt werden. Sie können zur Klassifizierung verwendet werden und stellen Abhängigkeiten zwischen unterschiedlichen Begriffen dar. Diese Abhängigkeiten in Begriffssystemen können mittels graphischer Darstellungen dauerhaft definiert werden. Die hierfür verwendeten Repräsentationen wie zum Beispiel Liniendiagramme, Felderdiagramme, Beziehungsgraphen und Listen lassen sich als grundlegende Ordnungsprinzipien auf andere Anwendungsfälle in den nachfolgenden Kapiteln übertragen. [9]

Diese allgmeinen Grundlagen im Bereich Terminologien und Sematik von Begriffen können zur Spezialisierung bestimmter Domänen verwendet werden. So sind in DIN EN 61512-1 [10] zum Beispiel die Modelle und Terminologie der chargenorientierten Fahrweise definiert. Dabei wird zu Beginn die Bedeutung aller erforderlichen Begriffe definiert. Danach werden der strukturelle Aufbau von Anlagen sowie deren Ablaufstruktur (Prozess- und zustandsorientiert) beschrieben.

Ein anderes Beispiel zur Beschreibung von Terminologie und Semantik ist der DIN EN 62264-1 [11] zu entnehmen. Hier werden Modelle und Terminologien aus dem Bereich der Integration von Unternehmensführungs- und Leitsystemen beschrieben. Auch hier wird zu Beginn die Bedeutung der verwendeten Begriffe festgelegt. Darauf aufbauend können nun zum Beispiel Hierarchiemodelle, funktionale Datenflussmodelle und Objektmodelle zur eigentlichen Beschreibung der Integration von Unternehmensführungs- und Leitsystemen entwickelt werden.

Im Bereich des Anlagen-Engineerings bzw. der Mechatronik existiert eine solche standardisierte Terminologie bisher noch nicht. Da sich in diesem Bereich jedoch die Domänen „Mechanik“, „Elektrik“ und Software überschneiden, ist eine Zusammenarbeit verschiedener Berufsgruppen erforderlich. Dies impliziert in besonderem Maße den Bedarf an einer eindeutigen und einheitlichen Terminologie.

2.1.2 Überblick über Grundlagen zur Modellbildung im Kontext automatisierter Produktionssysteme

Fundamentale Grundlagen für das Modellverständnis und für die Modellbildung wurden bereits 1973 von Herbert Stachowiak in dem Standardwerk „Allgemeine Modelltheorie“ [12] veröffentlicht. Diese Ansätze sind noch immer übertragbar auf

Grundlagen der Modellbildung wie zum Beispiel bei der computergestützten ereignisdiskreten Ablaufsimulation [13; 14; 15], die zur Logistik- und Materialflusssimulation von Produktionssystemen eingesetzt wird. [4]

An dieser Stelle soll nur ein Überblick über die Allgemeine Modelltheorie gegeben werden. Umfassende Abhandlungen zur Modelltheorie bzw. Modellbildung können [12; 16; 17] entnommen werden. Die Allgemeine Modelltheorie definiert Ansätze zur Beschreibung von Modellen und Metamodellen. So ist ein Modell durch mindestens drei Merkmale gekennzeichnet:

- Abbildung: Modelle bilden Originale ab. Originale und Modelle werden als Attributklassen verstanden. Modelle selbst können wieder als Original verwendet werden. [12, S. 131–132]
- Verkürzung: Modelle bilden Originale bezüglich ihrer Attribute nicht vollumfänglich ab. [12, S. 132]
- Pragmatismus: Da Modelle die Abbildungen ihrer Originale sind, ist ihre Gültigkeit bzw. Anwendbarkeit an den Verwender bzw. Nutzer des Modells, den Verwendungszeitpunkt und den Verwendungszweck gebunden. [12, S. 132–133]

Das oben beschriebene modelltheoretische Verständnis wurde bereits 1973 veröffentlicht. Insbesondere die Fachbereiche der Informatik und der Informations- und Kommunikationstechnik beschäftigen sich mit der Modellbildung. Dies lässt sich an einigen Beispielen veranschaulichen:

- Datenbanksysteme werden zur strukturierten Speicherung von Informationen eingesetzt. Eine Produktdatenbank enthält zum Beispiel Informationen bezüglich verschiedener Produkte bzw. Produktvarianten. Somit bildet die Produktdatenbank die realen Produkte hinsichtlich bestimmter Attribute (Verkürzung) ab und ist dadurch ein Modell der physischen Produkte (Originale). Datenbanken werden in ihrer Struktur in der Regel auf den Anwendungsfall angepasst und umfassen applikationsspezifische Informationen (Pragmatismus).

- Softwareengineering beschäftigt sich mit der Entwicklung von Software, den erforderlichen Datenstrukturen und der Umsetzung in eine Laufzeitumgebung. Das grundlegende Konzept, der strukturelle Aufbau, Vererbungshierarchien und verwendete Kommunikationsmechanismen sowie die verwendeten Datenstrukturen sind kreative Leistungen der Entwickler. Nach Stachowiak entsprechen diese kreative Ideen den Originalen, die mittels Modellierungssprachen oder auch Programm-Code (Modellen) abgebildet werden.

Im Bereich des Softwareengineerings hat sich bereits der Trend zur modellgetriebenen Softwareentwicklung etabliert. So erweitert zum Beispiel der Ansatz der Modellgetriebenen Architektur (MDA) den klassischen Softwareentwicklungsansatz und differenziert zwischen der Beschreibung der Funktionalität und der Realisierung der Implementierung auf dem Laufzeitsystem. Um eine komplexe Applikation nicht in ei-

nem einzelnen Gesamtmodell beschreiben zu müssen, wird in mehrere Partialmodelle unterteilt:

- Computation Independent Model (CIM): Es definiert die Aufgaben eines Systems mittels formaler Spezifikation und legt das Vokabular der Domäne fest.
- Platform Independent Model (PIM): Baut auf CIM auf und verwendet die domänenspezifischen Beschreibungsmittel zur implementierungsunabhängigen Beschreibung von Struktur und Verhalten des Systems.
- Platform Specific Model (PSM): Das PSM überführt PIM auf ein plattformspezifisches Modell. Dieses Modell kann von Code-Generatoren zur Erzeugung von Programm-Code verwendet werden.

Bei der Entwicklung einer Steuerungslösung für automatisierte Produktionssysteme gibt es Analogien zur Softwareentwicklung. Während des Planungsprozesses wird eine Anlage in verschiedenen Aspekten digital nachgebildet (Modellbildung). Dies umfasst das Layout der Gesamtanlage, die Konstruktion jeder einzelnen Zelle oder Maschine. Diese Strukturinformationen können als Basis für eine Kinematisierung und anschließende Beschreibung des Bewegungsverhaltens genutzt werden. Bereits nach heutigem Stand der Technik können Informationen aus der digitalen Fabrik zur Generierung von Steuerungslösungen verwendet werden. Entsprechend der Motivation bei der Softwareentwicklung, die Komplexität der Software durch Verwendung von modellgetriebenen Ansätzen beherrschbarer zu machen, lässt sich im Bereich der Anlagenplanung ein ähnlicher Trend beobachten. Steigende Komplexität durch zunehmend anspruchsvolle Anforderungen der Kunden, steigender Einsatz von Sensorik und Aktuatorik und der Wunsch nach kürzeren Realisierungszeiträumen führen auch bei Planung und Inbetriebnahme automatisierter Produktionssysteme zu einem zunehmenden Einsatz modellgetriebener Softwarewerkzeuge zum Beispiel im Bereich der digitalen Fabrik bzw. durch Einsatz von Simulationstools während der Planungsphase.

Im Vergleich zur Softwareentwicklung besteht noch ein wesentlicher Unterschied in der Trennung zwischen der Modellbildung in der Planungsphase und der Programmierung bei Inbetriebnahme der Anlage sowie in den verschiedenen beteiligten Domänen. Die Modellierung während der Planungsphase wird in der Regel von Projektierungsingenieuren durchgeführt, die Inbetriebnahme von Inbetriebnahmeingenieuren, die für die Umsetzung in SPS-Programme verantwortlich sind. Darüber hinaus sind Mitarbeiter aus den Bereichen Mechanik, Elektrik und Software an der Realisierung beteiligt. Diese Heterogenität stellt einen wesentlichen Unterschied zu Projekten im Bereich Softwareengineering dar.

2.1.3 Konzepte und Methoden zur Entwicklung von Lösungsansätzen

Die Entwicklung automatisierter Produktionssysteme stellt in vielerlei Hinsicht eine Herausforderung dar. Die verschiedenen Fachdisziplinen Mechanik, Elektrik und Software müssen gemeinsam betrachtet bzw. projektiert werden, damit ein Produktionssystem die gewünschten Anforderungen erfüllt.

An dieser Stelle sollen grundlegende Methoden und Konzepte vorgestellt werden, die eine generelle Entwicklung von Lösungsansätzen erlauben.

Das TOTE-Schema

Das TOTE-Schema (Test – Operate – Test – Exit) definiert ein Modell (vgl. Abbildung 2), das elementare Denkprozesse des Menschen bei der Problemlösung beschreibt. Dieses Vorgehen bei der Problemlösung ist ein sehr intuitiver Vorgang. Dabei wird der Regelkreis bestehend aus Analyse (Test) und Anpassung/Veränderung (Operate) so lange durchlaufen, bis entweder keine Abweichung mehr zwischen Soll- und IST-Zustand existiert oder der Regelkreis vorzeitig abgebrochen wird. [18, S. 85–88]

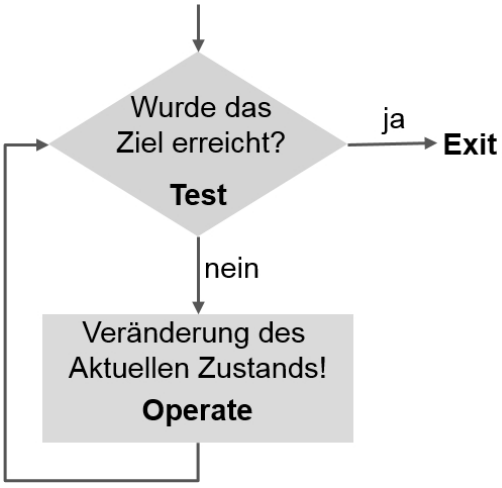


Abbildung 2: *Das TOTE-Schema [18, S. 86]*

Diese Art des Lösungsvorgehens lässt sich insbesondere bei einfachen und übersichtlichen Problemstellungen sinnvoll und effektiv anwenden. Bei steigender Komplexität zum Beispiel durch bestehende Abhängigkeiten, Einflussbedingungen oder andere Randbedingungen steigt die Gefahr, die Problemstellung unzulässig zu vereinfachen bzw. bestimmte Einflussfaktoren bei der Problemlösung nicht zu berücksichtigen. Aus diesem Grund sind in allen Bereich der Problemlösung Konzepte und Methoden erforderlich, die eine systematische Herangehensweise ermöglichen. Dies soll in den folgenden Abschnitten für verschiedene Abstraktionsebenen kurz vorgestellt werden.

Aufgliederung und Verknüpfung zur Problem- und Systemstrukturierung

Die in der VDI-Norm 2221 [19] beschriebene Methode der Aufgliederung und Verknüpfung zur Problem- und Systemstrukturierung (vgl. Abbildung 3) unterteilt das betrachtete Gesamtproblem (Aufgabenstellung) in mehrere Teilprobleme sowie anschließend in Einzelprobleme.

Dieses schrittweise Vorgehen vom abstrakten Gesamtproblem zum konkreten Einzelproblem ermöglicht eine Strukturierung komplexer Systeme und untergliedert umfangreiche, unübersichtliche Problemstellungen in überschaubare, handhabbare Einzelproblemstellungen. Diese können individuell gelöst werden, wobei die Einzellösungen anschließend wieder zu Teil- bzw. Gesamtlösungen aggregiert werden können.

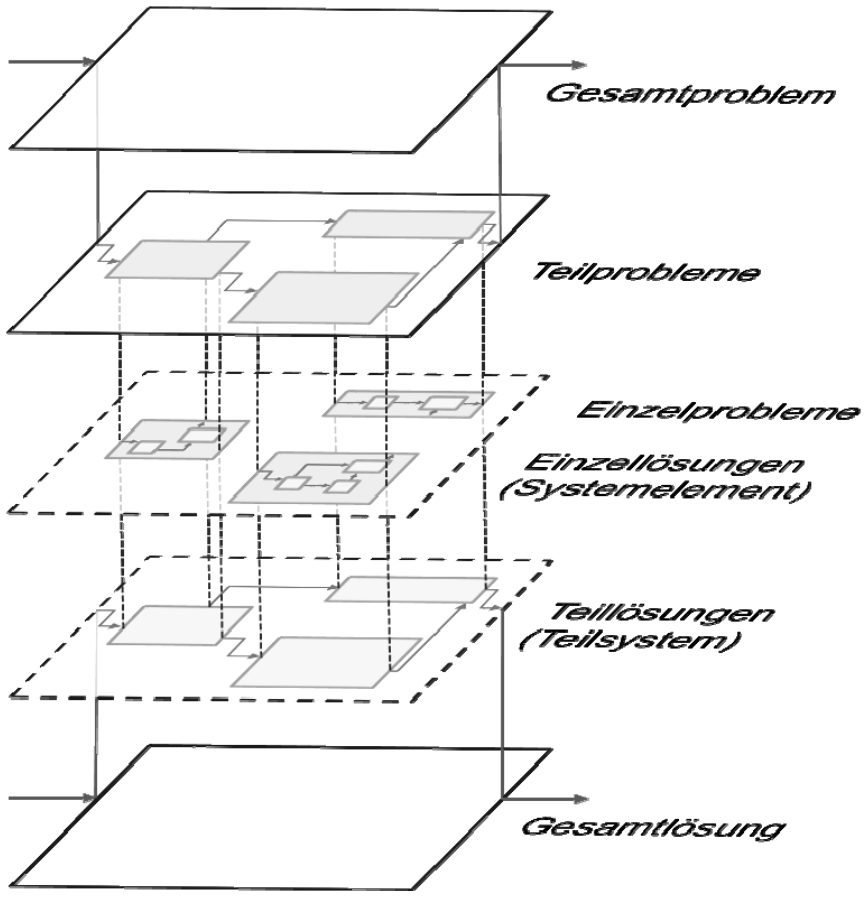


Abbildung 3: Methode der Aufgliederung und Verknüpfung zur Problem- und Systemstrukturierung nach VDI 2221 [19, S. 4]

Das Systems Engineering Konzept definiert den Problemlösungszyklus der Systemtechnik mittels folgender Schritte:

- Situationsanalyse
- Zielformulierung
- Analyse von Lösungen

- Synthese von Lösungen
- Bewertung und Entscheidung

Dieser Problemlösungszyklus kann bei Bedarf als allgemeingültiges Vorgehen während jeder Engineering-Phase eines Projekts zur Lösung von Problemen jeder Art eingesetzt werden. (vgl. [20, S. 47])

Allgemeingültige Vorgehensweisen beim Entwickeln und Konstruieren

Die Richtlinie VDI 2221 „Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte“ [19] beinhaltet allgemeingültige und branchenunabhängige Grundlagen methodischen Entwickelns und definiert hierfür grundlegende Arbeitsschritte und Arbeitsergebnisse, die als Leitlinie für die Praxis dienen sollen. Das in der VDI 2221 vorgeschlagene grundlegende Vorgehen zum Entwickeln und Konstruieren (siehe Abbildung 4) gliedert sich in sieben Arbeitsabschnitte, aus denen jeweils auch wiederum Arbeitsergebnisse resultieren.

Abbildung 4: Generelles Vorgehen beim Entwickeln und Konstruieren nach VDI 2221 [19, S. 9]

Während des Entwicklungs- und Konstruktionsprozesses werden diese Arbeitsschritte durchlaufen, wobei zwischen den Arbeitsschritten abhängig von der Aufgaben- bzw. Problemstellung iterativ vor- und zurückgesprungen werden kann. Je größer der Neuheitsgrad des zu entwerfenden Produkts oder Systems ist, desto ausgeprägter

ist das iterative Vorgehen. Insbesondere die Anforderungsliste stellt ein dynamisches Dokument dar, welches über den Entwicklungsprozess sukzessive angereichert und über die Iterationsschleifen angepasst werden muss. Die Anforderungsliste stellt somit eine „Informationsbrücke“ [19, S. 10] zwischen den verschiedenen Arbeitsschritten dar, die untereinander abhängig sein können und deshalb bei Veränderung konsistent gehalten werden müssen.

Vorgehenszyklus für die Systemsynthese zur Produkt- und Prozessentwicklung

Basierend auf dem Problemlösungszyklus der Systemtechnik entwickelte Ehrlenspiel den Vorgehenszyklus für die Systemsynthese, die sich auf die Bereiche Produkt- und Prozessentwicklung beschränkt. Dieser Vorgehenszyklus gliedert sich in die drei Arbeitsschritte Aufgabenklärung, Lösungssuche und Lösungsauswahl.

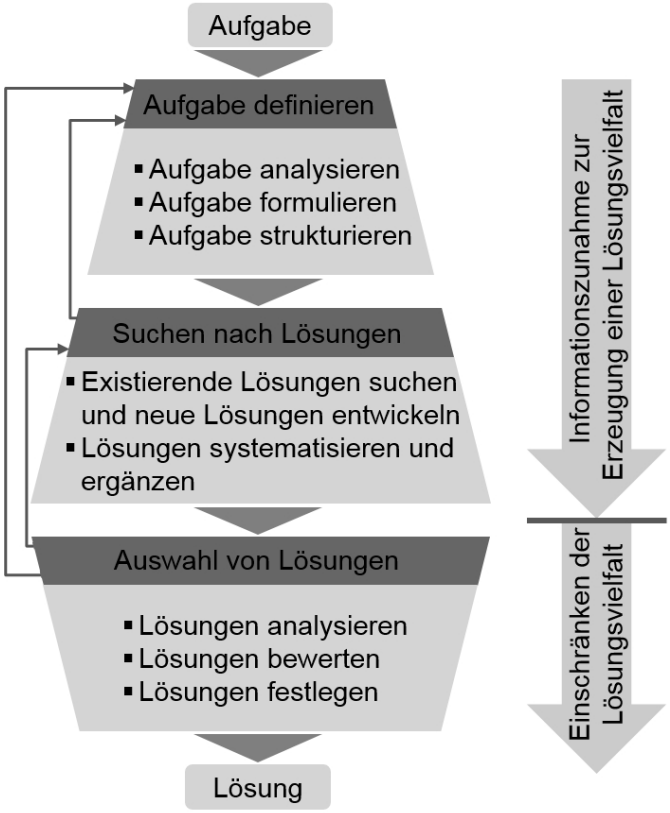


Abbildung 5: Vorgehenszyklus für die Systemsynthese nach Ehrlenspiel [21, S. 89]

Bei diesem Vorgehen ist die gezielte Suche nach verschiedenen Lösungsmöglichkeiten entscheidend, da hierbei eine Lösungsvielfalt entsteht, die bei der Lösungsauswahl wieder reduziert wird. Dabei sollen möglichst alle existierenden Lösungen betrachtet werden, wobei dann nach bestimmten Kriterien eine Lösungsauswahl stattfindet. Des Weiteren beschreibt Ehrlenspiel das wiederholte Durchlaufen von Arbeitsschritten mittels iterativer und rekursiver Rücksprünge.

Das Münchener Produktkonkretisierungsmodell (MKM)

In Erweiterung zur Richtlinie VDI 2221 entwickelte Ehrlenspiel das Pyramidenmodell der Produktkonkretisierung (vgl. Abbildung 6). Dieses Modell abstrahiert den Entwicklungsprozess auf vier differenzierbare Ebenen:

- funktionelle Lösungsmöglichkeit
- prinzipielle physikalische Lösungsmöglichkeit
- gestalterische und stoffliche Lösungsmöglichkeit
- fertigungs- und montagetechnische Lösungsmöglichkeit der Produktion

Jede Ebene lässt sich als Lösungsraum verstehen, der bestimmte Variationsmöglichkeiten bietet.

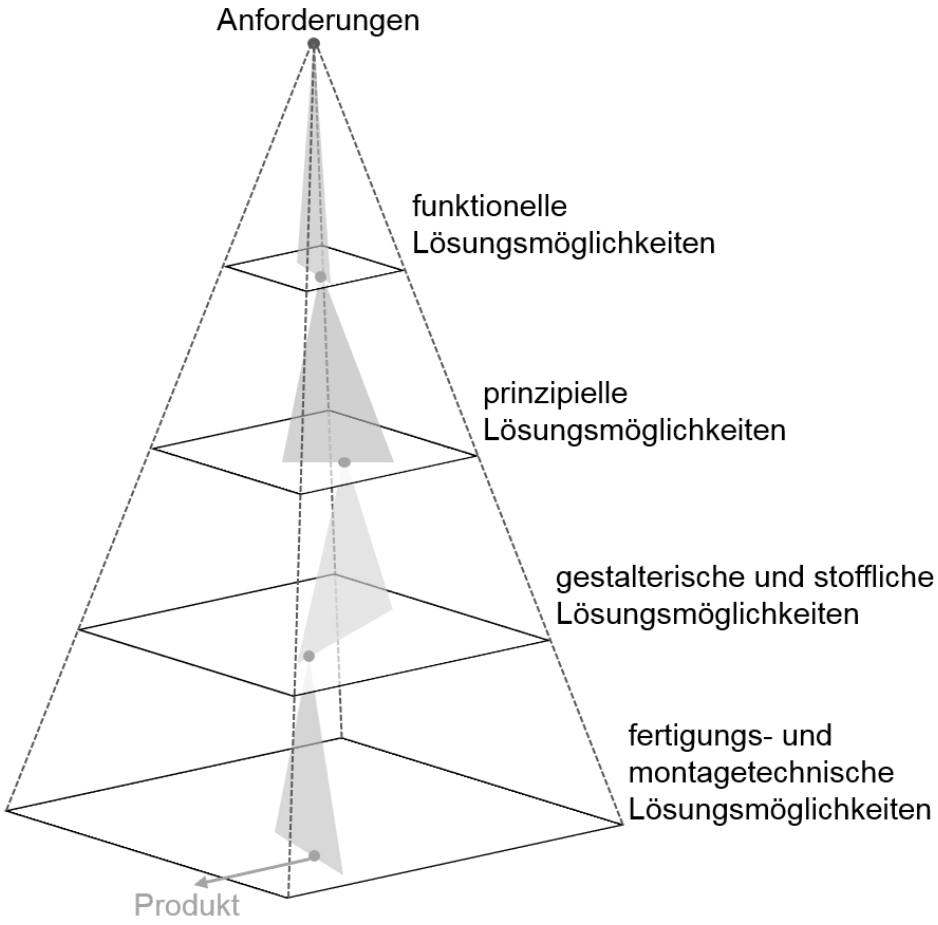


Abbildung 6: *Pyramidenmodell der Produktkonkretisierung nach Ehrlenspiel [21]*

Bedingt durch die variablen Lösungsmöglichkeiten auf den vier Ebenen potenzieren sich die Lösungsvariationen. Dies ist in Abbildung 6 durch eine fächerartige Darstellung ausgehend von einer Lösung (Punkt in der übergeordneten Ebene) zu der unterlagerten Ebene dargestellt.

Ein weiterer Modellierungsansatz stellt der Modellraum des Konstruierens von Rude dar (vgl. Abbildung 7). Auch hier wird der Entwicklungsprozess des Konstruierens in die vier Ebenen Anforderung, Funktion, Prinzip und Gestalt untergliedert. Diese Untergliederung wird als Abstraktion bezeichnet.

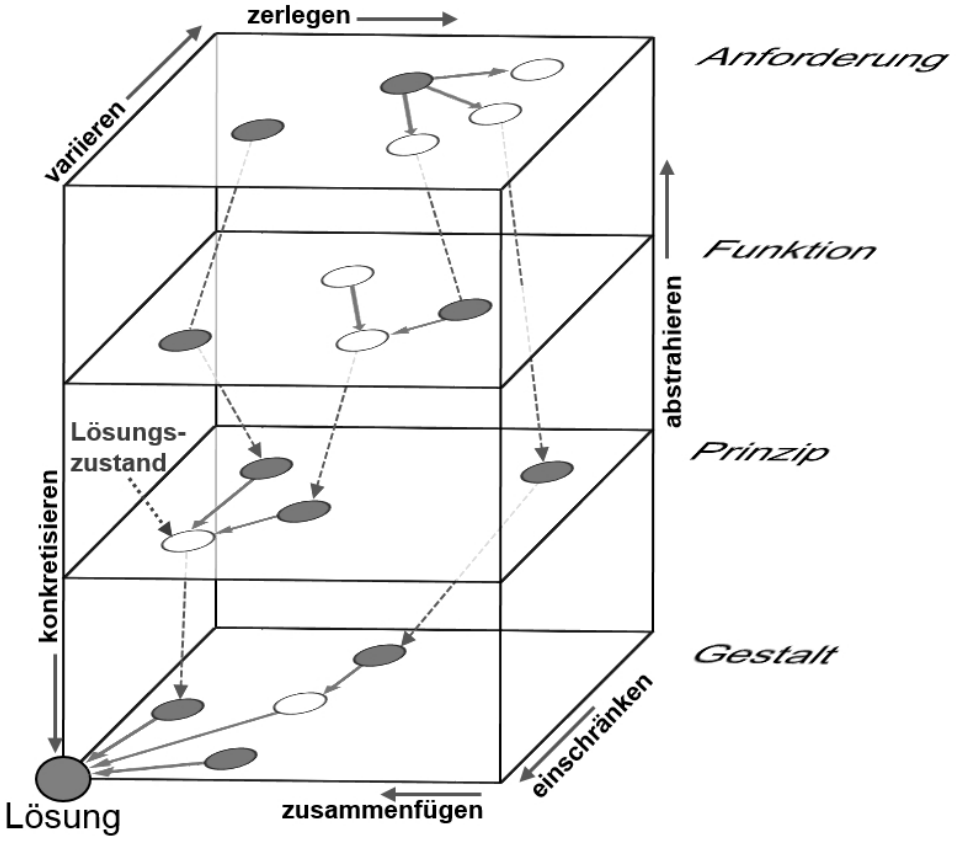


Abbildung 7: *Modellraum des Konstruierens nach Rude [22]*

Der Lösungsraum jeder Ebene wird durch Variation der Lösung sowie durch Zerlegung in Teilaspekte einer Problemstellung aufgespannt. Das Finden einer Lösung wird durch Konkretisierung, Zusammenfügen und Einschränken erreicht. Das besondere bei diesem Modell ist die Unterteilung in die bereits genannten vier Ebenen bei gleichzeitiger Integration des gegenläufigen Vorgehens

- abstrahieren – konkretisieren,
- zerlegen – zusammenfügen,
- variieren – einschränken

in ein gemeinsames Modell. Dies verdeutlicht, dass das Erarbeiten von Lösungsvarianten auf den verschiedenen Ebenen eine Grundlage schafft, um anhand bestimmter Kriterien konkrete Lösungsvarianten auswählen zu können. Dies erfordert die bereits genannten gegenläufigen Maßnahmen zur Auswahl und Festlegung konkreter Lösungsvarianten. In Anlehnung an die beiden vorgestellten Modelle wurde das Münchener Produktkonkretisierungsmodell (MKM) entwickelt. Wie in Abbildung 8 ersicht-

lich, setzt es sich aus einem Anforderungs- und Lösungsraum zusammen, wobei sich der Lösungsraum in Funktionsebene, Wirkebene und Bauebene unterteilt.

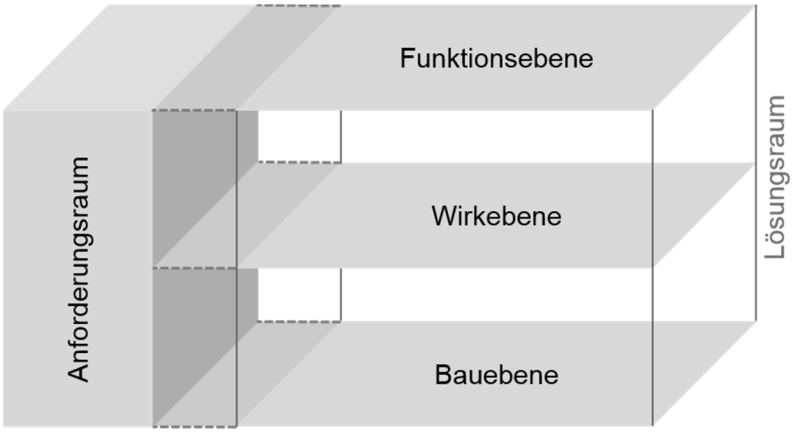


Abbildung 8: *Das Münchener Produktkonkretisierungsmodell [23, S. 27]*

Das MKM unterscheidet sich von den vorangegangenen Modellen durch die besondere Rolle, die der Anforderungsraum einnimmt. Dieser existiert parallel zu allen im Lösungsraum befindlichen Ebenen. Somit umfasst der Anforderungsraum funktions-, wirk- und bauspezifische Anforderungen, die auch auf den jeweiligen Ebenen während des Entwicklungsprozesses dynamisch entstehen können.

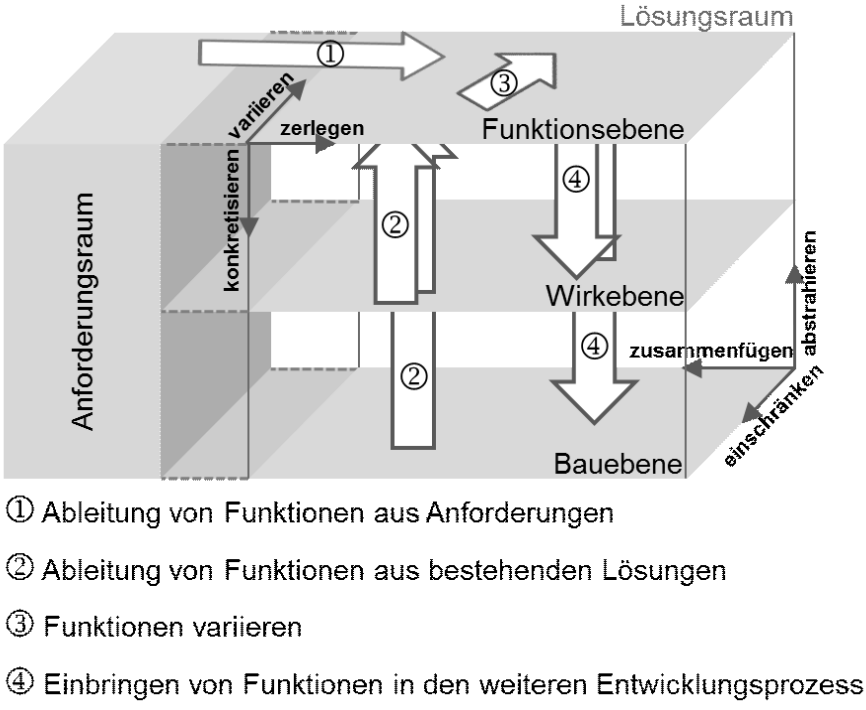


Abbildung 9: Einordnung in das Münchener Produktkonkretisierungsmodell (MKM) [23, S. 69]

Das in Abbildung 9 dargestellte Vorgehen beschreibt, wie das MKM Entwicklungsanforderungen mit bestehenden Lösungsvarianten abgleicht und eine adaptierte Lö-

sungsvariante entwickelt. Dabei werden im ersten Schritt aus den gestellten Anforderungen die erforderlichen Funktionen abgeleitet.

Im zweiten Schritt werden bereits existierende Lösungen bezüglich ihrer Verwendbarkeit für die gesuchte Funktion analysiert. Anschließend erfolgt im dritten Schritt eine anforderungsgerechte Variation der Funktionen. Diese werden im vierten und letzten Schritt in den weiteren Entwicklungsprozess auf Wirk- und Bauebene weiter eingebracht.

Verwendung des V-Modells zur Entwicklung mechatronischer Systeme

Die VDI-Norm 2206 „Entwicklungsmethodik für mechatronische Systeme“ beschreibt mit Hilfe des V-Modells einen ganzheitlichen Planungs- und Entwicklungsprozess, an dem verschiedene Gewerke beteiligt sind (siehe Abbildung 10). Dabei wird insbesondere auf ein generelles Vorgehen für eine getrennte parallele Abarbeitung der verschiedenen beteiligten Domänen eingegangen.

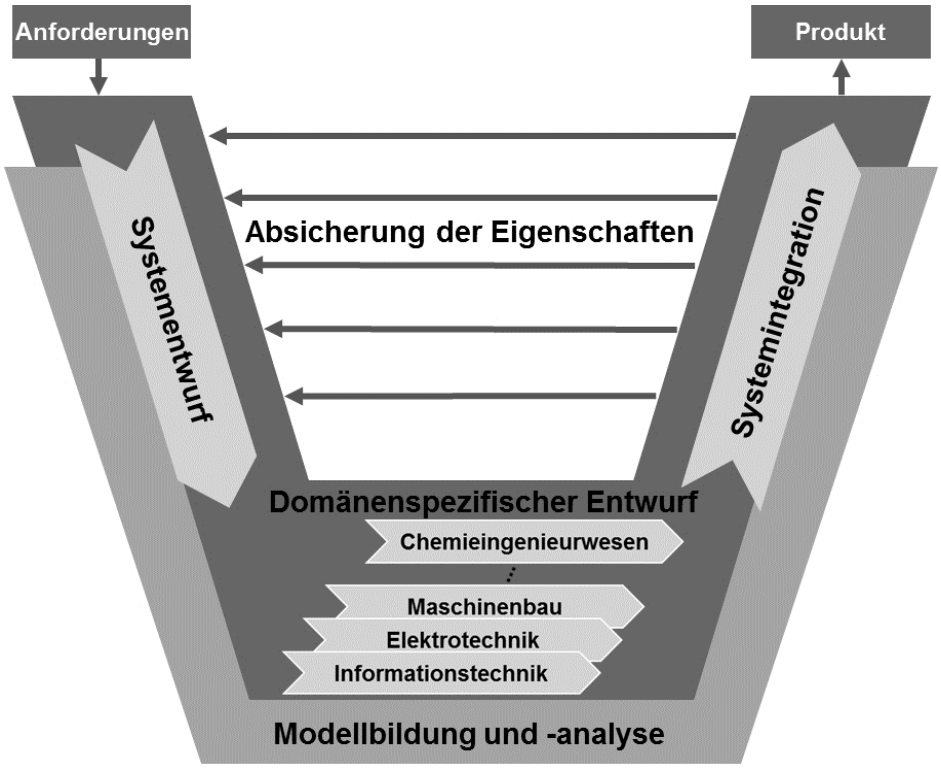


Abbildung 10: V-Modell als Makrozyklus nach VDI 2206 [24, S. 29]

Die Modellbildung und –analyse des V-Modells umfasst den Systementwurf, den domänenspezifischen Entwurf sowie die Systemintegration.

Während der Phase des Systementwurfs wird das grundlegende Lösungskonzept entwickelt. Dieses wird anschließend beim domänenspezifischen Entwurf für die verschiedenen beteiligten Gewerke domänenübergreifend ausgearbeitet. Die unter-

schiedlichen Arbeitsergebnisse der beteiligten Gewerke werden während der Systemintegrationsphase zusammengeführt.

Die Eigenschaftsabsicherung ist eng mit der Entwurfsphase verknüpft und realisiert einen sukzessiven Abgleich zwischen dem anforderungsgetriebenen Systementwurf und den Eigenschaften des sich in Entwicklung befindlichen Systems.

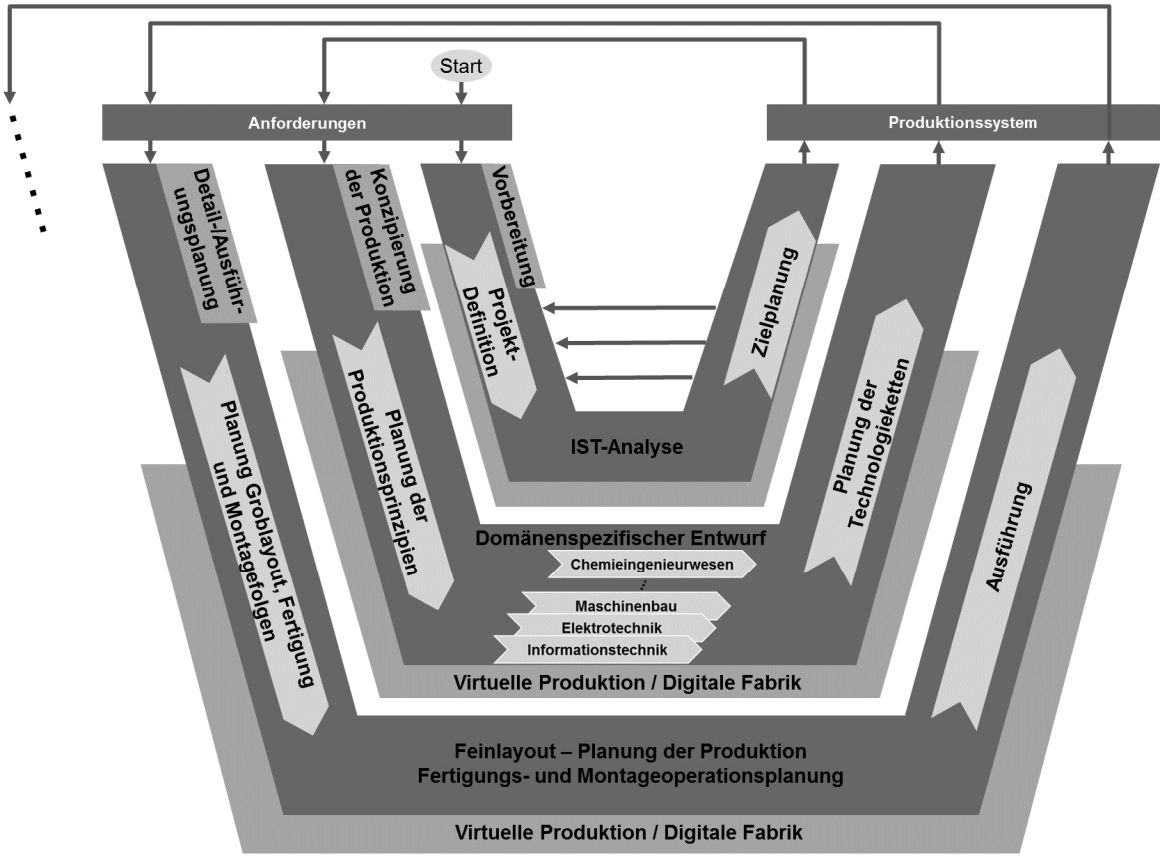


Abbildung 11: Iteratives Vorgehen beim Entwurf automatisierter Produktionssysteme nach VDI 2206 [24, S. 44]

Das vorgestellte methodische Vorgehen des V-Modells wird beim praktischen Einsatz auf verschiedenen Konkretisierungsebenen während der Planungsphase automatisierter Produktionssysteme mehrfach angewendet, wodurch eine sukzessive Detaillierung der Planung erreicht wird (siehe Abbildung 11). Der Reife- und Detailierungsgrad nimmt bei jedem iterativen Durchlauf des V-Modells stetig zu. Die komplette Planung und Entwicklung des Systems erfolgt in drei Phasen (Vorbereitung, Produktionskonzipierung und Ausführungsplanung), wobei in jeder Phase Planungsspezifikation und Planungsergebnisse kontinuierlich abgeglichen werden.

2.1.4 Beschreibungsstandards im Kontext automatisierter Produktionssysteme

Für die strukturierte Beschreibung technischer Systeme existieren verschiedene Beschreibungsstandards, die auch für automatisierte Produktionssysteme angewendet

werden können. Dabei muss der jeweilige Fokus und die Intension der Verwendung des betrachteten Beschreibungsstandards berücksichtigt werden. An dieser Stelle soll ein kurzer Überblick über ausgewählte Beschreibungsstandards gegeben werden. Es soll dabei insbesondere die Mächtigkeit der Ausdrucksfähigkeit der betrachteten Standards erörtert werden.

Semantische Datenmodelle

Semantische Datenmodelle dienen einer abstrakten, formalen Beschreibung und der Darstellung einer wahrgenommenen Realität, die in einem bestimmten Ausschnitt bzw. Zusammenhang beschrieben wird. Ziel ist die Abbildung der betrachteten Entitäten im gewünschten Detailgrad und deren Relationen untereinander in einem Datenmodell. Hierfür existieren verschiedene Modellierungssprachen, deren bekanntester Vertreter das Entity-Relationship-Modell ist.

Unified Modeling Language

Ziel der Unified Modeling Language (UML) ist es, Systemarchitekten und Softwareentwicklern Beschreibungsmittel zur Analyse, Entwicklung und Implementierung von (Software-)Systemen zur Verfügung zu stellen. Diese können zur Modellbildung in unterschiedlichen Bereichen verwendet werden. Die genaue Spezifikation von UML (Version 2.4.1) kann der zweiteiligen Norm ISO/IEC 19505 [25; 26] entnommen werden und soll an dieser Stelle lediglich in den elementaren Grundzügen der Beschreibungs- und Modellierungsmöglichkeiten vorgestellt werden. UML2 kann in eine Metamodellhierarchie mit vier Ebenen eingeordnet werden. Der Sprachumfang von UML2 (Metaebene M2) wird dabei von der darüber liegenden Metaebene M3 (Meta Object Facility) definiert. Mit Hilfe des Sprachumfangs von UML2 kann ein benutzerspezifisches Modell (Metaebene M1) beschrieben werden. Das auf Metaebene M1 definierte benutzerspezifische Modell kann in konkreten Ausprägungen instanziiert werden, was die Beschreibung von sich dynamisch verändernden Systemen zu Laufzeit ermöglicht. UML2 definiert auf Metaebene M2 sieben Struktur- sowie sieben Verhaltensdiagramme, die in Tabelle 1 aufgelistet sind.

Tabelle 1: Struktur- und Verhaltensdiagramme von UML2

Strukturdiagramme	Verhaltensdiagramme
▪ Klassendiagramm	▪ Aktivitätsdiagramm
▪ Kompositionsstrukturdiagramm	▪ Anwendungsfalldiagramm
▪ Komponentendiagramm	▪ Interaktionsübersichtsdiagramm
▪ Verteilungsdiagramm	▪ Kommunikationsdiagramm
▪ Objektdiagramm	▪ Sequenzdiagramm
▪ Paketdiagramm	▪ Zeitverlaufsdiagramm
▪ Profildiagramm	▪ Zustandsdiagramm

Mit Hilfe dieser Diagramme ergeben sich umfassende Möglichkeiten, ein System bezüglich dessen struktureller Aspekte sowie die Dynamik dessen kontinuierlichen Verhaltens zu beschreiben.

Systems Modeling Language

Die Systems Modeling Language (SysML) wurde zur Modellierung komplexer Systeme entwickelt und fokussiert sich insbesondere auf das Systems Engineering. SysML bietet hierfür verschiedene Diagrammtypen (vgl. Abbildung 13), die zur graphischen Systemspezifikation von Hard- und Software, auszutauschenden Informationen, beteiligten Akteuren und Funktionen und dem geplanten Einsatzort verwendet werden können. SysML macht sich die Ausdrucksmächtigkeit der UML 2 zunutze und erweitert diese mit Fokus auf das Systems Engineering (vgl. Abbildung 12). Ein Teil der verwendeten Diagrammtypen basieren auf UML 2 (Abbildung 13, grau), manche UML 2 Diagrammtypen werden durch die SysML in abgewandelter Form verwendet (Abbildung 13, blau) und andere Diagramme sind bei der SysML neu hinzugekommen (Abbildung 13, grün).

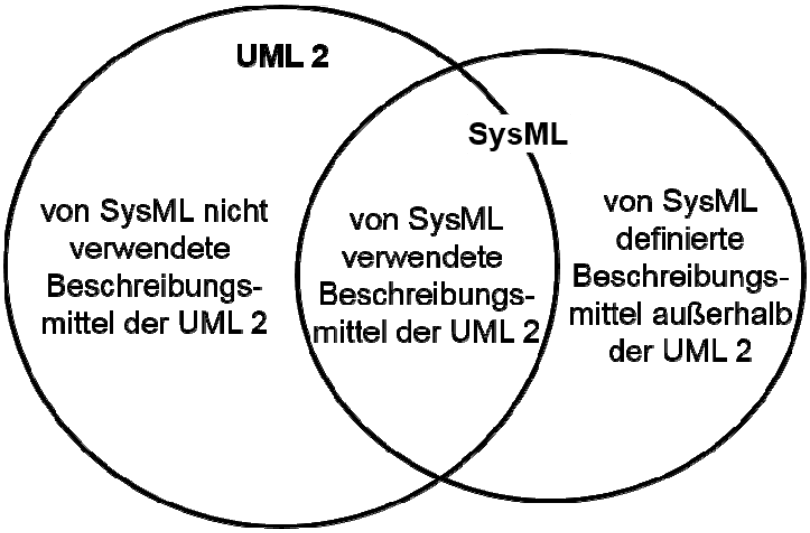


Abbildung 12: SysML basiert auf einer Untermenge der UML 2 und erweitert diese nach [27, S. 7]

SysML verfolgt einen ganzheitlichen Beschreibungsansatz technischer Systeme. Dies impliziert, dass ein System hinsichtlich verschiedener Aspekte betrachtet wird. Durch die Verwendung unterschiedlicher Beschreibungsmittel (Diagrammtypen) wird dies realisiert. Wie in Abbildung 13 dargestellt, klassifiziert SysML die verfügbaren Diagramme in drei Kategorien:

- Strukturdiagramm
- Verhaltensdiagramm
- Anforderungsdiagramm

Die Klassen der Struktur- und Verhaltensdiagramme werden durch weitere Diagrammklassen spezialisiert (siehe Abbildung 13).

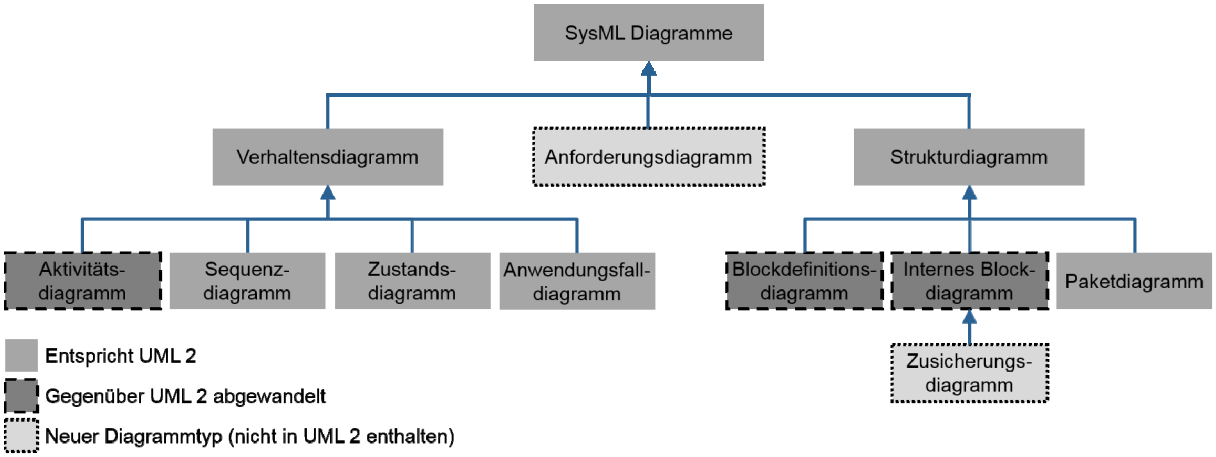


Abbildung 13: Darstellung der von SysML verwendeten Diagramme nach [27, S. 167]

Zusammenfassend lässt sich feststellen, dass SysML aufgrund der verschiedenen Diagrammtypen umfangreiche Möglichkeiten einer ganzheitlichen Systemspezifikation bietet, da das abzubildende System aus unterschiedlichen Sichten betrachtet und beschrieben werden kann. Der Detailgrad kann den Anforderungen entsprechend angepasst werden. Da wesentliche Bestandteile von SysML auf der UML 2 beruhen, wird die Einarbeitung deutlich erleichtert.

Computer Aided Engineering Exchange (CAEX)

Im Planungsprozess von Anlagen werden Informationen beispielsweise mittels Excel-Tabellen, Zeichnungen oder per Telefon ausgetauscht. Um einen standardisierten Informationsaustausch von Anlageninformationen zu unterstützen, wurde CAEX als XML-basiertes Austauschformat eingeführt. CAEX ist objektorientiert, wodurch Klassen-/Instanzkonzepte als auch Vererbungshierarchien verwendet werden können. Darüber hinaus ist es möglich, Objekten bestimmte Attribute und Schnittstellen zuzuweisen und Abhängigkeiten zwischen Objekten mittels Relationen auszudrücken. [28]

Hierfür definiert CAEX einige grundlegende Beschreibungselemente, die an dieser Stelle nur kurz vorgestellt werden. Da CAEX wie bereits erwähnt ein XML-basiertes Austauschformat ist, sind die CAEX-Elemente als XML-Tags notiert. Detaillierte Beschreibungen sowie die Spezifikation von CAEX können dem Anhang 11.2 sowie [29, S. 48–53; 30] entnommen werden.

An einem einfachen Beispiel lässt sich die Verwendung von CAEX darstellen. In Abbildung 14 ist eine einfache Objekthierarchie (oben links), deren Abbildung in die In-

stanceHierarchy „Example“ in einem CAEX-Editor (oben rechts) und die entsprechende Repräsentation in XML (unten) dargestellt.

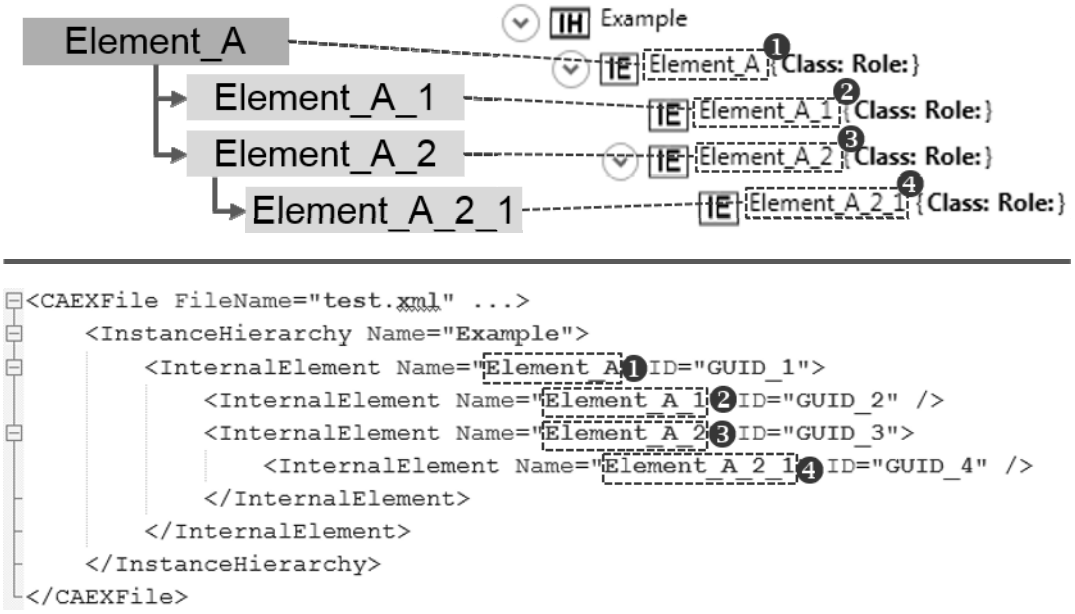


Abbildung 14: Darstellung einer CAEX-Objekthierarchie (oben) und deren Repräsentation in XML (unten)

Durch Verwendung dieser generischen Beschreibungselemente kann CAEX für beliebige Betrachtungsebenen und –tiefen verwendet werden. Es ist somit möglich, dass die InstanceHierarchy eine ganze Fabrik oder beispielsweise auch nur ein Teilsystem einer Maschine beschreibt.

Automation Markup Language (AutomationML)

Die Automation Markup Language (AutomationML) wurde eingeführt, um im Engineering einen herstellerübergreifenden Informationsaustausch zu unterstützen. So müssen Informationen bzgl. Topologie, Geometrie/Kinematik und Logik/Verhalten von Produktionssystemen bzw. Anlagenteilsegmenten beschrieben werden. Hierzu setzt AutomationML auf die Kombination verschiedener bereits bestehender Standards. Für die Beschreibung der (Teil-)Anlagentopologie werden die Strukturierungselemente von CAEX (siehe Anhang 11.2) verwendet. In diese übergeordnete Struktur integriert AutomationML weitere Datenaustauschformate zur Beschreibung von Geometrie, Kinematik, Verhalten sowie Ablaufsequenzen, indem ein CAEX-Objekt bezüglich der genannten Informationen erweitert wird (siehe Abbildung 15). Für Geometrie und Kinematik findet das XML-basierte offene Austauschformat COLLADA Verwendung. COLLADA steht für COLLAborative Design Activity und definiert ein XML-basiertes offenes Austauschformat, das für einen herstellerunabhängigen Informationsaustausch zum Beispiel zwischen CAD-Programmen entwickelt wurde. Die Spezifikation des Datenschemas liegt von COLLADA seit 2008 in der Version 1.5.0 vor [31]. Neben den Informationen von Geometriemodellen und Texturen können mit-

tels COLLADA auch Informationen bezüglich Programmeinstellungen und Bearbeitungsschritten gespeichert werden. Das zugrundeliegende XML-Schema von COLLADA v1.5.0 kann [32] entnommen werden.

Die Beschreibung der Logik bzw. des Ablaufverhaltens wird durch den Beschreibungsstandard PLCopen umgesetzt. PLCOpen ist eine Vereinigung verschiedenen Unternehmen und Institutionen mit dem Ziel einer produkt- und herstellerübergreifenden Standardisierung in den Bereichen Steuerungslogik, Motion Control, Safety und Kommunikation. PLCOpen definiert offene XML-basierte Schemata, die zur Abbildung von Informationen aus den verschiedenen bereits genannten Bereichen ermöglicht.

Wie in Abbildung 15 dargestellt, basiert die AutomationML-Architektur auf der Einbindung existierender Datenformate. Durch das offene Architekturmodell bietet AutomationML den Vorteil, dass zukünftig auch neue Formate eingebunden werden können, soweit dies notwendig sein sollte. Der hierfür erforderliche Mechanismus einer Referenzierung zwischen topologie- und domänenspezifischem Beschreibungsstandard ist generisch verwendbar.

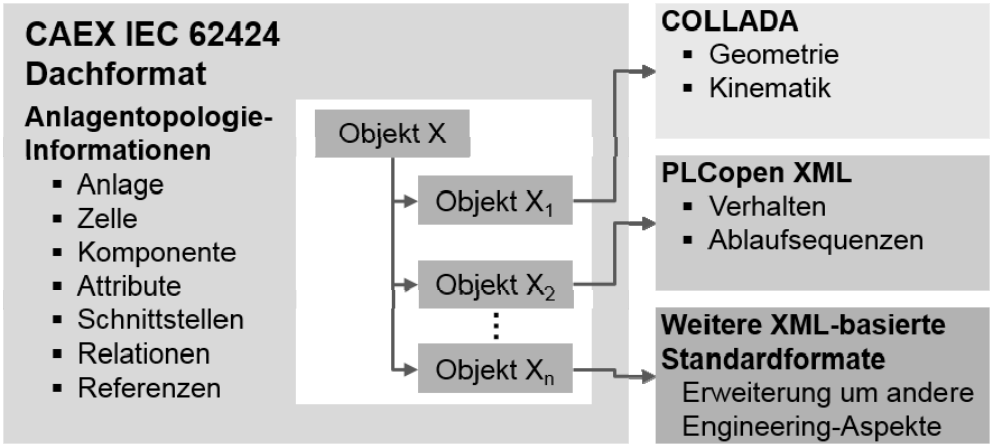


Abbildung 15: Darstellung der AutomationML-Architektur nach [29, S. 26]

Eine Besonderheit bei AutomationML ist der Intermediate Modelling Layer (IML). Ziel ist die Bereitstellung eines Metamodells zur Abstraktion graphischer Beschreibungsstandards wie zum Beispiel Gantt Diagramme, Pert Diagramme, Impuls-Diagramme und Zustandsdiagramme. Durch die Abbildungsfähigkeit der IML nach PLCopen XML ist eine Überführung der oben genannten Diagramme in einen Beschreibungsstandard möglich, der auf Steuerungsebene zur SPS-Programmierung verwendet werden kann. Ausführliche Beschreibungen zu AutomationML können [29; 33; 34; 35] entnommen werden.

Elektrokonstruktion mechatronischer Systeme

Mechatronische Systeme bestehen zu einem wesentlichen Anteil aus elektrischen und elektronischen Komponenten [36]. Die Konstruktion und Verschaltung dieser Komponenten, auch als Elektrokonstruktion bezeichnet, leistet einen wichtigen Beitrag zur erfolgreichen Planung und Umsetzung automatisierter Produktionssysteme.
Bezugnehmend auf den bereits vorgestellten Produktentwicklungsprozess ist die Elektrokonstruktion dem Realisierungsteil zuzuordnen, erfordert jedoch Schnittstellen zu den Domänen Mechanik und Software.
Der Bezug zur Mechanik ist bei der Elektrokonstruktion wichtig, da sowohl mechanische Design-Entscheidungen bezüglich Layout und Bauraum als auch die geometrischen Informationen elektrischer und elektronischer Komponenten auf die Elektrokonstruktion Einfluss nehmen.
Der Bezug zwischen Elektrokonstruktion und Software ist ebenfalls essentiell, da die Software nur bei korrekter Adressierung und Signalzuordnung die gewünschte Funktionalität erfüllen kann. Die Entwicklung der Steuerungssoftware und die Elektrokonstruktion überschneiden sich vor allem bei der Zuordnung der Ein- und Ausgänge der SPS. Diese Ein- und Ausgänge können funktional in der Steuerungssoftware verwendet werden und stellen den Übergang zur elektrischen Anbindung der Automatisierungskomponenten an die Steuerungshardware dar.
Als Ergebnis der Elektrokonstruktion resultieren fest definierte Unterlagen, die aufgrund der verwendeten genormten Kennzeichnungen und Schaltzeichen eindeutig interpretierbar sind.
Ein weiterer Aspekt der digitalen Elektrokonstruktion ist der Schaltschrankentwurf. Aufgrund der engen Interaktion mit der mechanischen Konstruktion können die Geometrieinformationen der elektrischen und elektronischen Baugruppen seitens des ECAD-Werkzeugs wie zum Beispiel bei EPLAN Electric P8 mit dem Erweiterungsmodul ProPanel verwendet werden, um neben der elektrischen Verschaltung auch die Anordnung und Positionierung der Komponenten im Schaltschrank festzulegen. Diese Projektierungsinformationen wiederum können zur automatisierten Bestückung und Montage der Baugruppen im Schaltschrank, zur automatisierten Konfektionierung der benötigten Kabel in passenden Längen sowie zur automatisierten Verschaltung elektrischer und elektronischer Baugruppen mittels der vorkonfektionierten Kabel verwendet werden.

OPC Unified Architecture

OPC Unified Architecture (OPC UA) ist die neueste OPC-Spezifikationen der OPC Foundation, die als IEC-Norm in mehreren Teilen vorliegt [37; 38; 39; 40; 41; 42; 43; 44; 45; 46; 47; 48]. Im Gegensatz zu den Vorgängerstandards ist OPC UA plattformunabhängig und dadurch auf verschiedenen Hardwarearchitekturen bzw. Betriebssystemen einsetzbar.

OPC UA verbindet Vorteile und Fähigkeiten eines Kommunikationsprotokolls mit den semantischen Möglichkeiten eines objektorientierten Beschreibungsstandards. Wie in

Abbildung 16 dargestellt, basiert die OPC UA Architektur auf dem OPC UA Binary Layer, der für Serialisierung bzw. Deserialisierung der ihm übergebenen Daten verantwortlich ist und setzt typischer Weise auf einer TCP/IP-Kommunikation auf. Es kann auch das sogenannte Hybrid-Protokoll eingesetzt werden, wobei HTTP/HTTPS als Transport-Protokoll und darauf aufbauend OPC UA Binary verwendet wird. Der „OPC UA Base Technology"-Layer stellt das grundlegende Service Set (Basisdienste) sowie das Adressraummodell von OPC UA zur Verfügung. Dieses umfasst zum Beispiel Datentypen, Ereignistypen, Objekttypen, Referenztypen und Variablentypen.

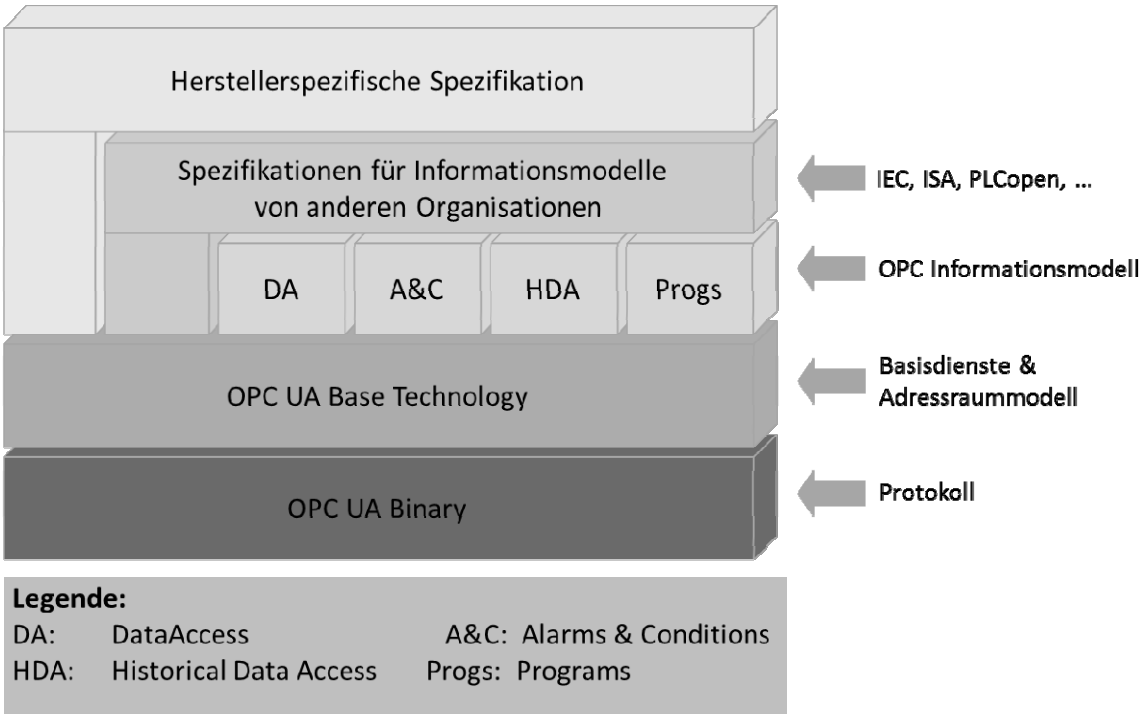


Abbildung 16: Darstellung der Architektur des OPC UA Protokolls sowie weitere darauf aufbauende Beschreibungsmodelle nach [49]

Die Datentypen umfassen unter anderem auch semantisch definierte SI-Einheiten [50], was einen wesentlichen Unterschied zur bisherigen Kommunikation und Datenverarbeitung im Bereich der Automatisierungstechnik darstellt. So müssen ausgetauschte Daten nicht mehr interpretiert werden, da diese bereits typisiert und semantisch beschrieben sind. Im Rahmen der Spezifikation von OPC UA existieren bestimmte Erweiterungen wie zum Beispiel „DataAcccess (DA)", „Historical Data Access (HDA)" und „Alarms & Conditions (A&C)". Diese standardisieren den Zugriff auf Aktualwerte (DA), auf zeitliche Werteverläufe (HDA) sowie Zustandsüberwachung mit Alarmmanagement. Durch die grundlegende Beschreibungsfähigkeit von OPC UA ist die Abbildung von anderen Standards (IEC, ISA, PLCopen, …) auf der Ebene zur Spezifikation von Informationsmodellen anderer Organisationen möglich. Durch Erzeugung von benutzerdefinierten Namensräumen können die genannten Typsysteme darüber hinaus noch individuell erweitert werden (herstellerspezifische Spezifikation).

Die OPC UA Spezifikation schreibt die Unterstützung bestimmter Kryptoalgorithmen vor, was zum Beispiel eine Applikationsendpunkt-Verschlüsselung erlaubt. Im Gegensatz zu Protokoll-Endpunktverschlüsselungen wie zum Beispiel HTTPS bietet dies erheblich mehr Sicherheit, da eine Sicherung des Informationsaustauschs bis auf Applikationsebene realisiert wird. Dadurch kann eine Schadsoftware auf dem Zielsystem die entschlüsselten Daten hinter einer HTTPS-Verbindung nicht mehr auslesen, wodurch „man-in-the-middle“-Angriffe ohne vorheriger Entschlüsselung der Kommunikation ausgeschlossen werden können.

OPC UA unterstützt Publish&Subscribe. Dies ist ein Mechanismus, der einem Kommunikationspartner ermöglicht, sich zum Beispiel auf Wertänderungen bestimmter Variablen anzumelden (subscribe). Bei Wertänderung werden alle angemeldeten Clients vom Host-System aktiv über die Wertänderung informiert (publish). Dies vermeidet clientseitig unnötiges Polling und reduziert daher den Kommunikationsoverhead erheblich. OPC UA verwendet dabei ein von unterlagerten Protokollen unabhängiges Session-Konzept. Dadurch bleiben zum Beispiel OPC UA Sessions weiterhin gültig, auch wenn unterlagerte Protokolle aufgrund von Timeouts bereits ihre Session beendet haben. Dadurch abstrahiert OPC UA von unterschiedlichem Protokollverhalten und bietet eine einheitliche Verwendung auf Applikationsebene.

Die von OPC UA verwendeten Ports 4840 sowie 4843 sind von der „Internet Assigned Numbers Authority (IANA)“ festgeschrieben und werden sowohl für TCP-Verbindungen als auch zum Austausch von UDP-Datagrammen verwendet. [51]

2.1.5 Verhaltensmodelle im Kontext automatisierter Produktionssysteme

Zur Beschreibung von Abläufen bzw. des Verhaltens von mechatronischen Systemen existieren verschiedene Ansätze, Konzepte und Standards. Nachfolgend werden ausgewählte Beschreibungs- und Modellierungsmöglichkeiten dargestellt.

Balkendiagramme

Balkendiagramme oder auch Gantt-Diagramme wurden erstmalig in dieser Form von Henry L. Gantt im frühen zwanzigsten Jahrhundert verwendet und dienten einer Transparenzschaffung im Fertigungsumfeld mit dem Ziel der Produktivitätssteigerung [52; 53]. Heute werden einfache Balkendiagramme (Gantt-Diagramme) zum Beispiel im Bereich des Projektmanagements zur Ablaufplanung eingesetzt und finden auch unter anderem Anwendung bei der Planung von Prozessabläufen im Engineering automatisierter Produktionssysteme. Gantt-Diagramme untergliedern sich in zwei grundlegende Elemente:

- Eine Liste, die alle Aktivitäten enthält (oftmals auf der linken Seite des Zeichnungsbereichs angeordnet),
- Ein Zeichnungsbereich, in dem der Zeitverlauf der Aktivitäten dargestellt wird.

Jede Aktivität aus der Aktivitätenliste besitzt eine Balkendarstellung im Zeichnungsbereich, wobei die Länge des Balkens auf dem Zeitstrahl der veranschlagten Dauer der Aktivität entspricht. Die verschiedenen Balken können mittels Kanten verknüpft und dadurch Vorgänger- bzw. Nachfolgerbeziehungen ausgedrückt werden.

Netzplantechnik

Die Netzplantechnik findet hauptsächlich in der Terminplanung von Projekten ihre Anwendung und dient der graphischen Darstellung der zeitlichen Abfolge von Aufgaben und deren Anordnungsbeziehung. Wie in Abbildung 17 dargestellt, untergliedert sich die Netzplantechnik in die drei Methoden Vorgangspfeilnetzplan, Ereignisknotennetzplan sowie Vorgangsknotennetzplan.

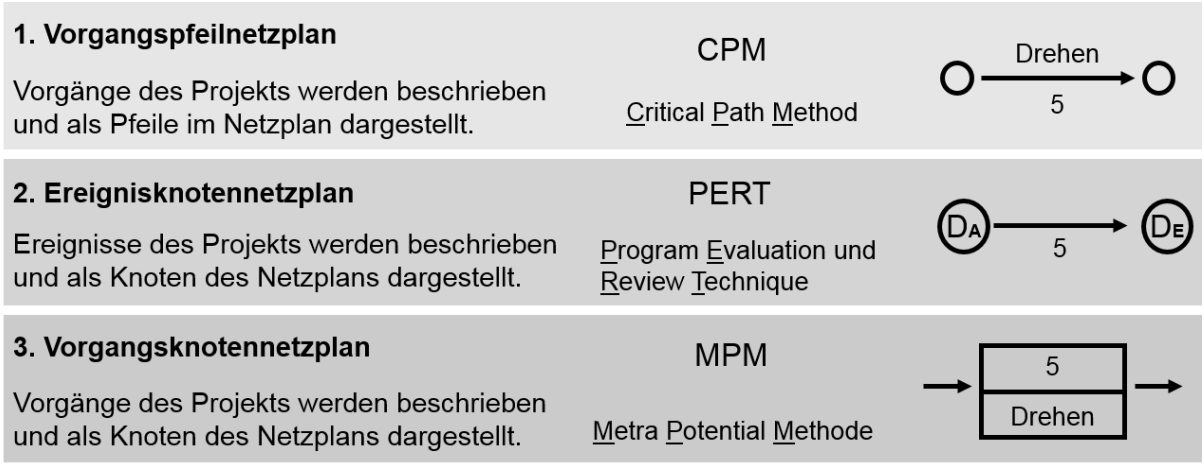


Abbildung 17: Darstellung der drei wesentlichen Netzplantechniken

Beim Vorgangspfeilnetzplan werden den Kanten des Netzplans die Vorgänge inklusive ihrer Dauer angetragen. Durch dieses Vorgehen lässt sich relativ einfach der Pfad mit der längsten Gesamtdauer ermitteln, was als *Critical Path Method* bezeichnet wird.

Der Ereignisknotennetzplan stellt die Ereignisse als Knoten dar und trägt der nachfolgenden Kante die Dauer des Ereignisses an. Das bekannteste Beispiel aus dem Bereich der Graphentheorie ist PERT (Program Evaluation and Review Technique).

Bei dem Vorgangsknotennetzplan werden die Vorgänge als auch deren Dauer in den Knoten dargestellt. Die Kanten dienen ausschließlich der Festlegung der Anordnungsbeziehungen. [54; 55]

GRAFCET-Diagramme

Das Akronym GRAFCET steht für Graphe Fonctionnel de Commande Etape Transition. Ins Deutsche übersetzt bedeutet das so viel wie *Darstellung der Steuerungsfunktion mit Schritten und Weiterschaltbedingungen*.

Die graphische Spezifikationssprache von GRAFCET ist in der Norm IEC 60848 [56] definiert, wobei die wichtigsten Elemente von GRAFCET nachfolgend kurz vorgestellt werden sollen:

- Schritte: Der Gesamtablauf eines Prozesses wird in einzelne Schritte unterteilt. Den Schritten sind jeweils eine oder mehrere Aktionen zugeordnet, die bei Aktivierung des Schritts ausgeführt werden. Um den Beginn einer GRAFCET-Ablaufsequenz zu kennzeichnen, wird ein initialer Schritt definiert.
- Transitionen: Der Übergang zwischen zwei Schritten wird mittels Transitionen definiert. Den Transitionen sind bestimmte Übergangsbedingungen zugeordnet, die erfüllt sein müssen, damit ein Übergang stattfinden kann. Die Transitionsbedingungen werden mittels boolescher Logik spezifiziert.
- Aktionen: Mittels Aktionen wird das System zum Beispiel durch Wertsetzung von Variablen entsprechend der Vorgaben des GRAFCET-Steuerungsprogramms manipuliert. Aktionen können auch in Abhängigkeit zu Bedingungen oder Ereignissen stehen.
- Verzweigungen: Mit Hilfe von Verzweigungen kann die Ablaufkette zum Beispiel in nebenläufige Ausführungsstränge aufgeteilt werden. Die klassischen Verzeigungen sind die Alternativ- (ODER) bzw. Parallelverzweigung (UND).
- Rückführung: Für eine zyklische Ausführung einer Ablaufkette, wie es im industriellen Umfeld von speicherprogrammierbaren Steuerungen üblich ist, wird bei GRAFCET eine Rückführung verwendet, die einen Rücksprung einer Ablaufkette zu einem bereits ausgeführten Ablaufsegment ermöglicht.

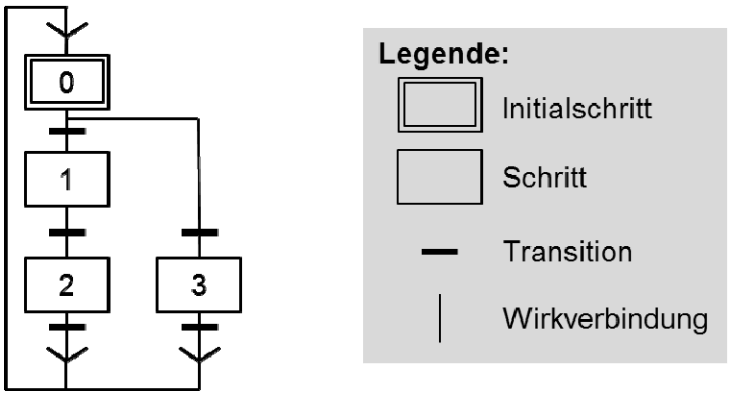


Abbildung 18: Darstellung eines GRAFCET Diagramms

Ein GRAFCET-Diagramm (siehe Abbildung 18) setzt sich aus den oben aufgeführten Elementen zusammen, wobei zwischen zwei Schritten immer eine Transition angeordnet ist. Bei Verwendung von Verzweigungen bedeutet dies, dass bei Alternativverzweigungen nach der Verzweigung in jedem Ablaufzweig sich eine Transition vor dem ersten Schritt sowie nach dem letzten Schritt befindet. Bei Parallelverzweigungen ist jeweils eine Transition vor der Verzweigung in parallele Ausführungsstränge sowie nach der Zusammenführung der Ausführungsstränge angeordnet.

Die Modellierung komplexer Abläufe wird durch die Möglichkeit der Modularisierung von Teilabläufen unterstützt, indem mehrere Schritte und Transitionen in einem Schritt zusammengefasst werden können. Dies erlaubt eine hierarchische Strukturierung des Ablaufprogramms. [57]

Impulsdiagramme

Impulsdiagramme werden zur graphischen Darstellen von Schaltsignalen verwendet und können zum Beispiel resultierende Ausgangssignale bei Variation der Eingangssignale darstellen, um die Funktionsweise digitaler Schaltungen zu überprüfen. [58]

Des Weiteren ist eine Visualisierung der Signallaufzeit im System in Bezug zum Systemtakt möglich. Somit lässt sich darstellen, nach wie vielen Taktzyklen zum Beispiel ein Berechnungsergebnis an einem Ausgang des betrachteten Systems vorliegt.

Zustandsdiagramme

Zustandsdiagramme beschreiben den dynamischen Zustand eines Systems in Abhängigkeit von äußeren Einflüssen zur Laufzeit. Ein Zustandsdiagramm wird mit Hilfe von Zuständen und Transitionen erstellt, wobei den Transitionen Bedingungen angelagert werden können. In Abhängigkeit des aktuellen Zustandes und den gegenwärtig erfüllten Transitionsbedingungen ist ein Zustandsübergang zu einem nachfolgenden Zustand möglich.

Aufgrund ihrer intuitiven Handhabung werden Zustandsdiagramme auch zur Projektierung von speicherprogrammierbaren Steuerungen nach IEC 61131-3 verwendet. In dieser Norm wird dies als *Ablaufsprache* bezeichnet. Bei STEP 7 (Siemens) wird zum Beispiel S7 GRAPH eingesetzt, was einer Art Zustandsdiagramm entspricht und als Weiterentwicklung von GRAFCET gilt [59].

2.1.6 Programmierstandards zur Projektierung speicherprogrammierbarer Steuerungen

Für eine herstellerübergreifende Standardisierung der Programmierung von speicherprogrammierbaren Steuerungen wurden die zwei grundlegenden Normen IEC 61131 und IEC 61499 etabliert, die nachfolgend in ihren Grundzügen beschrieben werden. Diese Standardisierung erleichtert dem Anwender die Projektierung von Steuerungskomponenten verschiedener Hersteller, da die zugrundeliegenden Programmierparadigmen gleich sind und sich lediglich die Bedienung der Projektierungstools unterscheidet.

IEC 61131

Die Norm IEC 61131 fasst verschiedene Normen zusammen und referenziert mehrere internationale Standards. Sie definiert die Verwendung von Zeichencodes und Nomenklatur sowie den Aufbau graphischer Darstellungen. Die Norm besteht aktuell

aus sechs Teilen sowie aus zwei *Technical Reports* (TR). Diese beschreiben jeweils bestimmte Aspekte der Steuerungsprogrammierung und sind nachfolgend aufgelistet:

- Teil 1 - Allgemeine Informationen:
  In diesem Teil der Norm werden allgemeine Begriffsbestimmungen und typische Funktionsmerkmale speicherprogrammierbarer Steuerungen beschrieben. Dies beinhaltet zum Beispiel die zyklische Abarbeitung von Steuerungsprogrammen mit einem persistierten Abbild der Ein- und Ausgangswerte sowie die Differenzierung zwischen Programmier-, Automatisierungs-, Bedien- und Beobachtungsgerät. [60]
- Teil 2 - Betriebsmittelanforderungen und Prüfungen:
  Dieser Teil der Norm definiert die elektrischen, mechanischen und funktionellen Anforderungen an die Geräte und umfasst außerdem entsprechende Typprüfungen. [61]
- Teil 3 – Programmiersprachen:
  Im dritten Teil der Norm werden das Softwaremodell und die zur Verfügung stehenden Programmiersprachen festgelegt. [62]
- Teil 4 - Anwenderrichtlinien (TR):
  Der vierte Teil der Norm ist als Technical Report verfasst und dient als Leitfaden für alle Projektphasen der Automatisierung. Die praxisorientierten Hinweise erstrecken sich von der Systemanalyse über die Gerätewahl bis hin zur Wartung. [63; 64]
- Teil 5 - Kommunikation:
  Der fünfte Teil der Norm behandelt Kommunikationsaspekte im Bereich der SPS-Programmierung sowohl zwischen unterschiedlichen Herstellern als auch mit anderen Geräten. [65]
- Teil 6 - Sicherheitsgerichtete SPS:
  Im sechsten Teil der Norm werden die Anforderungen der Sicherheitsnormen IEC 61508 [66] und der Maschinenrichtlinie IEC 62061 [67] aufbereitet und in die Norm IEC 61131 integriert. [68]
- Teil 7 - Fuzzy Control Language:
  Teil sieben der Norm IEC 61131 vermittelt ein Verständnis von Fuzzykontrollierten Anwendungen sowie von deren Portierungsmöglichkeiten. [69]
- Teil 8 - Leitlinien für die Anwendung und Implementierung von Programmiersprachen für Speicherprogrammierbare Steuerungen (TR):
  Der achte Teil der Norm ist als Technical Report verfasst und beinhaltet Richtlinien zur Implementierung, Anwendungshinweise und programmiertechnische Hilfestellungen. [70]

Im Kontext dieser Arbeit ist insbesondere der dritte Teil der IEC 61131 als aktueller Stand der Technik bei der Projektierung und Programmierung von speicherprogram-

mierbaren Steuerungen relevant. Aus diesem Grund wird dieser Teil selektiv aufgegriffen und dessen Inhalte nachfolgend kurz dargestellt.

Er umfasst neben der Definition der SPS-Programmiersprachen grundlegende Konzepte und Bestimmungen zum Aufbau von Steuerungsprojektierungen.

Zur Erstellung eines Steuerungsprogramms existieren drei verschiedene Bausteintypen, die auch als Programmorganisationseinheiten (POE) bezeichnet werden und nachfolgend kurz beschrieben sind:

- Programm (PROG): Hauptprogramm mit Zuordnung der SPS-Peripherie, globalen Variablen und Zugriffspfaden
- Funktionsbaustein (FB): Baustein mit Ein- und Ausgangsparametern und mit statischen Variablen, der bei Programmerstellung hauptsächlich Verwendung findet
- Funktion (FUN): Baustein zur Erweiterung des SPS-Operationsvorrats ohne statische Variablen

Eine Programmorganisationseinheit besitzt einen strukturellen Aufbau, der für alle drei Bausteintypen gleich ist. Dieser umfasst (siehe Abbildung 19)

- die Angabe des POE-Typs mit POE-Namen,
- einen Deklarationsteil, der die Variablendeklarationen beinhaltet, die dann im Anweisungsteil verwendet werden, und
- einen Anweisungsteil, der die Programmierung enthält.

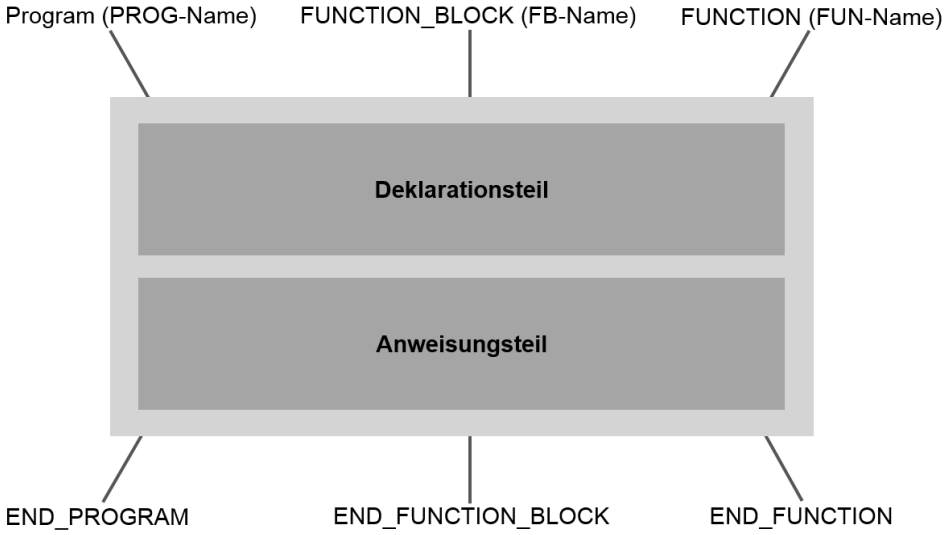


Abbildung 19: Darstellung des einheitlichen Aufbaus der drei POE-Typen „Programm“ (links), „Funktionsbaustein“ (Mitte) und „Funktion“ (rechts) [71]

Im Deklarationsteil werden alle von der POE verwendeten Variablen inklusive ihres Datentyps festgelegt. Des Weiteren können den Variablen Anfangswerte oder aber

auch physikalische Speicherbereiche zugewiesen werden. Grundsätzlich werden Variablen zur Speicherung und Weiterverarbeitung von Anwendungs- und Anwenderdaten verwendet. Dabei wird zwischen lokalen und damit nur intern sichtbaren Variablen und von außen sichtbaren Variablen (POE-Schnittstellen) differenziert.

Der Anweisungsteil (Rumpf) einer POE schließt sich dem Deklarationsteil unmittelbar an und definiert die Steuerungsaufgabe. Je nach Anwendungsbereich kann eine der fünf zur Verfügung stehenden Programmiersprachen der IEC 61131-3 verwendet werden, deren Spezifikation in der Norm IEC 61131-3 [62] nachgeschlagen werden kann:

- Ablaufsprache (AS)
- Kontaktplan (KOP)
- Funktionsbausteinsprache (FBS)
- Anweisungsliste (AWL)
- Strukturierter Text (ST)

Neben diesen fünf Programmiersprachen ist es nach IEC 61131-3 ausdrücklich erlaubt, weitere Hochsprachen zur Programmierung zu verwenden, soweit diese bezüglich folgender Bedingungen konform Verwendung finden:

- Verwendung von Variablen konform zum Deklarationsteil einer POE
- konformer Aufruf von Funktionen und Funktionsbausteinen nach IEC 61131-3
- Vermeidung jeglicher Widersprüche zu den in IEC 61131-3 spezifizierten Programmiersprachen und der graphischen Darstellungsart der Ablaufsteuerung

IEC 61499

Die Steuerungsprojektierung nach IEC 61131 fokussiert sich weitgehend auf Erstellung einzelner Steuerungsprogramme in Abhängigkeit bestimmter Ausführungsbedingungen. Der Informationsaustausch zwischen Steuerungsprogrammen erfolgt dabei über ACCESS-Variablen oder globale Datenbereiche. Für komplexe Automatisierungsaufgaben kann eine räumliche Verteilung der Steuerungslogik auf verschiedene Ausführungsressourcen sinnvoll sein. Dadurch resultieren Kommunikations- und Ausführungsstrukturen, die mit der IEC 61131 in dieser verteilten und ggf. parallelisierten Form nur umständlich projektiert werden können. Deshalb wurde die Norm IEC 61499 eingeführt, die eine Verteilung verschiedener Funktionsbausteine auf unterschiedliche physikalische Hardware und deren Synchronisation ermöglicht.

Der Hauptfokus liegt dabei auf einer konsistenten Ausführung des verteilten Steuerungsprogramms. Hierbei ist sicherzustellen, dass sowohl Datenflüsse zur lokalen Anwendungsausführung als auch der Datenaustausch zwischen verteilten Systemen zu bestimmten Zeitpunkten zur Koordinierung des verteilten Anwendungsprogramms projektiert werden können. Dies wird in der IEC 61499 durch zwei Mechanismen des Informationsaustauschs realisiert:

- Datenfluss mit Anwendungsdaten und
- Kontrollfluss zur Steuerung des dezentralen Programmablaufs.

Für die Spezifikation der Norm IEC 61499 werden folgende Begriff verwendet, die wie folgt kurz definiert werden:

- System: Das System setzt sich aus verschiedenen Geräten (Automatisierungskomponenten) zusammen und dient der Realisierung eines Prozesses.
- Gerät: Ein Gerät kann über Ressourcen zur Ausführung von Steuerungsprogrammen verwendet werden. Ein Gerät kann mehrere Ressourcen beinhalten.
- Ressource: Eine Ressource beschreibt die Ausführungsfähigkeit für eine Anwendung oder einen Teil davon.
- Anwendung: Eine Anwendung ist ein Steuerungsprogramm, das in seiner Ausführung das funktional gewünschte Verhalten des Gesamtsystems erzielt und aus einem Funktionsbausteinnetzwerk besteht. Dieses Netzwerk kann auf verschiedene Ressourcen und Geräte verteilt sein.
- Funktionsbaustein: Im Vergleich zu einem Funktionsbaustein nach IEC 61131 wurde der Funktionsbaustein nach IEC 61499 um eine ereignisdiskrete Ausführungssteuerung erweitert. Diese aktiviert den Funktionsbaustein zu bestimmten Zeitpunkten und realisiert dadurch eine konsistente Ausführung des verteilten Steuerungsprogramms.

Nach IEC 61131 wird ein PROGRAMM einem TASK zugeordnet. Diese strikte Zuordnung gilt bei der IEC 61499 nicht mehr. Hier setzt sich eine Anwendung aus Funktionsbausteinen zusammen, die im Vergleich zu Funktionsbausteinen nach IEC 61131 um eine ereignisdiskrete Ausführungssteuerung erweitert wurden.

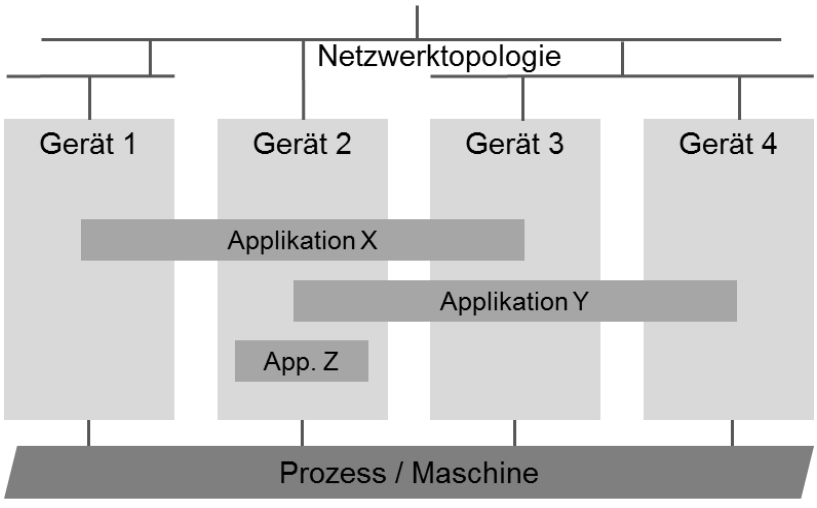


Abbildung 20: Systemmodell nach IEC 61499 [63]

Wie in Abbildung 20 dargestellt, ermöglicht dies bei Bedarf eine verteilte Ausführung von Anwendungsprogrammen auf unterschiedlichen Ressourcen bzw. Geräten, was einen wesentlichen Vorteil der IEC 61499 gegenüber der IEC 61131 ausmacht.

Die bereits angesprochene Erweiterung der Funktionsbausteine um eine ereignisdiskrete Ausführungssteuerung ist in Abbildung 21 als Gegenüberstellung zwischen IEC 61131 und IEC 61499 dargestellt.

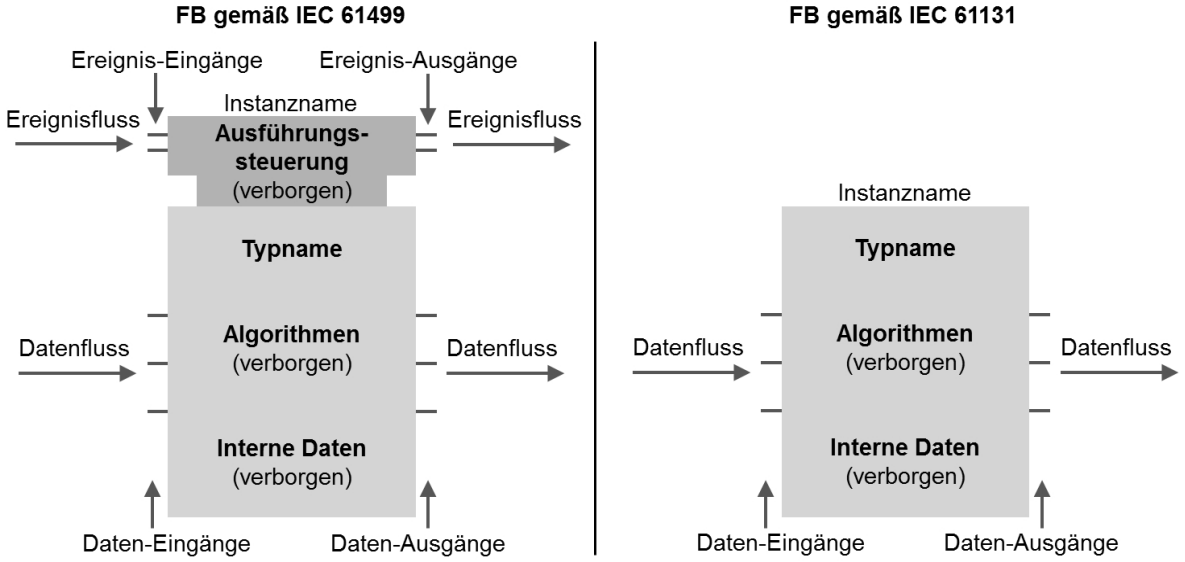


Abbildung 21: Funktionsbausteinmodell nach IEC 61499 [63]

Es ist ersichtlich, dass sich der Funktionsbaustein gemäß IEC 61499 in zwei Teile gliedert. Einerseits implementiert er bestimmte Algorithmen, die in Abhängigkeit von den anliegenden Dateneingängen bestimmte Daten an den Ausgängen erzeugen. Diese Realisierung erfolgt gemäß der Norm IEC 61131. Andererseits verfügt ein Funktionsbaustein nach IEC 61499 eine Ausführungssteuerung. Im Gegensatz zur zyklischen Ausführung der Funktionsbausteine bei Verwendung der Norm IEC 61131 entscheidet die Ausführungssteuerung der Funktionsbausteine nach IEC 61499 ereignisdiskret, ob Funktionsbausteine ausgeführt werden dürfen.

Ein weiterer wesentlicher Vorteil der Norm IEC 61499 gegenüber der Norm IEC 61131 ist die Möglichkeit der Hierarchisierung. So wird zwischen zwei Typen von Funktionsbausteinen differenziert:

- Basisfunktionsbausteine:
  Funktionsbausteine gemäß IEC 61499 stellen sich wie in Abbildung 21 dar.
- Zusammengesetzte Funktionsbausteine (siehe Abbildung 22): Funktionsbausteine gemäß IEC 61499 können Funktionsbausteinnetzwerke beinhalten, indem die Ein- und Ausgänge des Daten- und Ereignisflusses mit den Ein- und Ausgängen des Funktionsbausteinnetzwerks verbunden werden.

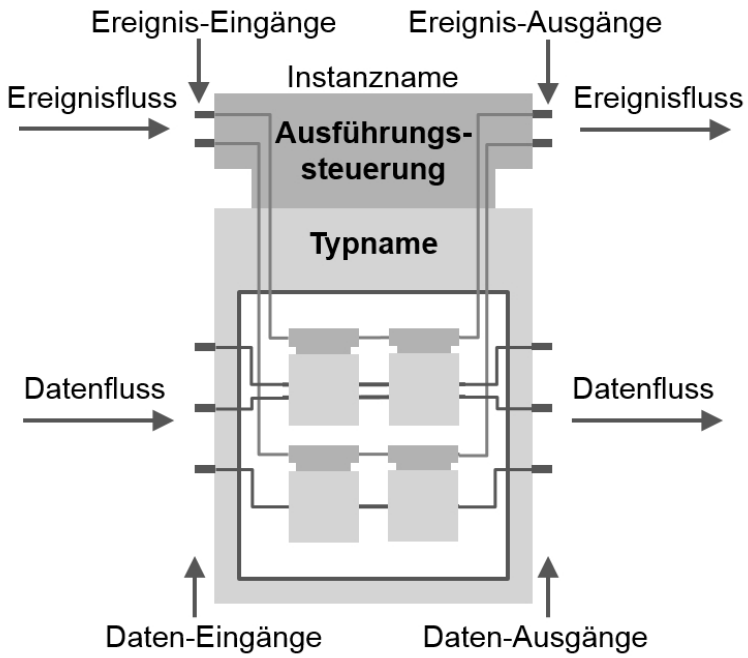


Abbildung 22: Zusammengesetzter Funktionsbaustein gemäß IEC 61499 [63]

Dieser kurze Überblick lässt erkennen, dass die Leistungsfähigkeit der Norm IEC 61131 insbesondere in dezentralen Anwendungsbereichen durch die Norm IEC 61499 ergänzt wurde. Trotz dieses zeitgemäßen Ansatzes ist die Marktdurchdringung der Norm IEC 61499 noch deutlich geringer als bei der Norm IEC 61131.

2.1.7 Diskussion der vorgestellten Standards, Normen und Konzepte und Abgrenzung zu anderen Forschungsarbeiten

Die eingangs in diesem Kapitel beschriebenen Konzepte und Methoden zur Problem- und Systemstrukturierung sowie das V-Modell zur Entwicklung mechatronischer Systeme beruhen auf einer Aufgliederung des komplexen Gesamtsystems in weniger komplexe Teilsysteme bzw. Teilbereiche, die weiter zerlegt werden können oder für die eine Lösung gefunden werden kann. Dieses Vorgehen zur Problemlösung eignet sich vor allem für Problemstellungen, die sich in unabhängige Teilprobleme bzw. Teilsysteme zerlegen lassen. Bei einem mechatronischen Systementwurf bestehen jedoch starke Abhängigkeiten zwischen den Domänen Mechanik, Elektrik und Software, die eine getrennte Bearbeitung und nachträgliche Synthese nicht erlauben. Aus diesem Grund ist ein integrierter Systementwurf erforderlich, der einen modularen Systementwurf unter Berücksichtigung der verschiedenen beteiligten Domänen ermöglicht [72].

Im Engineering gewinnen ganzheitliche Betrachtungs- und Beschreibungsansätze zunehmend an Bedeutung. Neben marktverfügbaren Lösungen wie zum Beispiel dem Teamcenter-Modul „Systems Engineering“ existieren auch Lösungen aus dem universitären Bereich wie zum Beispiel der „Mechatronic Modeller“ [73]. Dieser ist als

mechatronisches Modellierungswerkzeug in der Lage, eine mechatronische Systembeschreibung zu erstellen. Dabei findet das am Lehrstuhl für Produktentstehung des Heinz Nixdorf Instituts (Paderborn) entwickelte Konzept von Partialmodellen der Spezifikationstechnik CONSENS Anwendung [74], wobei CONSENS für „CONceptual design Specification technique for the Engineering of complex Systems“ steht.

Bei semantischen Informationsmodellen existieren verschiedene Ansätze und Methoden, die aktuell weiterentwickelt werden. Dieses Thema wird unter anderem stark aus dem Bereich des „Semantic Webs“ getrieben. Hierbei geht es um die semantische Abbildung von Informationen zum Beispiel unter Verwendung von Ontologien, um dieses Wissen dann ähnlich wie bei Datenbanken mittels semantischer Suchanfragen gezielt abfragen zu können. Ein weiteres Forschungsfeld in diesem Bereich sind „domain specific languages“ (DSL). Diese können bezüglich ihres Sprachumfangs und ihrer zugrundeliegenden grammatikalischen Regeln anwendungsfallspezifisch definiert werden. Dadurch ist es möglich, eine Sprache bezüglich der erforderlichen Beschreibungsmächtigkeit gezielt an bestimmte Einsatzzwecke anzupassen. Aus technischer Sicht sind in diesem Bereich Erweiterungen des Eclipse-Frameworks (Java-basiert) weit verbreitet. Die Sprachentwicklungserweiterung „Xtext“ zum Beispiel kann Sprachelemente sowie eine zugeordnete Grammatik definieren und ist auch in Verbindung mit graphischen Modellierungsframeworks einsetzbar.

Auf Steuerungsebene bzw. bei der Entwicklung von neuen Ansätzen zur Projektierung von Ablaufumgebungen sind die Forschungstätigkeiten etwas konsolidierter, da dies nur in enger Zusammenarbeit mit Steuerungsherstellern sinnvoll möglich ist. Es lassen sich im Bereich der Steuerungsprojektierung verschiedene Entwicklungstrends beobachten. So gewinnen zum Beispiel graphische bzw. modellbasierte Projektierungsansätze zunehmend an Bedeutung. In diesem Bereich wurde zum Beispiel in Kooperation zwischen der Firma 3S-Smart Software Solutions GmbH und dem Lehrstuhl für Automatisierung und Informationssysteme (TU München) ein Ansatz entwickelt, UML Zustandsdiagramme in eine IEC 61131-konforme Abbildung zu überführen, um diese auf der Steuerungsplattform CODESYS ausführen zu können [75]. Des Weiteren verlangen die Kunden der Automatisierungshersteller zunehmend offene Steuerungskonzepte, die eine Ausführung von benutzerspezifischem Code auf der Steuerungsplattform ermöglichen [76]. Dies ist auch dadurch bedingt, da komplexe Steuerungsalgorithmen mit Hilfe von Software-Lösungen von Drittherstellern erstellt [77] und diese Algorithmen als Programm-Code in einer Hochsprache wie zum Beispiel Ansi C oder C++ unidirektional ausgeleitet werden können. Diese werden in einem abgegrenzten Speicherbereich der SPS (Sand-Box) ausgeführt, sodass ein fehlerhaftes benutzerspezifisches Programm nicht zum Absturz oder Fehlverhalten der SPS führen kann.

Das in den nachfolgenden Kapiteln dargestellte Konzept zur semantischen Beschreibung von Automatisierungssystemen und dessen Realisierung differenzieren sich

von den oben beschriebenen Forschungsaktivitäten durch die Integration in bestehende Engineering-Werkzeuge und Methoden. Darüber hinaus wird eine konzeptionelle sowie technische Verbindung der Semantik auf Engineering- und Steuerungsebene belegt. Aktuell sind keine weiteren Forschungsarbeiten bekannt, die die Semantik des Engineerings durch Anbindung von OPC UA auf Steuerungsebene zur Verfügung stellen, um dadurch eine bidirektionale Verbindung zwischen Engineering und Laufzeitsystem herzustellen. Dies kann zu einer erheblichen Verbesserung der Unterstützung des Anlagen-Life-Cycles beitragen. Die verwendete Ablaufumgebung wurde für eine performante Interaktion mit dem semantischen Informationsmodell (OPC UA) weiterentwickelt. Darüber hinaus führt der Einsatz des dargestellten Konzepts zu einer Modularisierung des Steuerungsprogramms, was zur Verteilung des Steuerungsprogramms auf Multicore-Systemen oder dezentraler Peripherie genutzt werden kann. Das modularisierte Steuerungsprogramm kann funktional den entsprechenden Komponenten im Strukturmodell des Informationsmodells zugeordnet werden, was bisher unter Verwendung der klassischen SPS-Programmierung nicht möglich ist.

2.2 Darstellung des Handlungsbedarfs und der Zielstellung sowie Ableitung des Lösungsansatzes zur Verbesserung der IT-Integration zwischen Anlagen-Engineering und Steuerungsebene

In den nachfolgenden Abschnitten wird der Handlungsbedarf anhand verschiedener Aussagen prägnant formuliert und anschließend näher erläutert. Daran anknüpfend werden die Zielstellung und der grundlegende Lösungsansatz dieser Arbeit beschrieben.

Steuerungsprogramme von Produktionssystemen sind inhaltlich oftmals schlecht nachvollziehbar und ungenügend strukturiert

Heutige automatisierte Produktionssysteme sind in vielen Fällen in eine Zellenstruktur gegliedert, wobei jede Zelle einen oder auch mehrere Prozesse realisiert. Durch die Verknüpfung mehrerer Zellen können komplexe Aufgaben umgesetzt werden, wodurch der zeitliche Projektierungsaufwand jedoch ebenfalls deutlich steigt. Bei Betrachtung der Steuerungsarchitektur kommt je Zelle in der Regel eine zentrale Speicherprogrammierbare Steuerung (SPS) und optional dezentrale Peripherie zum Einsatz. Mit Hilfe von Projektierungssystemen wird ein SPS-Programm aus dem impliziten Wissen des Projektierungsingenieurs erstellt, das als Steuerungsprogramm auf einer SPS läuft. Dabei obliegt es dem Projektierungsingenieur, wie er das entwickelte Steuerungsprogramm strukturiert und bezüglich des gewünschten Verhaltens dokumentiert. Die Anlagenstruktur des Produktionssystems wird oftmals bei der Strukturierung der Steuerungslösung nicht berücksichtigt.

Steigende Komplexität der Produktionssysteme erhöhen den Planungsaufwand

Anspruchsvolle Produkte in hoher Qualität werden durch innovative Prozesse und Technologien gefertigt. Zunehmende Miniaturisierung, steigende Variantenvielfalt sowie die steigende Komplexität der Funktionalitäten der Produkte stellen dabei weitere Komplexitätstreiber dar. Hierfür geeignete Produktionssysteme erfordern einen hohen Investitionsaufwand, was auf die zunehmende Integration von Automatisierungstechnik sowie auf einen verstärkten Einsatz von Sensor- und Messtechnik zur Zustandsüberwachung von Prozess, Maschine und Produkt zurückzuführen ist. Aufgrund der zunehmenden Integration von Automatisierungstechnik wird eine frühzeitige Korrektur von Prozessparametern ermöglicht, um höchste Qualitätsanforderungen erfüllen zu können. Darüber hinaus kann eine bedarfsgerechte Wartung erfolgen, um die Prozessanforderungen umsetzen zu können sowie Schäden an der Maschine zu vermeiden. Daraus resultiert eine nachhaltige Reduktion des Ausschusses und der ungeplanten Maschinenausfälle. Diese Anforderungen begründen die zunehmende Komplexität von Produktionssystemen, was insbesondere Einfluss auf deren Projektierungsaufwand bei Erstellung und Änderung hat.

Konsequente und hochauflösende Überwachung von Prozessparameter führt zu stetig steigendem Kommunikationsaufwand

Die zunehmende Integration von Sensorik und Aktuatorik in moderne Produktionssysteme wirkt sich auch auf die Art der Informationsverarbeitung und Kommunikation aus. Im Bereich der Informationsverarbeitung ist eine zunehmende Dezentralisierung und Vorverarbeitung von Sensordaten zu beobachten. Dies bietet sich insbesondere bei Grenzwertüberwachungen von Messgrößen an, die nicht echtzeitfähig in ein Steuerungsprogramm eingebunden sind. Der erzielbare Vorteil liegt in einer Modularisierung der Steuerungsarchitektur. Damit verbunden resultiert eine bessere Beherrschbarkeit der Komplexität und eine ereignisdiskrete Verarbeitung der Sensorinformation in der Steuerung einer Zelle. Die Kommunikation in heutigen automatisierten Produktionssystemen wird überwiegend statisch projektiert. Das bedeutet, dass die auf einem Übertragungsmedium zur Verfügung stehende Bandbreite für essentiell wichtige Kommunikationsbeziehungen garantiert und damit auf dem Medium reserviert wird. Der Informationsaustausch mit geringerer Priorität kann folglich nur noch die verbleibende Bandbreite nutzen, selbst wenn die reservierte Bandbreite nur teilweise verwendet wird. Insbesondere der Entwicklungstrend in Richtung Dezentralisierung sowie ereignisdiskretem Informationsaustausch stellen neue Anforderungen an die Kommunikationsfähigkeiten von Automatisierungssystemen.

Die zunehmend flexible Wandlung von Produktionssystemen während des Life-Cycles verstärkt das Problem der Inkonsistenz zwischen Planungsdaten und Anlagenzustand

Während des Life-Cycles eines automatisierten Produktionssystems ist davon auszugehen, dass sich die Anforderungen und auch der Einsatzzweck einer Anlage grundlegend ändern. Dies kann die Organisation eines Produktionssystems oder aber auch die mechanische Struktur einzelner Maschinen betreffen. Ändert sich zum Beispiel eine Prozessreihenfolge oder auch bestimmte Parameter eines Prozesses, so ist dies bereits bei heutigen Anlagen durch flexible Materialflusssysteme und durch rezeptgesteuerte Prozesse umsetzbar (z.B. in der Halbleiterindustrie). Ist auf Grund der Änderung des Einsatzzwecks eine Anpassung zum Beispiel der geometrischen Struktur einer Maschine erforderlich, ist dies nur durch eine bauliche Maßnahme umsetzbar. Da strukturelle Änderungen während der Planungsphase in der Regel nicht genauer spezifiziert werden können, ist die Umsetzung einer begrenzten, definierten strukturellen Flexibilität, die allen Anforderungen der Zukunft gerecht wird, nicht möglich. Daher ist die Verfügbarkeit des aktuellen Projektstandes aus dem Engineering entscheidend, um gezielt die gewünschten Anpassungen planen und danach umsetzen zu können. Bei heutigen Produktionssystemen nimmt die Diskrepanz zwischen Soft- und Hardware einer Anlage und dem letzten verfügbaren Planungsstand des Engineerings stetig zu, was im Wesentlichen durch die durchgeführten, jedoch nicht mit dem Engineering synchronisierten, mechanischen und softwareeitigen Änderungen begründbar ist.

Fehlende Informationen im realen Produktionssystem bezüglich der Anforderungen an Automatisierungskomponenten in ihrem Prozessumfeld erschweren einen funktionalen Komponententausch

Ein weiterer wesentlicher Nachteil heutiger automatisierter Produktionssysteme ist die fehlende modulare und funktionale Sichtweise auf Komponenten der Automatisierungstechnik unter Berücksichtigung der Anforderungen an diese Komponenten in ihrem Anwendungsumfeld. So ist die Umsetzung einer Aufgabe in der Regel von den verwendeten Komponenten, dem Prozessumfeld sowie von deren spezifischen Einbindung in das Steuerungsprogramm der SPS abhängig. Dies bedeutet, dass eine bestimmte Komponente, wie zum Beispiel ein Ultraschallsensor, nicht durch einen ähnlichen, geeigneten Sensor funktional ausgetauscht werden kann, ohne das Steuerungsprogramm anpassen zu müssen. Besonders bei schlecht dokumentierten, komplexen Steuerungsprogrammen ist das Risiko einer fehlerhaften oder unvollständigen Einbindung von funktionsäquivalenten aber nicht typgleichen Komponenten vielen Anwendern zu groß, weshalb über den gesamten Life-Cycle einer Anlage bei einem erforderlichen Komponententausch oftmals ausschließlich Komponenten des gleichen Typs eingebaut werden.

Heutige Engineering-Werkzeuge unterstützen eine bidirektionale vertikale Integration mit dem Produktionssystem sowie eine bidirektionale horizontale Integration zwischen Engineering-Werkzeugen noch unzureichend

Für die Planung, Entwicklung und Simulation automatisierter Produktionsanlagen steht bereits heute eine Vielzahl digitaler Werkzeuge zur Verfügung. Diese ermöglichen eine Projektierung in sämtlichen Engineering-Phasen bis hin zur lauffähigen Anlage. Die verschiedenen Softwarewerkzeuge werden oftmals sequenziell aufeinander aufbauend eingesetzt, wobei der Informationsaustausch zwischen zwei Engineering-Werkzeugen mittels eines bestimmten Formats häufig unidirektional erfolgt. Daraus resultiert der verknüpfte Einsatz von Engineering-Werkzeugen, deren Aufgabe in einer Informationsanreicherung von allgemeinen Anforderungen bis hin zur konkreten Auslegung einer Anlage besteht. Dabei müssen die verschiedenen Domänen (horizontale Integration), wie zum Beispiel Mechanik, Elektrik, Informationstechnik als auch Prozesswissen über die verschiedenen Phasen des Engineerings (vertikale Integration) berücksichtigt bzw. integriert werden. Dies stellt aktuell noch eine Herausforderung dar, da die Sicherstellung einer Überschneidungsfreiheit, die Konsistenz der Informationen sowie die Darstellung der Interdependenzen zwischen mehreren Domänen über verschiedene Engineering-Phasen hinweg sichergestellt werden muss. Eine Rückübertragung von Informationen ist in der Regel nicht oder nur sehr eingeschränkt möglich.

Der Projektierungsaufwand für Planung und Realisierung von Produktionssystemen bietet durch Verbesserung der IT-Integration zwischen Hard- und Software erhebliches Einsparpotential

Die wesentlichen Kostenfaktoren bei der Total Cost of Ownership (TCO) eines automatisierten Produktionssystems sind Kosten zur Planung, Realisierung, Inbetriebnahme, Änderung während des Life-Cycles, Kosten für Betrieb, Wartung und Instandsetzung und Kosten für Rückbau und Recycling. Durch Methoden zur Bewertung und zum Vergleich von Investitionen [78; 79] kann eine industriespezifische Abschätzung der Kosten-Nutzen-Verhältnisse vorgenommen werden.

In der folgenden Abbildung ist die Kostenstruktur bei der Planung und Realisierung von automatisierten Produktionssystemen des Automobilbaus dargestellt. Hier fällt auf, dass die Projektierung der Hard- und Software mit 50% den größten Kostenanteil ausmacht [80].

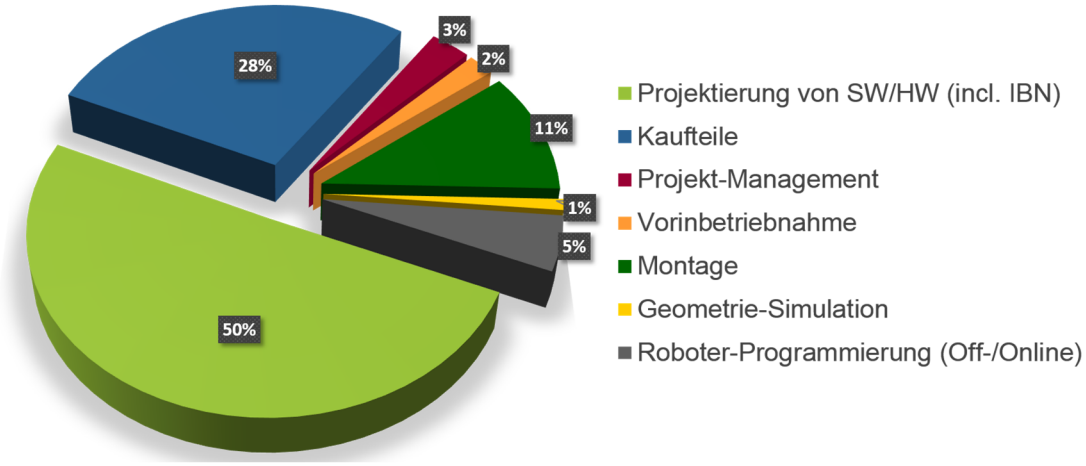


Abbildung 23: Kostenstruktur von Investitionskosten im Bereich der Steuerungstechnik [80] zitiert nach [81, S. 4]

Somit wird klar, dass Verbesserungen in diesem Bereich große Einsparpotentiale bieten. Deshalb soll der Fokus dieser Arbeit auf der Entwicklung eines Konzepts zur Verbesserung des Zusammenspiels von Softwarewerkzeugen im Engineering und auf der Verbesserung der Interaktion zwischen Engineering und realer Anlage liegen.

Darstellung der Zieldefinition

Die grundlegende Zielsetzung dieser Arbeit ist eine nachhaltige Beherrschbarkeit der zunehmenden Komplexität von automatisierten Produktionssystemen. Dabei sollen Komponenten und Werkzeuge eingesetzt werden, die in einem komplizierten Umfeld mit einer Vielzahl an Interdependenzen hochwertige, kontextbezogene Funktionen zur Verfügung stellen. Dies verspricht Effizienzsteigerungen im Bereich der Planung, der Anpassung und des Betriebs von automatisierten Produktionssystemen. Dies lässt sich nur durch eine Kombination von Maßnahmen in unterschiedlichen Bereichen erzielen, die im Zusammenspiel zu der geforderten nachhaltigen Effizienzsteigerung führen können.

So ist es zum Beispiel erforderlich, dass die Wandlungsfähigkeit von automatisierten Produktionssystemen durch einen modularen Aufbau, der neben der Mechanik auch die Domänen Elektronik und Software berücksichtigt, besser unterstützt wird. Dies ist die Grundlage, um einen funktionalen Komponententausch zu realisieren. Dabei wird eine Komponente an definierten Schnittstellen (Mechanik, Elektrik und Software) durch eine funktional gleich- oder höherwertige, jedoch nicht typgleiche Komponente ersetzt. Soweit sich auszutauschende Komponenten und Ersatzkomponenten in ihren Eingangs-, Ausgangs- bzw. Messgrößen unterscheiden, ist dies durch geeignete Maßnahmen zu berücksichtigen.

Darüber hinaus bildet eine funktionale, komponentenbasierte Modularisierung der Software die Grundlage für die Verteilung eines Steuerungsprogramms auf verschie-

dene Ressourcen, die Speicher, Rechenleistung bzw. Bandbreite zur Kommunikation zur Verfügung stellen können. Dies ermöglicht unter anderem eine gerätenahe Datenvorverarbeitung verknüpft mit einer ereignisbasierten Interaktion mit der Steuerungsebene. Im Vergleich zu einer zyklischen Datenverarbeitung einer SPS stellt dies einen wesentlichen Unterschied dar.

Dieser Aspekt der Verteilung des Steuerungsprogramms impliziert bestimmte Anforderungen an die zugrundeliegende Kommunikations- und Informationsarchitektur. Eine Verteilung bzw. Zusammenführung von Teilen des Steuerungsprogramms kann nur funktionieren, wenn das Kommunikationssystem einerseits bestimmte Fähigkeiten, wie zum Beispiel Priorisierung von Informationsflüssen zwischen Kommunikationspartnern, besitzt und andererseits bestimmte Rahmenbedingungen definiert sind. So muss zum Beispiel bekannt sein, welche Kommunikationsbeziehungen nach Verschiebung von Programmteilen eines Steuerungsprogramms auf andere Ausführungsplattformen (Ressourcen mit Rechenleistung) etabliert werden. In diesem Zusammenhang sind auch die Anforderungen bezüglich *Quality of Service* dieser Kommunikationsbeziehungen zu berücksichtigen. Die Anforderungen bezüglich Rechenleistung, Speicher- und Kommunikationsbedarf müssen ebenfalls mit den zur Verfügung stehenden Ressourcen abgeglichen werden, um die Funktionsfähigkeit eines verteilten Steuerprogramms zu gewährleisten. Darüber hinaus soll ein flexibler Aufbau von Kommunikationsbeziehungen zur Laufzeit möglich sein, um eine dynamische M2M-Interaktion zu realisieren. Dies kann zum Beispiel durch die Umsetzung von Consumer- / Providerkonzepten zur dynamischen Bereitstellung bzw. Nutzung von Information und Funktionalität umgesetzt werden.

Ein weiteres Ziel ist eine engere Verknüpfung der Planungsdaten automatisierter Produktionssysteme mit der realen Anlage über deren gesamten Life-Cycle. Dies erfordert Fähigkeiten der Abbildung und Synchronisierung von Struktur und Verhalten zwischen realer und digitaler Anlage. Hierfür sind Methoden und Wege zu entwickeln, um die Konsistenz des Planungsstandes mit dem Zustand der realen Anlage zu gewährleisten bzw. Mittel und Wege bereitzustellen, um dies bestmöglich zu unterstützen.

Die Umsetzung der Maßnahmen hat zum Ziel, die Planungskosten zu verringern, die den Hauptanteil an den Gesamtkosten der Investitionskosten im Bereich der Steuerungstechnik ausmachen (vgl. Abbildung 23). Durch Verbesserung der Funktionalität von Hardwarekomponenten und Softwarewerkzeugen sollen insbesondere Personalkosten durch Verringerung der erforderlichen Aufwände eingespart werden. Dies kann noch weiter unterstützt werden, indem die verschiedenen Planungsphasen sowie die unterschiedlichen beteiligten Domänen enger verzahnt werden und damit gegebenenfalls Arbeit auch parallel ausgeführt werden können, die heute noch sequentiell erledigt werden. Der Kostenanteil für eingesetzte Hard- und Software könnte aufgrund der erforderlichen Entwicklungsarbeit hingegen ansteigen, wobei die erwar-

teten Einsparungspotentiale bei den Planungskosten im Vergleich zum prognostizierten Kostenanstieg im Bereich der Hard- und Software deutlich höher sein werden.

Ableitung des Lösungsansatzes

Aus den oben genannten Zielen lassen sich allgemeine Aufgabenstellungen ableiten, die für eine erfolgreiche Umsetzung der Zielstellung adressiert werden müssen. Um dem grundlegenden Problem der Komplexitätsbeherrschung zu begegnen, bietet sich unter anderem eine zunehmende Modularisierung an, um Problemstellungen in Teilproblemstellungen zu untergliedern. Um dies im Umfeld der Automatisierungstechnik sinnvoll durchführen zu können, müssen die Domänen Mechanik, Elektrik und Software bei einer solchen Modularisierung berücksichtigt werden. Um die Kommunikation zwischen den verschiedenen Domänen zu vereinfachen, wäre eine einheitliche, übergreifende Terminologie für die Automatisierungstechnik zielführend, um Missverständnisse aufgrund unterschliedlicher Vorstellungen bei gleichen Begrifflichkeiten zu vermeiden.

Bei Betrachtung des Vorgehens während der verschiedenen Phasen im Engineering bis zur Inbetriebnahme der realen Anlage ist zu beobachten, dass manuelle Eingriffe an verschiedenen Stellen der Engineering-Toolchain, wie zum Beispiel Zuweisungen oder aber auch Referenzierungen, vorgenommen werden müssen, da die erforderliche Semantik zur besseren Unterstützung der Projektierungsingenieure fehlt. So werden zum Beispiel Anforderungen oftmals als Freitext hinterlegt, wodurch eine automatisierte Überprüfung bezüglich der Erfüllung einer solchen Anforderung nicht möglich ist. Durch die Möglichkeit einer standardisierten formalen semantischen Beschreibung von Anforderungen an Systemmodule sowie Leistungsfähigkeit von Komponenten ist hier zukünftig eine wesentliche Verbesserung zu erwarten, die auch einen erheblichen Einfluss auf das Szenario eines funktionalen Komponententauschs haben wird. Bei Aufweitung des Fokus von Komponentenebene auf Anlagenebene kann dargestellt werden, dass die semantische Beschreibung von Automatisierungssystemen, die sowohl im Engineering als auch auf Feld- und Steuerungsebene verfügbar ist, neue umfassende Möglichkeiten bietet. So kann durch ein gemeinsames semantisches Verständnis von Engineering-Werkzeugen und Projektierungen realer automatisierter Produktionssysteme die Kopplung zwischen realem Produktionssystem und Engineering wesentlich erleichtert werden. Darüber hinaus ist die Anbindung von Simulationswerkzeugen an das reale Produktionssystem zum Beispiel mit dem Ziel einer betriebsbegleitenden Simulation deutlich einfacher, da die vorhandene und im Produktionssystem hinterlegte Semantik interpretiert und für einen automatisierten Modellaufbau genutzt werden kann.

Im weiteren Verlauf dieser Arbeit wird ein Konzept zur semantischen Beschreibung von Automatisierungssystemen vorgestellt. Der Fokus liegt dabei auf der Beschreibung von Struktur und Verhalten mechatronischer Systeme mit dem Ziel, die semantische Beschreibung aus dem Engineering in die Steuerungs- und Feldebene zu

überführen. Durch die Verwendung einer gemeinsamen Semantik wird die Umsetzung einer bidirektionalen Kopplung zwischen Engineering und Produktionsanlage dargestellt. Darüber hinaus ist die Steuerungslogik, die das Verhalten eines mechatronischen Systems beschreibt, derart modular aufgebaut, dass die verschiedenen Teile der Steuerungslogik auf unterschiedliche Ressourcen ausgeführt werden können, soweit die erforderlichen Rahmenbedingungen wie zum Beispiel Anforderungen an Kommunikationslatenzen eingehalten werden.

Der Forschungsbereich dieser Arbeit erstreckt sich vom Anlagen-Engineering über die Steuerungs- bis hin zur Feldebene. Deshalb wurden auf den verschiedenen Ebenen existierende Beschreibungsstandards mit der Fragestellung analysiert, wie heute Informationen in der Planungs- und Realisierungsphase von automatisierten Produktionssystemen abgebildet sowie horizontal und vertikal ausgetauscht werden können. Diese gesammelten Erkenntnisse konnten in die Entwicklung des Konzepts zur semantischen Beschreibung von Automatisierungssystemen mit einbezogen werden.

2.3 Zusammenfassung und Einordnung in den Kontext dieser Arbeit

In den vorausgehenden Abschnitten sind Standards und Normen sowie Konzepte und Methoden dargestellt, die zur Strukturierung und Beschreibung von automatisierten Produktionssystemen eingesetzt werden können. Dabei haben verschiedene Standards einen unterschiedlichen Fokus und sind deshalb für bestimmte Einsatzzwecke besser, für andere schlechter geeignet.
Zur Einordnung dieser Arbeit sind sowohl die Konzepte und Methoden zur Entwicklung von Lösungsansätzen als auch der Überblick über die verschiedenen Beschreibungs- und Programmierstandards hilfreich. Die Konzepte und Methoden zur Entwicklung von Lösungsansätzen verwenden einen Ansatz zur Zerlegung einer Problemstellung in Teilproblemstellungen, die dann im Idealfall lösbar sind, andernfalls weiter zerlegt oder variiert werden müssen. Diese Herangehensweise wird vor allem bei der mechanischen Konstruktion eingesetzt und bietet Potential für eine funktionale Erweiterung aus Sicht der automatisierungstechnischen Funktionalität einer Maschine, Teilanlage oder Anlage. Die aufgezählten Beschreibungsstandards verwenden unterschiedliche Beschreibungsparadigmen und Sichtweisen, um den unterschiedlichen Anforderungen und Zielen der jeweiligen Standards gerecht zu werden. Das Spektrum der Betrachtung erstreckt sich von Ansätzen zur allgemeinen Modellbildung über Konzepte zur abstrakten Metamodellierung bis hin zur Programmierung von speicherprogrammierbaren Steuerungen. Für diese Arbeit ist in diesem Kontext insbesondere die Erstellung eigener semantischer Modellierungselemente wichtig sowie die Einsatzfähigkeit für eine technische Realisierung.
Die vorgestellten Standards, Normen, Konzepte und Methoden dienen als Grundlage zur Abgrenzung zu anderer Forschungsarbeiten. Abschließend wird der Handlungsbedarf abgeleitet, die Zielstellung definiert und ein grundlegender Lösungsansatz dargestellt, der in den weiteren Kapiteln detailliert dargestellt ist.

3 Entwicklung eines Konzepts zur semantischen Beschreibung von Produktionssystemen

Die Anforderungen an die Projektierung und den Betrieb zukünftiger Produktionssysteme ist hauptsächlich motiviert durch den Aspekt der Kostensenkung unter Betrachtung der TCO. Dabei sollen insbesondere die Planungskosten, Erstellungskosten, Hardwarekosten, Betriebs- und Wartungskosten sowie die Kosten für den Rückbau reduziert werden. Diese Kosten erstrecken sich über den gesamten Lebenszyklus einer Anlage, der in folgende Phasen untergliedert werden kann:

- Planung / Projektierung
- Inbetriebnahme
- Betrieb / Wartung
- Umbau / Rückbau / Recycling

Ausgehend von dem heutigen Stand der Technik sollen Maßnahmen umgesetzt werden, um Effizienzsteigerungen sowohl in den jeweiligen Life-Cycle-Phasen als auch bei der TCO zu erzielen.

Ein grundlegendes Konzept und Forschungsthema ist die Wandlungsfähigkeit von Produktionssystemen [82; 83]. Hier ist das Ziel, Produktionssysteme zu befähigen, sich schnell an ein veränderndes Umfeld anzupassen. Wie in Abbildung 24 dargestellt, werden unter dem Begriff der Wandlungsfähigkeit fünf Wandlungsbefähiger subsummiert.

Diese Wandlungsbefähiger stellen Kernelemente zur Umsetzung der Wandlungsfähigkeit dar und abstrahieren von der individuellen Ausprägung der Realisierung. Für die Realisierung der Wandlungsbefähiger besteht noch umfangreicher Forschungs- und Entwicklungsbedarf. Das Spannungsfeld besteht hier in dem Wunsch einer generischen Realisierung der Wandlungsbefähiger. Dem entgegen steht die Diversität der industriellen Produktion unter Einsatz unterschiedlichster Technologien, die in vielen Fällen eine spezialisierte und individuelle Betrachtung erfordern.

Angelehnt an die Forschungsarbeiten im Bereich der wandlungsfähigen Produktionssysteme werden auch Ansätze zur Rekonfiguration und Umrüstung von Produktionssystemen diskutiert. Die Fähigkeit der Rekonfiguration einer Maschine beruht auf deren modularen Aufbau und der Möglichkeit, die Module im Rahmen einer Umrüstung in vertretbarer kurzer Zeit zu wechseln. Damit können hochspezialisierte Maschinen einen deutlich höheren Grad der Flexibilität des Einsatzspektrums erreichen. [82, S. 97]

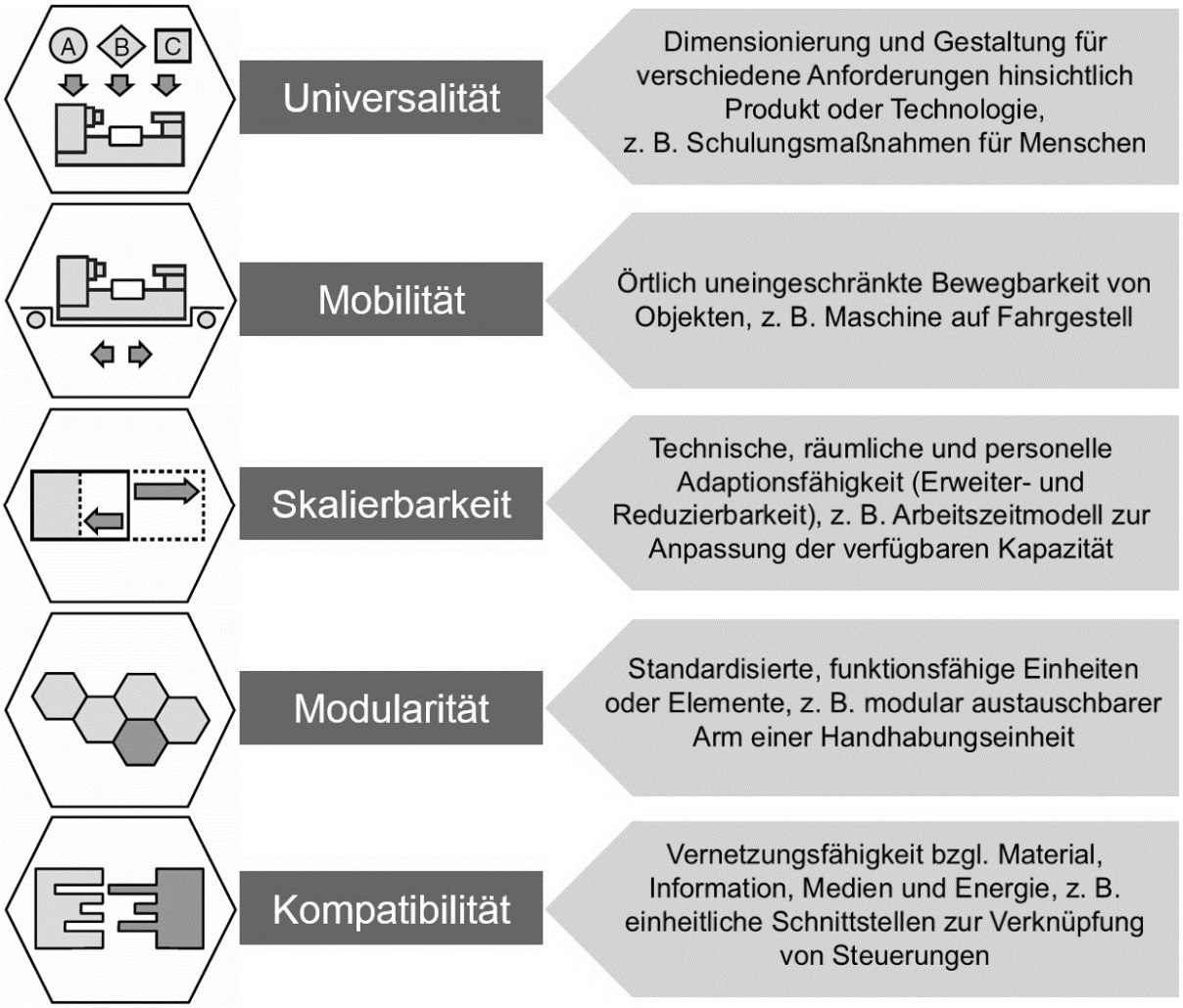


Abbildung 24: Darstellung der Wandlungsbefähiger nach Nyhuis et al. [82; 84]

Es werden insbesondere folgende Hauptanforderungen an die Projektierung und den Betrieb zukünftiger Produktionssysteme gestellt:

- Vereinfachung der Projektierung im Engineering: Durch geeignete Maßnahmen soll der Aufwand für die Planung und Erstellung von Produktionssystemen gesenkt werden. Hier stehen Konzepte im Vordergrund, die im Engineering eine ganzheitliche Projektierung der Mechatronik (Mechanik, Elektrik und Software) bzw. der Topologie/Geometrie, Kinematik sowie des Verhaltens einer Maschine ermöglichen.

- Verkürzung der Inbetriebnahme: Die Inbetriebnahmephase soll nachhaltig verkürzt werden, indem die Informationen aus der Planungs- und Projektierungsphase genutzt werden. Ziel ist eine enge Verzahnung zwischen IBN und Engineering, um Informationen bidirektional austauschen zu können.

- Unterstützung während der Betriebsphase der Anlage: Die Unterstützung während der Betriebsphase einer Anlage bezieht sich auf die Nachverfolgbarkeit von Änderungen während des Life-Cycles der Anlage und auf die Syn-

chronisierung mit dem Engineering. Des Weiteren soll ein funktionaler Komponententausch unterstützt werden.

- Flexibilisierung in der Nutzung der Hardwareressourcen: Die Dezentralisierung von Steuerungslösungen wird voraussichtlich in Zukunft zunehmen. Aus diesem Grund sind Konzepte erforderlich, die ein Steuerungsprogramm so beschreiben, dass dieses flexibel auf verschiedene Ressourcen verteilt werden kann.

- Reduzierung der benötigten Kommunikationsbandbreite: In Bezug auf die Kommunikation sollen in der diskreten FA ereignisorientierte Ansätze etabliert werden, sodass der Kommunikationsbedarf auf das erforderliche Maß ohne zyklische Übertragung redundanter Information reduziert werden kann.

In Anlehnung an ISO 9126 [85, S. 3] müssen darüber hinaus auch folgende nichtfunktionale Anforderungen erfüllt werden, um ein für die Automatisierungstechnik geeignetes Architekturkonzept zu entwickeln:

- Zuverlässigkeit (Reliability)
- Bedienbarkeit (Usability)
- Effizienz (Efficiency)
- Wartbarkeit (Maintainability)
- Portierbarkeit (Portability)

So sind Maßnahmen erforderlich, die diese Anforderungen erfüllen können. Gegebenenfalls muss die Methodik und deren Realisierung im Engineering angepasst werden. Dies umfasst vor allem die Strukturierungsmechanismen sowie die verschiedenen Sichten der bei der Projektierung beteiligten Domänen. Des Weiteren ist der semantische Informationsaustausch zwischen Engineering und Steuerungs- bzw. Feldebene zu spezifizieren, um zukünftig Semantik des Engineerings zum Beispiel direkt auf Komponenten speichern zu können. Dies kann die Konsistenz des Anlagenzustands mit den Planungsdaten während des Anlagen-Life-Cycles verbessern. Außerdem sind Kommunikationsmechanismen erforderlich, die sowohl im Engineering und der Laufzeitumgebung des Produktionssystems als auch zur Übermittlung semantischer Projektierungsinformationen verwendet werden können.

3.1 Semantische Beschreibung von Automatisierungssystemen während des Engineerings bis hin zum lauffähigen System

Zur semantischen Beschreibung von automatisierten Produktionssystemen während der Phasen des Engineerings bis hin zum laufenden System werden Beschreibungselemente sowie eine Methodik benötigt, die eine sukzessive Projektierung nach dem Prinzip einer schrittweisen Informationsanreicherung unterstützen [86].

Nach dem heutigen Stand der Technik stellt sich die Projektierung und (virtuelle) Inbetriebnahme (vIBN / IBN) von automatisierten Produktionssystemen auf einer abstrahierten Ebene wie in Abbildung 25 dar.

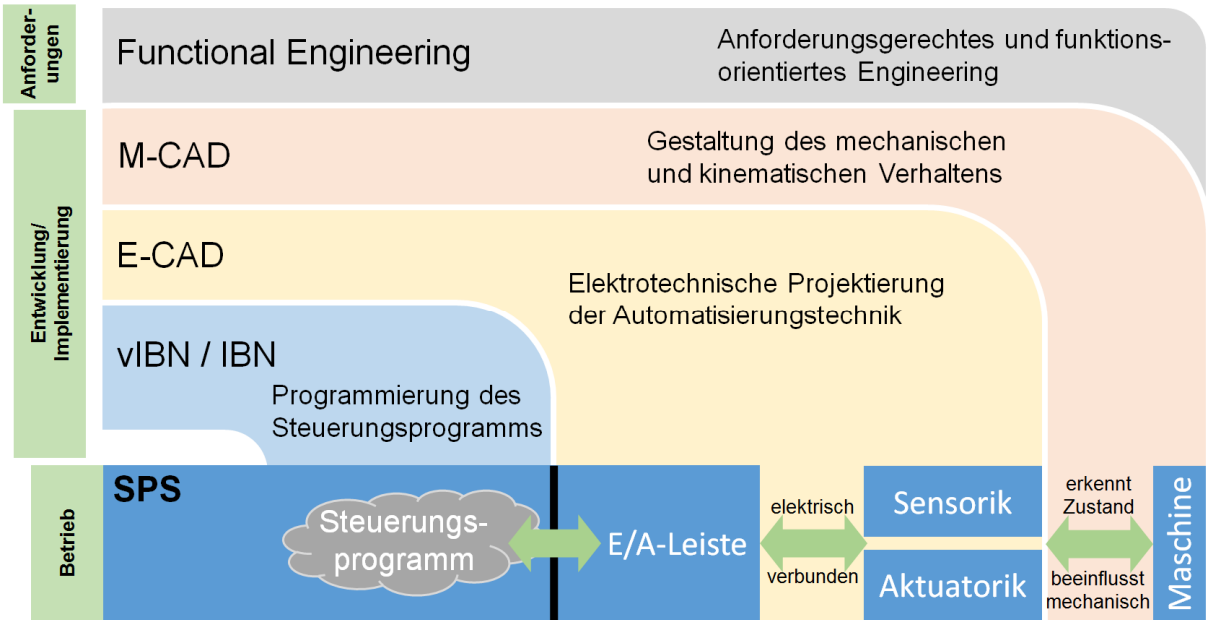


Abbildung 25: Engineering-Phasen und deren Beitrag zur Realisierung

Ausgehend von einem anforderungsgerechten, funktionsorientierten Engineering, erfolgt die Realisierung durch die mechanische und elektrotechnische Konstruktion. Die mechanische Konstruktion muss ein Modell entwickeln, das sowohl alle genannten Anforderungen als auch die gewünschten Funktionen erfüllt. Während der elektrotechnischen Projektierung werden alle elektrischen Verbindungen der Automatisierungskomponenten (Steuerungstechnik, Sensorik, Aktuatorik) definiert. Bei korrekter Verkabelung wird dadurch die grundsätzliche Funktionsfähigkeit der Maschine hergestellt. Darauf aufbauend implementiert das Steuerungsprogramm das gewünschte Ablaufverhalten der Maschine im Sinne des Prozesses.

Aus informationstechnischer Sicht existieren bei dem in Abbildung 25 dargestellten Vorgehen mehrere Informationsbrüche, die ein durchgängiges Engineering erschweren.

Die mechanische Konstruktion muss funktionale sowie nicht-funktionale Anforderungen aus dem Requirements Engineering erfüllen und verwendet Sensorik bzw. Aktuatorik, um erforderliche Fähigkeiten einer Maschine zu realisieren. Der Entscheidungsprozess und somit das implizite Wissen des Konstrukteurs wird heute semantisch nicht beschrieben. Deshalb ist anhand der Engineering-Daten nicht nachvollziehbar, aufgrund welcher Beweggründe technische oder technologische Designentscheidungen getroffen wurden.

Die elektrotechnische Konstruktion projektiert die elektrische Verbindung zwischen Komponenten und Gewerken. Bei der Auswahl von Komponenten müssen verschiedene Aspekte, wie zum Beispiel Bauraum, maximal mögliche Kabellänge der Signalleitung, Art des Kabels (Aderanzahl, Einsatz bei Kabelschlepp, Schirmung, Biegeradius), berücksichtigt werden, was eine enge Verzahnung von E-CAD und M-CAD bedarf. Auch hier ist es erforderlich, dass die Rahmenbedingungen für bestimmte Designentscheidungen semantisch beschrieben werden können und übergeordnet zwischen den Domänen „Mechanik“ und „Elektrik“ zur Verfügung stehen. Des Weiteren stellt E-CAD durch die zur Verfügung gestellten Informationen das Bindeglied zwischen Komponente und E/A-Leiste einer SPS dar. Bisher fehlt jedoch der direkte Bezug zwischen den zur Verfügung gestellten Funktionalitäten einer Komponente und deren funktionaler Verwendung im Steuerungsprogramm einer SPS.

Nach aktuellem Stand der Technik binden Speicherprogrammierbare Steuerungen die Ein- und Ausgänge entweder über deren Adresse oder anhand eines zu definierenden Symbolnamens in das Steuerungsprogramm ein. Das Steuerungsprogramm bietet jedoch keine semantische Beschreibungsmöglichkeit der Zuordnung zu bestimmten Komponenten oder Gewerken einer Maschine. Darüber hinaus sind die Systemgrenzen im Steuerungsprogramm nicht definiert, wodurch die Adaption des Steuerungsprogramms zum Beispiel bei funktionalem Komponententausch erschwert wird.

Um diesen Hemmnissen zu begegnen, soll ein Konzept zur semantischen Beschreibung von Automatisierungssystemen entwickelt werden. Hierfür werden verschiedener Beschreibungselemente verwendet, die in den folgenden Absätzen dargestellt sind.

Als grundlegende Strukturierungsmechanismen wird eine Einteilung in Rollen und Komponenten vorgenommen, was einem Typ-/Instanzkonzept ähnelt. Ein hierarchisches Zusammenfügen mehrerer Teilkomponenten oder Rollen zu einer funktional höherwertigen Komponente oder Rolle wird mit Hilfe eines Aggregationsprinzips umgesetzt. Sowohl die Strukturierungsmechanismen als auch die semantischen Beschreibungselemente sind nachfolgend beschrieben.

Um die nachfolgenden Konzepte besser verdeutlichen zu können, werden ausgewählte Komponenten einer einfachen Handhabungseinheit (siehe Abbildung 26) als Beispiel verwendet. Ziel dabei ist es, einen Bezug zwischen der mechanischen Struktur, dem Verhalten der Handhabungseinheit und den neu definierten semantischen Beschreibungselementen herzustellen.

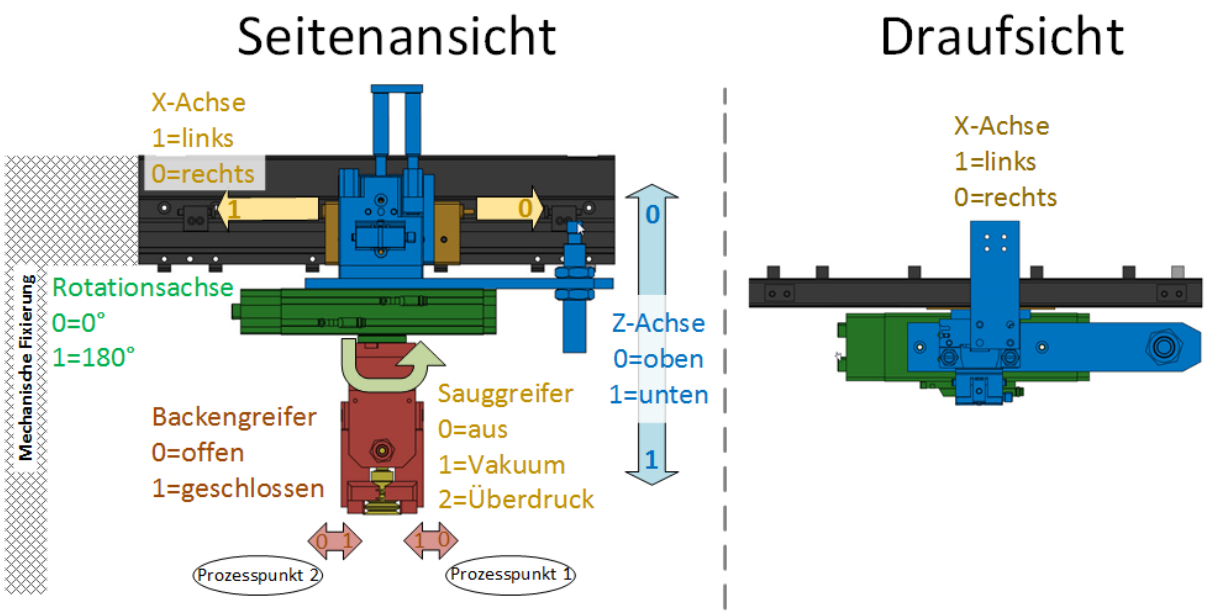


Abbildung 26: Überblick über den mechanischen Aufbau einer Handhabungseinheit

3.1.1 Art der graphischen Darstellung

Für die graphische Darstellung von Abhängigkeiten zwischen verschiedenen Entitäten wird oftmals das Entity-Relationship-Modell (ERM) verwendet. Dies beschreibt die Abhängigkeiten zwischen den aus der Wirklichkeit entliehenen Objekten (Entitäten) und deren Attributen.

Bei der Darstellung mechanischer Baugruppenstrukturen zeigt sich, dass die Verwendung von Entity-Relationship-Modellen für ein intuitives Verständnis nicht förderlich ist. Aus diesem Grund wird die Beschreibungsform entsprechend angepasst und eine graphische Darstellung entwickelt, die die Vorteile einer Komponentenhierarchie entsprechend der Baugruppenstrukturierung der mechanischen Konstruktion mit einer relationalen Darstellung verbindet. In Abbildung 27 sind zentrale Beschreibungselemente der nachfolgenden Abschnitte abgebildet.

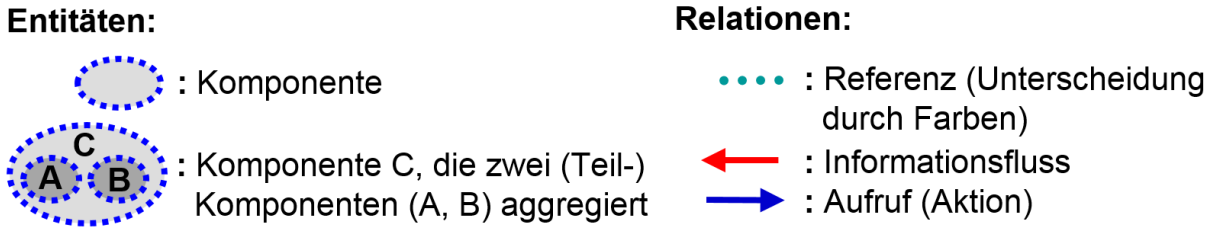


Abbildung 27: Darstellung ausgewählter Elemente für die graphische Darstellung der semantischen Beschreibung von Automatisierungssystemen

Konzeptbedingt können Komponenten beliebig hierarchisch zusammengesetzt werden. Dieser Ansatz folgt dem Baugruppenprinzip bei der mechanischen Konstruktion. Innerhalb der durch die Komponenten aufgespannten Aggregationsstruktur können

weitere Beschreibungselemente (Entitäten) eingefügt und mittels bestimmter typisierter Relationen referenziert werden. Eine spezielle Form der Relationen stellen der „*Informationsfluss“* sowie der „*Aufruf“* dar. Diese werden für die Abbildung der Interaktion zwischen Steuerungslogik und Sensorkomponenten bzw. Aktuatorkomponenten eingesetzt.

3.1.2 Einführung eines „Rollen“-Verständnisses

Rollen werden als Elemente zur Beschreibung von Anforderungen verwendet, beschreiben jedoch nicht, auf welche Art und Weise - bezogen auf die technische Umsetzung - verschiedene Komponenten miteinander interagieren. Rollen ermöglichen eine komponentenunabhängige Beschreibung der Anforderungen im Sinne des Prozesses. Jede Komponente, welche die in der Rolle definierten Anforderungen erfüllt, kann als Instanz (Komponente) der Rolle eingesetzt und mit dieser durch eine bidirektionale Relation verknüpft werden.

Im Gegensatz zu vielen Softwarelösungen des heutigen Standes der Technik geht die Projektierung mittels Rollen und ihre jeweiligen inhärenten Anforderungen auch nach der Auswahl der Instanzen (Komponenten) nicht verloren, sondern die Rollen werden in der Laufzeitumgebung des Produktionssystems persistiert. Dies stellt sicher, dass die Anforderungen an die jeweiligen Komponenten nicht verloren gehen und bildet die Grundlage für die Möglichkeit eines funktionalen Komponententauschs. Im Gegensatz hierzu wird bei einer lösungsorientierten Projektierung heutiger Engineering-Systeme die gewünschte Funktionalität bzw. die Anforderungen an eine Komponente nicht beschrieben. Beim Komponententausch wird daher meist eine baugleiche Komponente eingesetzt, da die erforderlichen Informationen zur Auswahl alternativer Austauschkomponenten oftmals fehlen. Andernfalls muss sich der Projektierungsingenieur die Anforderungen an die auszutauschende Komponente selbst erarbeiten, da die ursprünglichen Planungsinformationen häufig nicht mehr vorliegen.

3.1.3 Einführung eines „Komponenten“-Verständnisses

Die Einteilung in Komponenten dient der Strukturierung und Modularisierung komplexer technischer Systeme. Komponenten können in reale sowie in logische Komponenten untergliedert werden.

Eine reale Komponente repräsentiert eine physikalische Instanz (d.h. ein physikalisch vorhandenes Gerät zum Beispiel Sensor, SPS, …), die eine bestimmte Basisfunktionalität, wie zum Beispiel „*messen“* oder bestimmte Ressourcen, wie zum Beispiel „*Rechenkapazität“,* bereitstellt. Sie wird im Engineering einer Rolle zugewiesen. Reale Komponenten können „*Rechenkapazität“* zum Beispiel im Sinne einer Ablaufumgebung bereitstellen.

Logische Komponenten umfassen „Berechnungsvorschriften“ und benötigen daher immer „*Rechenkapazität“*, um ihre Funktionalität in der Ablaufumgebung erfüllen zu können (zum Beispiel Mittelwertbilder, Regler, Histogramm, …). Es können mehrere logische Komponenten eine Ablaufumgebung nutzen, soweit genügend „*Rechenkapazität“* zur Verfügung steht.

Die Einteilung in reale und logische Komponenten erlaubt die Abbildung von heutiger Automatisierungshardware (reale Komponenten) sowie von Programmierbausteinen / Intellectual Property (logische Komponenten) unter einem Komponentenbegriff. Somit lässt sich zwischen der physikalischen Repräsentation einer Komponente und derem Verhalten, das durch logische Komponenten realisiert wird, differenzieren.

3.1.4 Spezifikation der Basisfunktionalitäten von Automatisierungskomponenten

Eine *BaseFunction* ist ein Interface zu einer Rolle oder Komponente, das einerseits die funktionalen Eigenschaften einer Rolle oder Komponente im Sinne der zu lösenden Automatisierungsaufgabe typisiert sowie andererseits den funktionalen Zugriff auf diese ermöglicht. *BaseFunctions* können – im Gegensatz zu den Informationen im Typenschild (*NumberPlate*) – erzeugt und gegebenenfalls auch geändert werden. Die *BaseFunction* ist die Interaktionsschnittstelle für die Automatisierungsaufgabe und kann prozessbezogen, steuerungslogikbezogen sowie datenverarbeitungsbezogen typisiert werden.

3.1.5 Bildung komplexer mechatronischer Komponenten mittels Aggregation von Teilkomponenten

Die Aggregation dient als Strukturierungswerkzeug von Rollen und deren zugeordneten Komponenten im Sinne der zu lösenden Automatisierungsaufgabe. Die Aggregation erlaubt das Zusammenfassen von Rollen bzw. Komponenten zu funktional höherwertigen Funktionseinheiten, die als Aggregate wiederum nach außen Rollen oder Komponenten darstellen. Ziel der Aggregation ist eine funktionale Kapselung unterlagerter Komponenten. Dadurch müssen einzelne Komponenten nicht mehr von beliebigen Steuerungsebenen direkt angesprochen werden, sondern werden von der Steuerungslogik des Aggregats zur Realisierung der Aggregatsfunktionalität im Sinne des Prozesses angesteuert.

Ein Aggregat kann als Whitebox oder aber auch als Blackbox definiert werden. Dadurch können Teilkomponenten von den jeweiligen Herstellern unter Verwendung des Informationsmodells entwickelt und das Intellectual Property durch Anwendung der Blackboxsicht geschützt werden. Das Verständnis der Blackbox-Sicht ist vergleichbar mit dem bereits heute existierenden Bausteinschutz [87, S. 1312] von Funktionsbausteinen (Know-How-Schutz) speicherprogrammierbarer Steuerungen.

Der funktionale Zugriff auf ein Aggregat bzw. eine Komponente erfolgt über mindestens eine BaseFunction. Diese dient als funktionale Schnittstelle und wird jeweils durch Steuerungslogik implementiert.

Mittels Aggregation wird eine hierarchische Strukturierung der mechanischen Funktionseinheiten entsprechend des Fertigungsprozesses und der kinematischen Abhängigkeiten vorgenommen. Dabei werden Komponenten mittels Aggregation zu funktional höherwertigen Komponenten integriert. Dies hat zwei wesentliche Vorteile: Einerseits kann das Kommunikationsaufkommen reduziert werden, da interagierende Komponenten direkt kommunizieren können. Somit werden unstrukturierte, flache Hierarchien aufgrund nicht erfasster funktionaler und mechanischer Abhängigkeiten und die damit verbundene zentrale Verarbeitung und Steuerung aller Sensoren bzw. Aktuatoren vermieden. Andererseits erlaubt dies die Aufgliederung der Aufgabe in Teilaufgaben einzelner Komponenten sowie die Verteilung der Steuerungslogik entlang der Aggregationsebenen bis hin zu einzelnen Komponenten. Die Strukturierung mittels Aggregation sollte sich als primäres Ordnungsmerkmal an der Strukturierung des Fertigungsprozesses sowie an den kinematischen Abhängigkeiten orientieren, was jedoch keinen Zwang darstellt, sondern lediglich der Verständlichkeit dient.

3.1.6 Spezifikation der Funktionalität und des Verhaltens von Automatisierungskomponenten

Die ControlLogic realisiert die Verschaltungs- und Ablauflogik eines Aggregats und implementiert eine BaseFunction, welche die Interaktionsschnittstelle für die Automatisierungsaufgabe darstellt. Es kann mehrere ControlLogic-Elemente in einem Aggregat geben, die, ähnlich wie bei Funktionsbausteinen, „verschaltet“ werden können. Die Verschaltungslogik der ControlLogic kann frei bzw. durch logische Bausteine unterstützt programmiert werden.

Im weiteren Verlauf wird in den Abbildungen eine schematische Darstellung der ControlLogic verwendet (vgl. Abbildung 28).

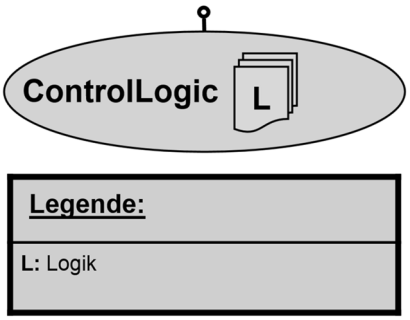


Abbildung 28: Schematische Darstellung der ControlLogic

Eine ControlLogic realisiert eine bestimmte Funktion und kann zum Beispiel von anderen ControlLogics mehrfach als Eingangsparameter verwendet werden. Ein Aggregat kann über dessen ControlLogic mit den BaseFunctions der untergliederten

Aggregate verschaltet werden. Der Hauptunterschied der ControlLogic gegenüber heutigen Funktionsbausteinen ist die maschineninterpretierbare Beschreibung der Funktionalität, die mittels einer oder mehrerer Funktionen – den BaseFunctions – zur Verfügung gestellt wird. Somit implementiert die ControlLogic die Funktionalität der Einzelkomponenten zur Realisierung der Gesamtaufgabe. Weitere Informationen über den konzeptionellen Aufbau der ControlLogic kann dem Kapitel 3.4 entnommen werden.

3.1.7 Indirektion des funktionalen Aufrufs unterlagerter Komponenten durch semantisch beschriebene Kopplungselemente (ProcessFunctionConverter)

Der ProcessFunctionConverter ermöglicht die Beschreibung der Abhängigkeiten zwischen den Funktionen der Automatisierungskomponenten und der mechanische Konstruktion. So können Abhängigkeiten zwischen funktionalem Engineering und den Komponenten der mechanischen Konstruktion (Sensorik / Aktuatorik / Mechanik) definiert werden. Dies erlaubt zum Beispiel die Beschreibung des funktionalen Verhaltens eines Zylinders, der sich aus einem Ventil (Komponente der Automatisierungstechnik) und einem mechanischen Kolben zusammensetzt. Durch die Ansteuerung des Ventils (Positionen 1, 2, 3) wird eine translatorische Bewegung des Zylinders verursacht. Der ProcessFunctionConverter beschreibt dabei die translatorische Bewegung des Zylinders in Abhängigkeit der Zeit sowie des Drucks der Druckluft. Außerdem werden im ProcessFunctionConverter die mechanischen Anbaupositionen von Sensorik bzw. Aktuatorik gespeichert (vgl. Abbildung 29). Dies ermöglicht bei der funktionalen Verwendung von Sensorik bzw. Aktuatorik die Referenzierung auf die Anbauposition der Sensor- bzw. Aktuatorkomponenten, wodurch der Zusammenhang zwischen mechanischer und funktionaler Realisierung hergestellt wird.

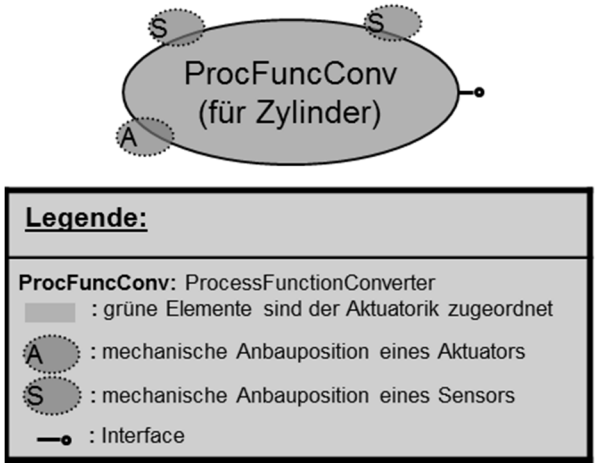


Abbildung 29: Schematische Darstellung des ProcessFunctionConverters

Andere gebräuchliche Anwendungsbeispiele für den ProcessFunctionConverter sind zum Beispiel die Umformung einer rotatorischen Bewegung eines Motors in eine translatorische Bewegung eines Förderbandes sowie die Verwendung eines Zylinders zur Umsetzung eines Greifers. In beiden Fällen können im ProcessFunction-

Converter Übersetzungsverhältnisse dargestellt werden. Dadurch können mechanische Relationen funktional sowie semantisch beschrieben werden.
Der ProcessFunctionConverter ist für das Konzept der Aggregation eine entscheidende Erweiterung, da dieser analog zur Aggregation beschreibt, wie aus mechanischer Sicht Komponenten aggregiert werden.
Am konkreten Beispiel der Handhabungseinheit (siehe Abbildung 26) mit Fokus auf den Backengreifer ist zu erkennen, dass ein Mechanismus erforderlich ist, der die Beschreibung funktionaler bzw. mechanischer Zusammenhänge in Abhängigkeit mit den Komponenten der Automatisierungstechnik ermöglicht. So wird zum Beispiel aus dem Schalten eines Ventils eine translatorische Bewegung eines Zylinders, die zum Öffnen bzw. Schließen des Backengreifers genutzt wird.

3.1.8 Definition semantisch beschriebener Zustandselemente (Interaktionspunkte) als Platzhalter für Transitionsbedingungen von Steuerungsprogrammen

Interaktionspunkte (IAP) stellen ein Bindeglied zwischen funktionaler Ebene und automatisierungstechnischer Realisierung dar. Mittels IAPs ist es möglich, bestimmte Konfigurationszustände auf Komponentenebene zu definieren, was eine funktionale Kapselung der Komponenten ermöglicht. Dies erlaubt einerseits eine Verteilung der Steuerungsinformation in einzelne Komponenten und ermöglicht andererseits die funktionale Aggregation von Komponenten zu funktional höherwertigen mechatronischen Komponenten. Im weiteren Verlauf dieser Arbeit wird in den Abbildungen eine schematische Darstellung des IAPs verwendet (vgl. Abbildung 30).

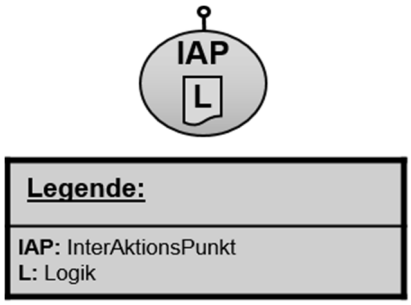


Abbildung 30: Schematische Darstellung des Interaktionspunkts

Das Konzept der Interaktionspunkte bietet den Vorteil, dass das implizite Wissen des Konstrukteurs über den Prozessablauf sowie über die Interaktion der Automatisierungskomponenten zur Realisierung des Prozesses während der Konstruktionsphase in den Interaktionspunkten gespeichert werden kann. So ist der Konstrukteur in der Lage, den Prozessablauf seiner mechanischen Konstruktion funktional zu beschreiben, ohne bereits alle Automatisierungskomponenten definiert zu haben.

Des Weiteren ist es nicht mehr erforderlich, dass in der Steuerungslogik semantisch unbeschriebene Inputs bzw. Outputs verwendet werden, sondern Interaktionspunkte funktional eingebunden werden können. Dadurch werden Interaktionspunkte als Indirektionselemente mit semantischer Bedeutung im Steuerungsprogramm verwendet.

Dies vermeidet die Verwendung von Realparametern im Steuerungsprogramm, was ein wesentlicher Hinderungsgrund bei einem funktionalen Komponententausch darstellt. Interaktionspunkte existieren in unterschiedlichen Ausprägungen:

- Semantische Beschreibung von diskreten Werten / Zuständen
- Semantische Beschreibung von Wertebereichen / Zustandsbereichen
- Semantische Beschreibung von Zustandskombinationen

Dadurch können Interaktionspunkte zum Beispiel zur Prozessbeschreibung, zur Festlegung von speziellen Bewegungsbereichen bzw. Endlagestellungen und zur Definition von Einschaltverriegelungen eingesetzt werden. Ein IAP verknüpft in sich mehrere Informationen. Einerseits dient er der semantischen Beschreibung einer bestimmten Konfiguration eines Aktuators im Sinne des Prozesses (zum Beispiel eine bestimmte Position im Raum, die kennzeichnet, wie weit eine Linearachse verfahren ist). Dabei ist neben der semantischen Beschreibung ebenfalls die erforderliche Konfiguration des Aktuators im Sinne des Prozesses im IAP enthalten. Als Beispiel kann ein pneumatischer Zylinder als Aktuator betrachtet werden. Dieser wird mittels Schaltung eines Ventils aus- bzw. eingefahren. Ein IAP, der diesem Zylinder zugeordnet ist, trägt eine semantische Beschreibung „Zylinder ausgefahren“ bzw. „Zylinder eingefahren“. In einem Interaktionspunkt muss die mechanische Konfiguration hinterlegt sein, die dem semantisch beschriebenen Zustand „Zylinder ausgefahren“ bzw. „Zylinder eingefahren“ entspricht. Dies kann entweder eine Längeneinheit (Zylinder auf 50 mm ausgefahren) oder eine Position im Raum sein, von der sich die Konfiguration des Zylinders ableiten lässt.

Andererseits können mit dem Interaktionspunkt ein oder mehrere Sensoren verknüpft werden, die das Erreichen dieser bestimmten Konfiguration des Aktuators detektieren. Dabei sind im IAP die Sensorwerte hinterlegt, die die Sensorik liefern würden, wenn der Aktuator die zu überwachende Konfiguration einnimmt. Zum Beispiel liefert ein Lagesensor ein Signal, sobald sich der pneumatische Zylinder auf der Sensorposition befindet. Ein IAP, der diesem Sensor zugeordnet ist, trägt die semantische Beschreibung „Zylinder steht auf Pos_links“. Im Interaktionspunkt muss hinterlegt sein, welcher Sensorwert der Information „Zylinder steht auf Pos_links“ entspricht.

IAPs können bei diskreten Sensorwerten (zum Beispiel zur Positionsüberwachung mittels eines Lagesensors) oder bei Sensorwertebereichen (bei zum Beispiel Überwachung des Temperaturbereichs eines Mediums) angewendet werden. Ein diskreter Sensorwert beschreibt dabei eine bestimmte, eindeutige Konfiguration, wie zum Beispiel eine bestimmte Position einer Linearachse, die zum Beispiel mittels Absolutwertgeber detektiert wird.

Ein Bereich kann in einem IAP definiert werden, soweit es das Ziel ist, einen bestimmten Freiheitsgrad auszudrücken, wie zum Beispiel „Temperatur des Mediums liegt zwischen 40° C und 50° C“. Sowohl für die mechanische Realisierung als auch

für eine Realisierung für die Prozessindustrie gibt es hier sinnvolle Anwendungsfälle. Dies ermöglicht eine Bereichsdefinition in der Prozessindustrie zum Beispiel zur Festlegung eines Temperaturbereichs oder eines Druckbereichs, der zur Prozessdurchführung erforderlich ist. Für die Automatisierungstechnik unter Berücksichtigung der Mechanik kann die Definition eines Bereichs genutzt werden, um zum Beispiel erlaubte Bewegungsbereiche der Mechanik zu definieren.

Semantisch beschriebene Wertebereiche mittels IAP:

Bewegungsbereich-IAPs stellen eine Spezialisierung des IAP dar und dienen der Festlegung von erlaubten bzw. nicht-erlaubten Bewegungsbereichen der Aktuatorik. Um die Bewegung der Aktuatorik hinreichend genau überwachen zu können, ist die Verwendung von Sensoren erforderlich. Je nach Anwendungsfall erlaubt der IAP eine spezifische Verschaltung der verwendeten Sensoren. Wird zum Beispiel eine Linearachse mit Absolutwertgeber eingesetzt, ist die Festlegung von Bewegungsbereichen mit Hilfe von IAPs möglich, um zulässige bzw. nicht-zulässige Bewegungsbereiche zu kennzeichnen. Dies erlaubt zum Beispiel die Verwendung eines IAPs in der Steuerungslogik als booleschen Wert mit der semantischen, menschenlesbaren Beschreibung „Komponente befindet sich in einem zulässigen Bewegungsbereich“.

Semantisch beschriebene diskrete Werte mittels IAP:

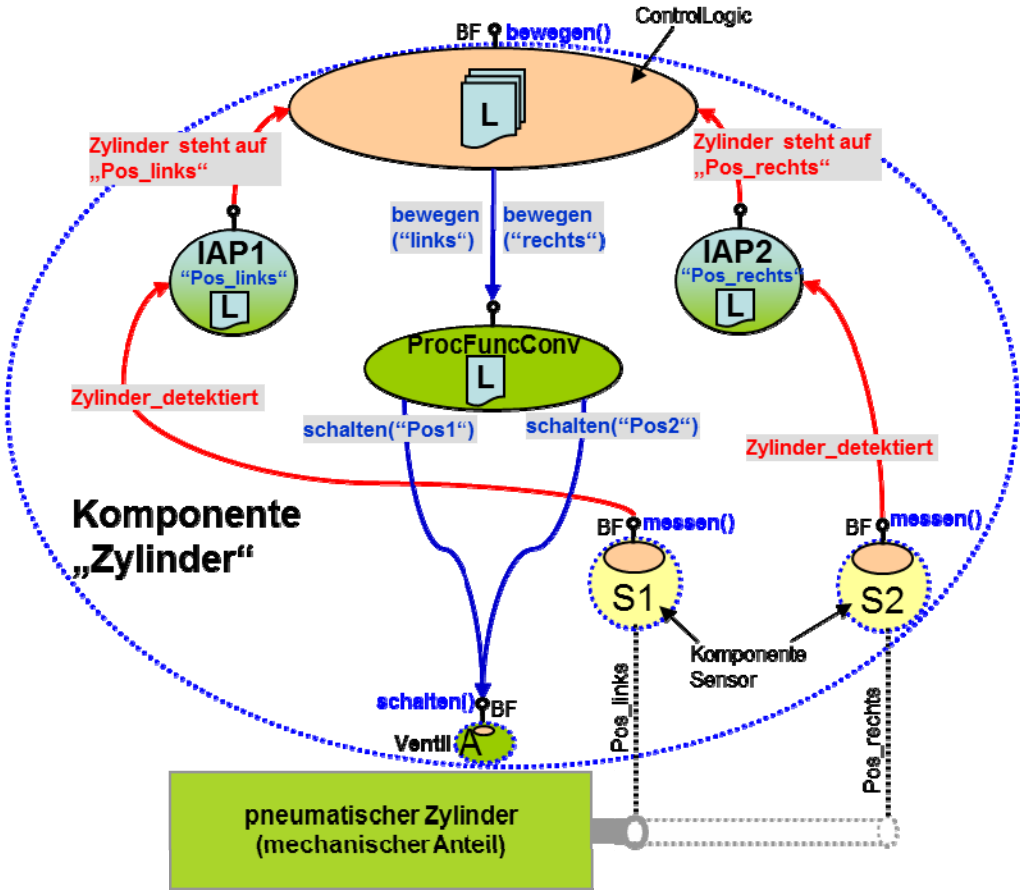
Endlagen-IAPs stellen eine Spezialisierung des IAP dar. Endlagen-IAPs können in unterschiedlichen Entfernungen zur Endlage definiert werden, um zum Beispiel das Bewegungsverhalten des Aktuators am Rande des zulässigen Bewegungsraums ggf. sogar mehrfach anzupassen.

Semantisch beschriebene Zustandskombinationen mittels IAP:

Einschaltverriegelungs-IAPs sind eine Spezialisierung des IAPs und erlauben eine semantische Beschreibung aller für die Prozessdurchführung relevanter Bedingungen. So kann es zum Beispiel bei einem Prozess erforderlich sein, dass Druckluft vorhanden sein muss und die Schutzeinrichtungen geschlossen sind. In diesem Fall gäbe es einen Interaktionspunkt „Druckluft vorhanden“ sowie den Interaktionspunkt „Schutzeinrichtungen geschlossen“, die von der Steuerungslogik als boolescher Wert Verwendung finden können. In den jeweiligen IAPs ist die Anbindung der relevanten Sensoren individuell implementiert, damit die Information, die der IAP semantisch beschreibt, aus einer definierten Datenbasis erzeugt werden kann.

Um die Interaktionspunkte mit Sensorik bzw. Aktuatorik zu verknüpfen, kann in einem IAP eine Verschaltungslogik definiert werden, mit deren Hilfe zum Beispiel die Messwerte mehrerer Sensoren zu einem Sensorwert zusammengefasst werden können. Diese optionale Verschaltungslogik in einem Interaktionspunkt differenziert sich von der Logik der ControlLogic, da sich die Logik in einem Interaktionspunkt auf eine ein-

fache arithmetische Verschaltungslogik, zum Beispiel zur Bildung des Mittelwerts mehrerer Sensorwerte, beschränkt.



Legende:	Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:
BF: BaseFunction **IAP:** InterAktionsPunkt **L:** Logik **S:** Sensor **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Informationsfluss : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet : gelbe Elemente sind der Sensorik zugeordnet : Interface : Komponente	Strukturelemente: - ControlLogic - Logik - IAP - BaseFunction - ProcessFunctionConverter - Sensor - Aktuator - Mechanische Anbaupositionen Basisfunktionalitäten: - **schalten()** / Ventil - **bewegen()** / Zylinder - **messen()** / Sensor

Abbildung 31: Beispiel zur Verwendung von IAPs in einer Komponente

In Abbildung 31 ist am Beispiel eines pneumatischen Zylinders in Kombination mit zwei Lagesensoren die Verwendung von zwei Interaktionspunkten dargestellt. Dabei sollen insbesondere die funktionale Sicht sowie die Verwendung der Interaktionspunkte verdeutlicht werden. Bei dem nachfolgend dargestellten Beispiel detektieren zwei Lagesensoren, ob sich ein Zylinder auf der Position „Pos_links“ (Zylinder um 10

mm ausgefahren) oder „Pos_rechts“ (Zylinder um 80 mm ausgefahren) befindet. Dem IAP1 ist die semantische Beschreibung „Zylinder Pos_links“ zugeordnet. Im IAP1 ist die mechanische Konfiguration des Zylinders (Zylinder um 10 mm ausgefahren) abgelegt. Außerdem ist der IAP1 mit dem Sensor S1 verknüpft sowie ein Sensorwert hinterlegt. Sollte der Sensor S1 diesen Wert annehmen, ist der Zustand, den der IAP1 semantisch beschreibt, eingetreten (Zylinder steht auf Pos_links).

Durch die Hinterlegung der mechanischen Konfiguration im IAP kann die Logik nach Aufruf der BaseFunction *bewegen(„Pos_links“)* über den IAP1 ermitteln, dass eine Positionsänderung des Zylinders auf 10 mm Ausfahrlänge gewünscht ist. Je nachdem, in welcher Konfiguration sich der Zylinder gerade befindet, muss dieser ein- bzw. ausgefahren werden. Dementsprechend erfolgt ein Aufruf *bewegen(„links“)* bzw. *bewegen(„rechts“)* des ProcessFunctionConverters (ProcFuncConv), der die Umsetzung in die jeweils erforderliche Ventilstellung vornimmt. Der Zylinder wiederum setzt sich aus einem Ventil als Aktuator sowie dem mechanischen Zylinder zusammen. Die Umwandlung der Aktuator-Funktion des Ventils (öffnen/schließen) in eine translatorische Bewegung des Zylinders erfolgt im ProcessFunctionConverter. Dieser übersetzt zum Beispiel den BaseFunction-Aufruf *bewegen(„links“)* in den Aufruf *schalten(“Pos1“)*.

Durch den Einsatz von Interaktionspunkten wird sichergestellt, dass bereits im Engineering in den Ablaufsequenzen der Logik der ControlLogic eines Aggregats semantisch beschriebene Elemente verwendet werden können, die sowohl die Parametrierung der unterlagerten Aktuatorik als auch die relevante Sensorik abstrahieren.

3.2 Darstellung der Abhängigkeiten zwischen den vorgestellten Strukturierungselementen

Ausgehend von der funktionalen Ebene als Schnittstelle zu den vorhandenen Ansätzen der funktionalen Modellierung wird in den folgenden Abschnitten mit den bereits vorgestellten Beschreibungselementen schrittweise ein Modell aufgebaut. Dieses Modell verwendet die bereits vorgestellten semantischen Beschreibungselemente sowie spezialisierte Relationen, um Struktur und Verhalten von mechatronischen Einheiten semantisch beschreiben zu können. Aus Gründen der Vollständigkeit sollen hier die Schritte im Engineering ausgehend von den allgemeinen Anforderungen bis hin zur funktionalen Ebene kurz dargestellt werden.

- Initial werden als erster Schritt im Engineering allgemeine Anforderungen an die zu projektierende Anlage spezifiziert. Als Beispiel hierfür könnte eine bestimmte herzustellende Stückzahl als minimaler Durchsatz pro Stunde als allgemeine Anforderung an die Anlage gestellt werden.

- Als zweiter Schritt werden im Engineering technologie- und realisierungsunabhängige Prozessanforderungen definiert. So könnte hier zum Beispiel fest-

gelegt werden, dass bei einem bestimmten Prozess eine Glasflasche mit einem Deckel verschlossen werden soll.

- Als dritter Schritt im Engineering erfolgt eine realisierungsunabhängige Technologiefestlegung. Dies bedeutet, dass definiert wird, auf welche Art und Weise der im Schritt zwei festgelegte Prozess ausgeführt wird. Um eine produktverträgliche Technologie wählen zu können, sind an dieser Stelle genaue Informationen über das Produkt erforderlich. Exemplarisch könnte hier eine Glasflasche einer bestimmten Größe mit einem Plastikschraubdeckel genannt werden. Dabei müssen zum Beispiel die Materialeigenschaften der verwendeten Produkte bekannt sein.

- Im vierten Engineering-Schritt werden technologieabhängige, allerdings noch realisierungsunabhängige Anforderungen spezifiziert. So kann hier zum Beispiel definiert werden, dass die vom Prozess verwendeten Glasflaschen mit höchstens 10 Nm durch das Aufsetzen der Deckel belastet werden dürfen.

Nach dem vierten Schritt ist die funktionale Ebene abgeschlossen. Die Anforderungen an den Prozess sind an dieser Stelle soweit spezifiziert, dass eine mechanische Realisierungsvariante in der Konstruktion erstellt werden kann.

3.3 Semantische Beschreibung der Struktur und des Verhaltens von Automatisierungssystemen

Im nachfolgenden Abschnitt soll mit Hilfe der im Kapitel 3.1 vorgestellten Konzepte und Beschreibungselemente ein Ausschnitt eines Automatisierungssystems schrittweise semantisch beschrieben werden. Dabei stehen die Beschreibungen der Abhängigkeiten zwischen den Komponenten der Automatisierungstechnik, deren mechanische Eigenschaften sowie die funktionale Verwendung der Komponenten im Vordergrund. Die folgenden Darstellungen entsprechen der Datenstruktur, die im Hintergrund von Engineering-Tools als Informationsmodell etabliert werden soll.

Grundlegendes Ziel ist die bidirektionale Verbindung zwischen Engineering und Runtime. Hierfür müssen relevante Informationen zur Verwendung bzw. zur Erstellung einer Steuerungslösung in der Runtime aus dem Engineering ausgeleitet werden. Darüber hinaus müssen alle weiteren Informationen in der Runtime abgelegt werden, die auch während des Anlagen-Life-Cycles eine Zuordnung der einzelnen Komponenten einer automatisierten Produktionsanlage zu deren Repräsentanten im Engineering ermöglichen. Dabei soll die Intension der Verwendung (Aufgabe) einer Komponente nicht verloren gehen. Dies erlaubt eine bidirektionale Kopplung zwischen realer Produktionsanlage und deren Projektierung im Engineering.

Ausgehend von der funktionalen Ebene werden die spezifizierten Anforderungen an die Konstruktion zur Umsetzung einer Realisierungsvariante übermittelt. Die kon-

struktive Umsetzung erfolgt jedoch in enger Interaktion mit der semantischen Aufgabenbeschreibung, da wesentliche Informationen und Abhängigkeiten nur der Konstruktion entnommen werden können.

Sollten die im funktionalen Engineering spezifizierten Anforderungen nicht durch eine konstruktive Lösung umsetzbar sein, müssen die Anforderungen derart angepasst werden, sodass eine konstruktive Realisierungsvariante erstellt werden kann. Die Interaktion zwischen Konstruktion und funktionalem Engineering kann auch zur Erstellung einer optimierten Realisierungsvariante genutzt werden.

Das Vorgehen in der Konstruktionsphase zur Erstellung einer Realisierungsvariante unterscheidet sich von den bisherigen Ansätzen durch einen erhöhten bidirektionalen automatisierbaren Informationsaustausch. Da keine Steuerungslösung für die Automatisierungstechnik einer Anlage im Sinne des Prozesses ohne Berücksichtigung mechanischer sowie konstruktiver Abhängigkeiten erstellt werden kann, spielt die Konstruktionsphase und die damit verbundene Interaktion mit dem funktionalen Engineering und der Inbetriebnahme eine wichtige Rolle.

Während der Konstruktionsphase muss das Rollen-Konzept (vgl. 3.1.2) weiter angewendet werden. Dies bedeutet, dass die Anforderungen an eine mechanische Komponente seitens des Konstrukteurs definiert werden können und somit nachhaltig zur Verfügung stehen. Des Weiteren besitzt der Konstrukteur Wissen über die Funktionsweise der von ihm entwickelten Realisierung. Ihm ist also bekannt, welche funktionale Aufgabe jede einzelne Komponente erfüllt, er kennt die Anforderungen des Prozesses und er hat eine Vorstellung, wie die Komponenten der von ihm entwickelten Konstruktion interagieren müssen, um den Prozess zu realisieren. Somit ist bereits der Konstrukteur in der Lage, schrittweise alle Sensor- bzw. Aktuatorkonfigurationen seiner Konstruktion im Sinne des Prozesses zu beschreiben. Dies erfolgt durch die Definition von Interaktionspunkten (IAP) auf allen Ebenen der Aggregation der konstruktiven Realisierungsvariante bis hin zur funktionalen Ebene. Die Interaktion zwischen mechanischer Konstruktion und dem Informationsmodell wird im folgenden Abschnitt näher erläutert.

Als erster Schritt werden die in der Konstruktion vorliegenden mechanischen Komponenten unter Berücksichtigung der vorhandenen kinematischen Abhängigkeiten in das Informationsmodell übernommen. Wie in Abbildung 32 zu sehen, wird der mechanische Repräsentant eines Pneumatikzylinders, auf dem zwei Sensoren und ein Aktuator räumlich lokalisiert sind (mechanische Anbauposition von Sensorik/Aktuatorik), in das Informationsmodell überführt.

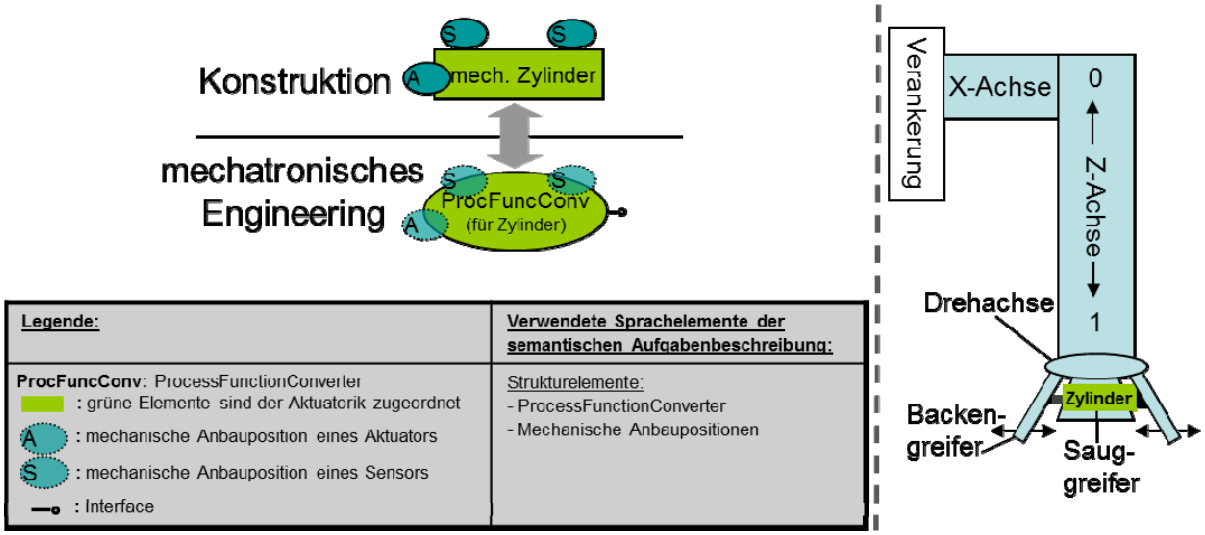
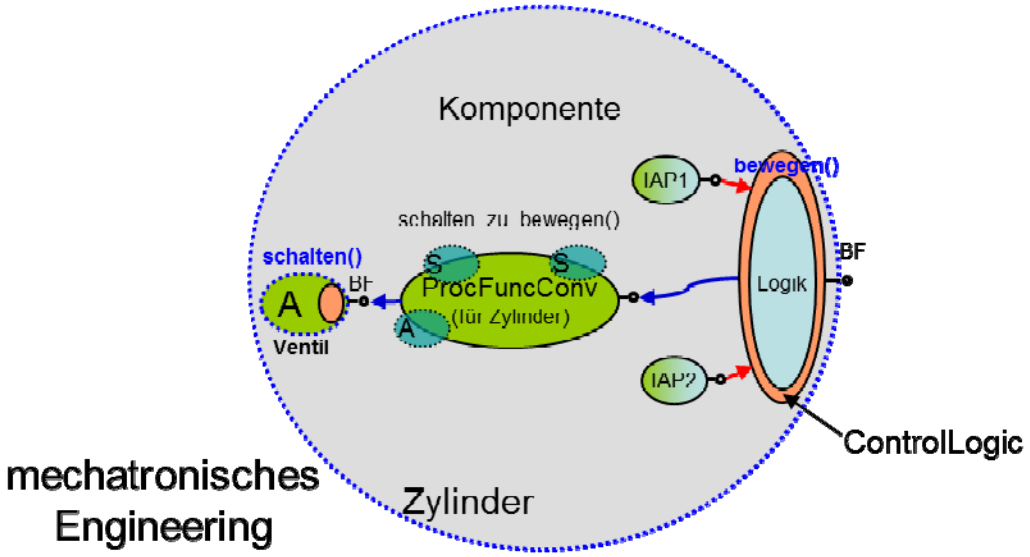


Abbildung 32: Erster Schritt zur Erzeugung eines mechatronischen Objekts

So werden Geometriedaten, die räumliche Positionierung, mechanische Abhängigkeiten sowie die konstruktive Zuordnung von zum Beispiel Sensoren in einer ProcessFunctionConverter-Struktur des Informationsmodells gespeichert. Diese enthält alle relevanten Informationen, um die rotatorische Bewegung eines Motors in eine translatorische Bewegung einer Linearachse umzuformen. Der ProcessFunctionConverter ist somit der automatisierungstechnische, funktionale Repräsentant der Mechanik und den damit verbundenen Abhängigkeiten.

Die Erfassung und Überführung von mechanischen Anbaupositionen ist erforderlich, da die mechanische Anordnung und die funktionale Verwendung von Sensorik und Aktuatorik oftmals abweichen. Aus diesem Grund werden in den nachfolgenden Abbildungen mechanische Anbaupositionen dargestellt, auf die erst in Abbildung 35 und Abbildung 36 referenziert wird. Darauf wird im weiteren Verlauf der Arbeit noch eingegangen.

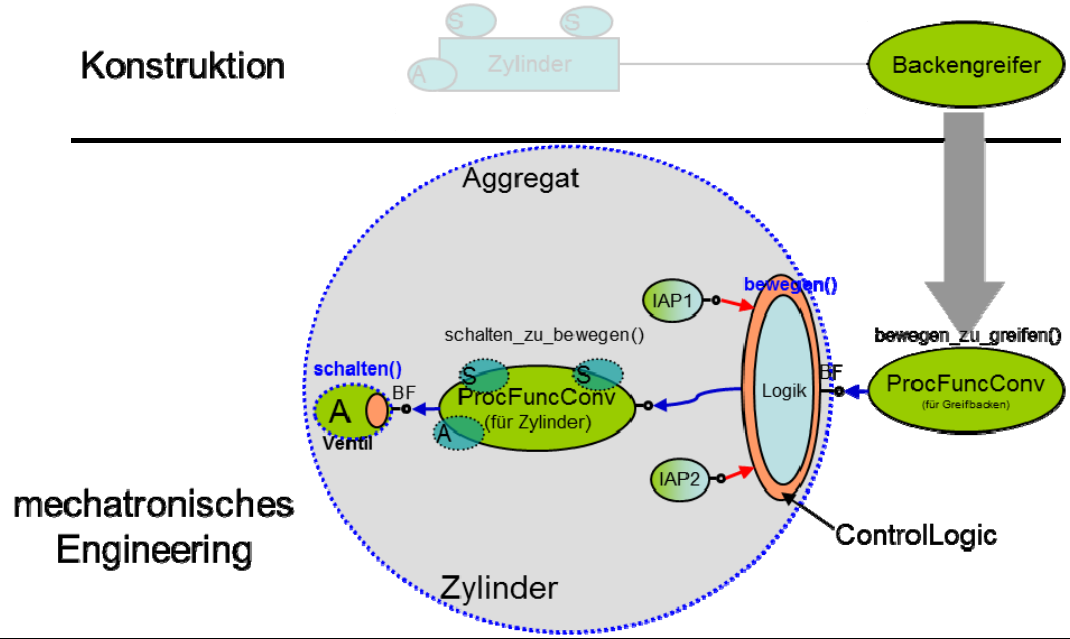
Ausgehend von den Informationen aus der Konstruktion werden in dem Informationsmodell aus der mechanischen Realisierung verschiedene Komponenten (siehe Abbildung 33) mit klarer Zuordnung zur Kinematik gebildet.



Legende:	Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:
BF: BaseFunction **IAP:** InterAktionsPunkt **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet : mechanische Anbauposition eines Aktuators : mechanische Anbauposition eines Sensors : Interface : Komponente	Strukturelemente: - ControlLogic - IAP - BaseFunction - ProcessFunctionConverter - Aktuator - Mechanische Anbaupositionen Basisfunktionalitäten / Komponente: - **schalten()** / Ventil - **bewegen()** / Zylinder

Abbildung 33: Bildung der Aggregatsstruktur

Als zentraler Schritt wird dabei eine ControlLogic generiert. Diese implementiert eine Logik zur Verwendung der Komponente und kann über funktionale Schnittstellen, den sogenannten BaseFunctions, mit anderen Komponenten interagieren bzw. mit diesen verschaltet werden. Durch die Definition von Interaktionspunkten (IAP) können bestimmte Zustände der Komponente im Sinne des Prozesses beschrieben werden. Die IAPs sind mit der ControlLogic verknüpft, welche die Funktionalität der BaseFunction (funktionale Schnittstelle) implementiert. Soweit die Überführung des Zylinders in das Informationsmodell abgeschlossen ist, können alle mechanisch abhängigen Komponenten in das mechatronische Informationsmodell ausgeleitet werden. Dies bedeutet, dass alle mechanischen Abhängigkeiten, wie zum Beispiel eine kinematische Verkettung mehrerer Aktuatoren, mit Hilfe der ProcessFunctionConverter-Struktur (in den Abbildungen als ProcFuncConv abgekürzt) beschrieben werden. Die funktionale Strukturierung im Sinne des Prozesses wird durch die Aggregation realisiert (siehe 3.1.5). Bei dem hier gewählten Beispiel eines Backengreifers (siehe Abbildung 34) ist aus der Konstruktion ersichtlich, dass die Greifbacken mechanisch direkt mit dem Zylinder gekoppelt sind.

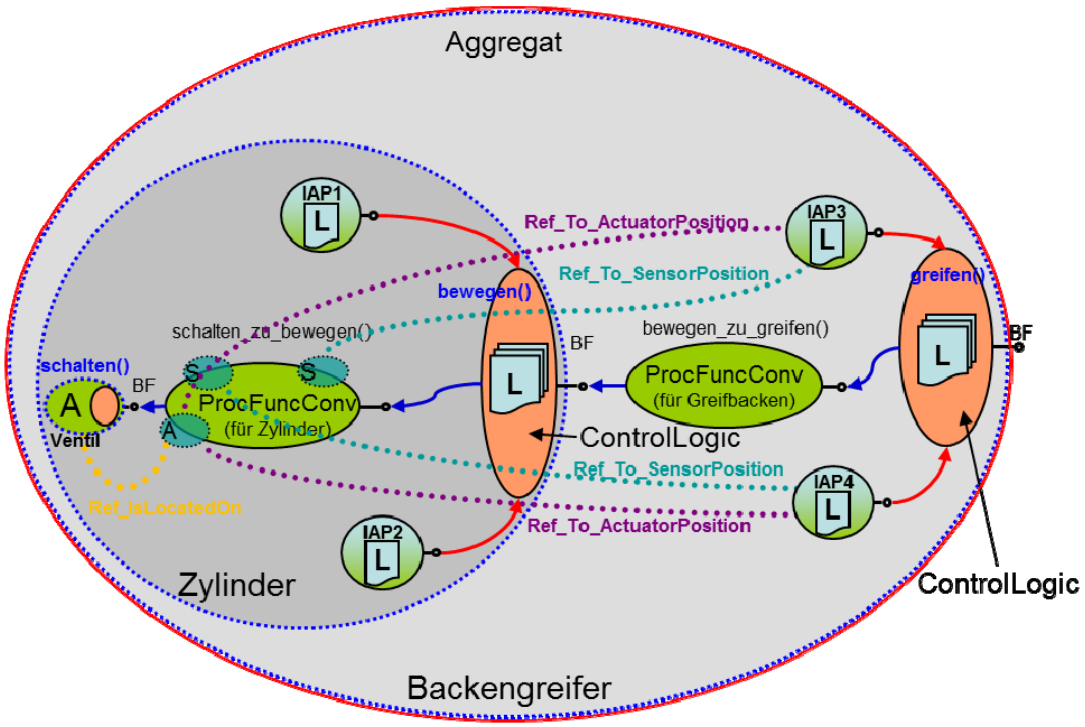


Legende:	Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:
BF: BaseFunction **IAP:** InterAktionsPunkt **L:** Logik **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Informationsfluss : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet : mechanische Anbauposition eines Aktuators : mechanische Anbauposition eines Sensors : Interface : Komponente	Strukturelemente: - ControlLogic - Logik - IAP - BaseFunction - ProcessFunctionConverter - Aktuator - Mechanische Anbaupositionen Basisfunktionalitäten / Komponente: - **schalten()** / Ventil - **bewegen()** / Zylinder

Abbildung 34: Erstellen der mechanischen Abhängigkeiten

In der Konstruktion wird der Zylinder zu einer komplexeren Funktionseinheit zusammengesetzt. Dies erfolgt im Informationsmodell analog mittels Aggregation. Es wird das neu erzeugte ProcFuncConv-Element der Greifbacken mit der Aggregatskomponente des Zylinders verknüpft. Dies bildet die mechanische Abhängigkeit im Informationsmodell ab. Die funktionalen Abhängigkeiten werden mittels Aggregation, der ControlLogic sowie den IAPs umgesetzt. Wie in Abbildung 35 ersichtlich, beinhaltet das Aggregat „Backengreifer“ die Komponente „Zylinder“. Der Zylinder wird zur mechanischen Realisierung der Komponente „Backengreifer unter Berücksichtigung der im ProcFuncConv beschriebenen Abhängigkeiten verwendet. Die Festlegung der Funktionalität des Backengreifers erfolgt in diesem Fall in der ControlLogic des Backengreifers durch zwei Interaktionspunkte, die den Zustand „Backengreifer offen“ bzw. „Backengreifer geschlossen“ definieren. Zur Zustandserkennung wird den beiden IAPs jeweils ein Lagesensor zugewiesen („*Ref_To_SensorPosition*“-Referenz zwischen „S“ und IAP1 bzw. IAP2). Beide Sensoren sind in der Konstruktion jedoch am Zylinder angebracht. Durch das Konzept der Referenzierung zwischen funktionaler Verwendung und mechanischer Realisierung ist eine Trennung der mechanischen

Realisierung und der eigentlichen Funktion einer Komponente im Sinne des Prozesses möglich.

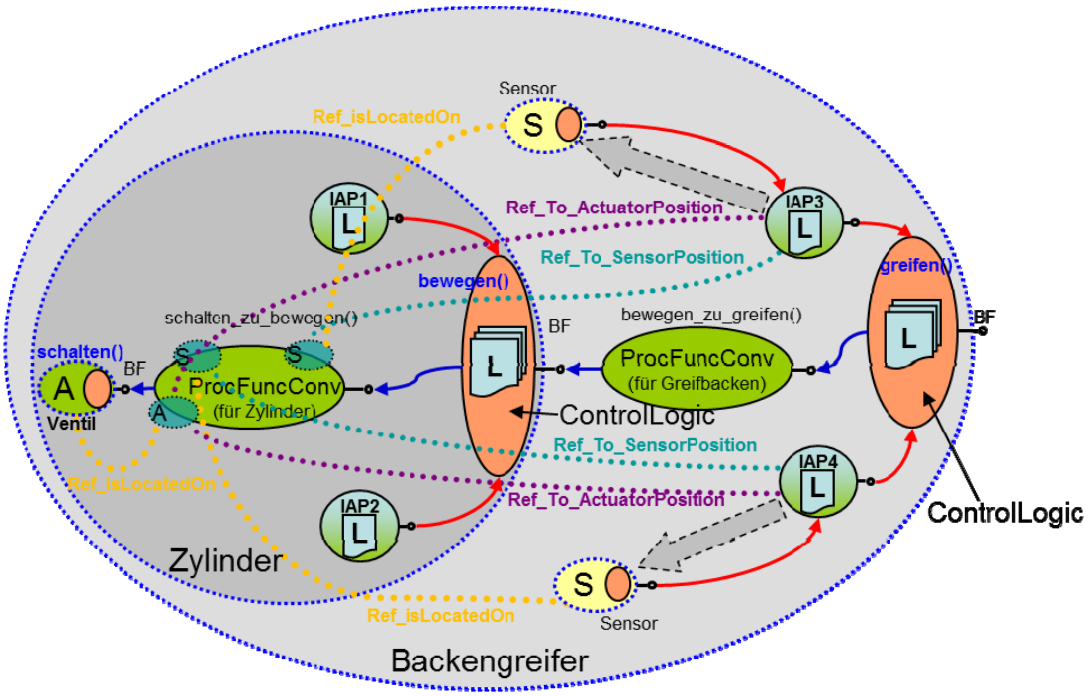


Legende:	**Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:**
BF: BaseFunction **IAP:** InterAktionsPunkt **L:** Logik **S:** Sensor **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Informationsfluss : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet A : mechanische Anbauposition eines Aktuators S : mechanische Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Aktuators : Referenz zwischen einer Komponente (Sensor/Aktuator) und deren mechanische Anbauposition : Interface : Komponente	Strukturelemente: - ControlLogic - Logik - IAP - BaseFunction - ProcessFunctionConverter - Sensor - Aktuator - Mechanische Anbaupositionen Referenzelemente: - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Sensorik (mech. Realisierung) - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Aktuatorik (mech. Realisierung) - Referenz zwischen einer Komponente und deren mechanische Anbauposition Basisfunktionalitäten: - **schalten()** / Ventil - **bewegen()** / Zylinder - **greifen()** / Backengreifer

Abbildung 35: Aggregation unter Berücksichtigung mechanischer Abhängigkeiten

Beide Sensoren sind zwar in der Konstruktion auf den Zylinder montiert. Funktional besteht jedoch eine Zuweisung zu den IAPs der ControlLogic des Backengreifers. Es ist eine Repräsentation der Sensoren mittels Rolle im Informationsmodell erforderlich. Die beiden Sensorrollen wurden in Abbildung 36 gelb eingezeichnet. Aus Sicht des Informationsmodells werden die beiden Sensoren funktional vom Aggregat des Backengreifers mittels dessen IAPs direkt verwendet. Aus diesem Grund befinden

sich die Rollen der Sensoren auf einer anderen Aggregationsebene als deren Repräsentanten in der Konstruktion.



Legende:	**Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:**
BF: BaseFunction **IAP:** InterAktionsPunkt **L:** Logik **S:** Sensor **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Informationsfluss : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet : gelbe Elemente sind der Sensorik zugeordnet A : mechanische Anbauposition eines Aktuators S : mechanische Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Aktuators : Referenz zwischen einer Komponente (Sensor/Aktuator) und deren mechanische Anbauposition : Interface : Komponente	Strukturelemente: - ControlLogic - Logik - IAP - BaseFunction - ProcessFunctionConverter - Sensor - Aktuator - Mechanische Anbaupositionen Referenzelemente: - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Sensorik (mech. Realisierung) - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Aktuatorik (mech. Realisierung) - Referenz zwischen einer Komponente und deren mechanische Anbauposition Basisfunktionalitäten: - **schalten()** / Ventil - **bewegen()** / Zylinder - **greifen()** / Backengreifer

Abbildung 36: Erzeugung der Sensor-Rollen im Aggregat des Backengreifers

Um den Bezug zwischen Engineering und den Rollen im Informationsmodell nicht zu verlieren, enthält jede Komponente in der Konstruktion eine „Ref_To_SensorPosition“ oder „Ref_To_AktuatorPosition“-Referenz, über die eine Zuordnung zu einem IAP erfolgt. Daraus lässt sich wiederum implizit eine Rolle ableiten. Diese enthält eine Referenz „Ref_isLocatedOn“ zum jeweiligen Repräsentanten in der Konstruktion. Somit lässt sich jederzeit ein bidirektionaler Bezug zwischen den Komponenten der Konstruktion und deren Funktionalität darstellen.

In Abbildung 37 ist exemplarisch dargestellt, wie mittels Aggregation die Teilkomponenten „Greifereinheit“ und „Drehachse“ zu einer „rotatorischen Greifeinheit“ zusammengefügt werden. Die Koordination zwischen den einzelnen Komponenten, zum Beispiel dem „Greifer“ und der „Drehachse“ wird in der ControlLogic der „rotatorischen Greifeinheit“ definiert. Da eine koordinierte Bewegungssequenz zwischen Greifeinheit und Drehachse erforderlich ist, um den Prozess zu realisieren, ist die „rotatorische Greifeinheit“ funktional sowohl mit der Greifeinheit als auch mit der Drehachse verbunden. Durch die Aggregation werden funktional somit immer „höherwertigere“ Einheiten gebildet.

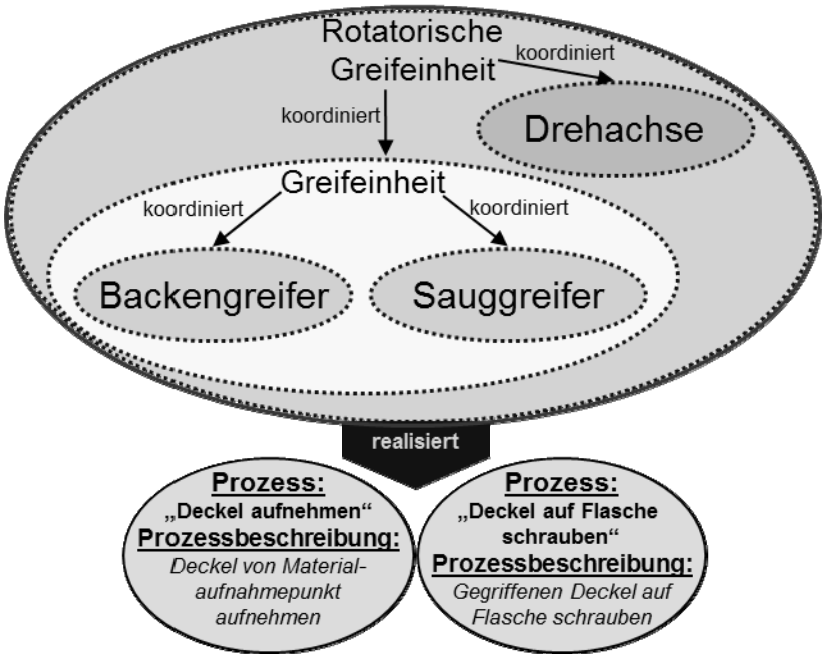


Abbildung 37: Darstellung des Aggregationsprinzips am Beispiel mehrerer Teilkomponenten

Um den Bezug zur mechanischen Konstruktion nicht zu verlieren, sind zusätzlich zu den funktionalen Zuordnungen noch Referenzen erforderlich, die eine mechanische Abhängigkeit kennzeichnen. Diese können entweder direkt durch die Ordnungsstruktur von Baugruppen und Bauteilen oder aber auch durch eine individuelle Referenzierung zum Beispiel im Rahmen der Kinematisierung der Mechanik erstellt werden. Dies stellt das Bindeglied zwischen funktionaler Verwendung der Komponenten und den verknüpften mechanischen Abhängigkeiten der Konstruktion dar.

Ein grundsätzliches Ziel ist die enge Verzahnung zwischen der Projektierung im Engineering und der realen Anlage. Die semantische Beschreibung einer Automatisierungsanlage (siehe Abbildung 38, links) mit definierten Beschreibungsmitteln einer festgelegten Syntax folgend stellt dabei die Grundlage dar, um die Projektierung bidirektional in Richtung Runtime abbilden zu können.

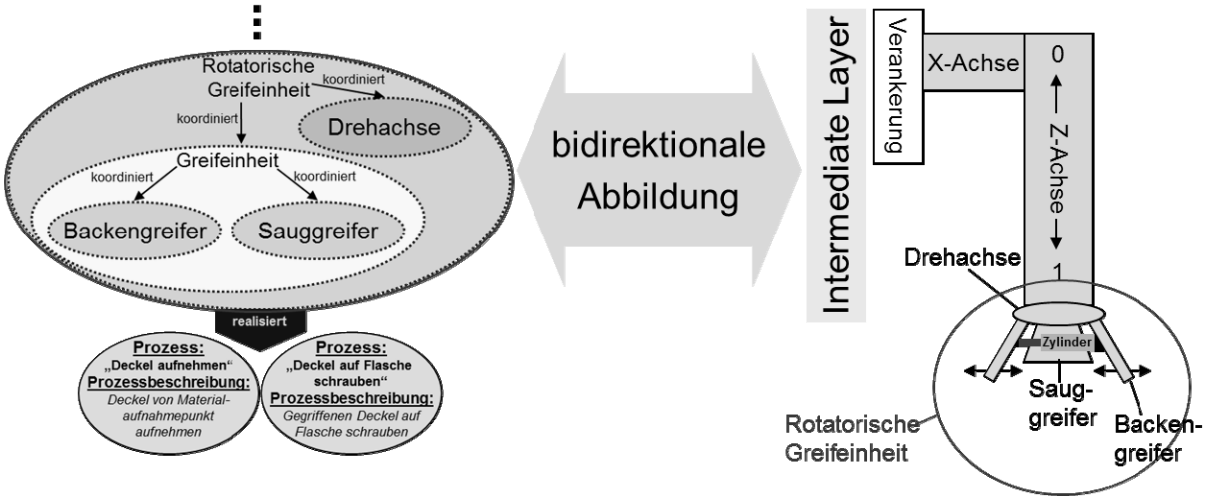


Abbildung 38: Bidirektionale Kopplung der semantischen Beschreibung eines Automatisierungssystems mit der realen Anlage

Die bidirektionale Abbildung zur Runtime erfolgt über eine Metaebene (Intermediate Layer) (siehe Abbildung 38, rechts). Diese Metaebene realisiert das virtuelle Abbild der realen Anlage oder Maschine auf deren Steuerungsebene. Dies erlaubt eine Abbildung des Konzepts sowohl ins Engineering als auch in die Ablaufumgebung einer realen Anlage. Die technische Realisierung wird später in Kapitel 4.3 erläutert.

3.4 Konzeptioneller Aufbau der ControlLogic

Die ControlLogic realisiert die Verschaltungs- und Ablauflogik zwischen Komponenten der Automatisierungstechnik. Sie definiert die Verwendung einer Komponente und implementiert deren Funktionalität. Die ControlLogic verwendet semantisch beschriebene Interaktionspunkte (IAP) zur

- Beschreibung von Konfigurationen im Sinne des Prozesses,
- Festlegung von speziellen Konfigurationsbereichen bzw. Endlagestellungen,
- Definition von Einschaltverriegelungen.

Diese Einteilung der Interaktionspunkte ermöglicht in der ControlLogic eine teilweise automatisierte Verarbeitung, wie zum Beispiel die automatische Überprüfung der Einschaltverriegelungen, wodurch die Komplexität der Steuerungslogik reduziert werden kann.

Mittels Interaktionspunkte zur Festlegung spezieller Bewegungsbereiche können Konfigurationen einer Komponente definiert werden, die bei einer zulässigen Verwendung nicht eingenommen werden dürfen. Dies könnten zum Beispiel Endlagensensoren einer Linearachse sein. Für die ordnungsgemäße Funktionalität des Mechanikverbunds sind mechanische Abhängigkeiten sowie räumliche Gegebenheiten zu berücksichtigen. Dies beschränkt das Funktionsspektrum von Automatisierungs-

komponenten auf den jeweiligen Lösungsraum bezüglich funktionaler Verwendung im Sinne des Prozesses.

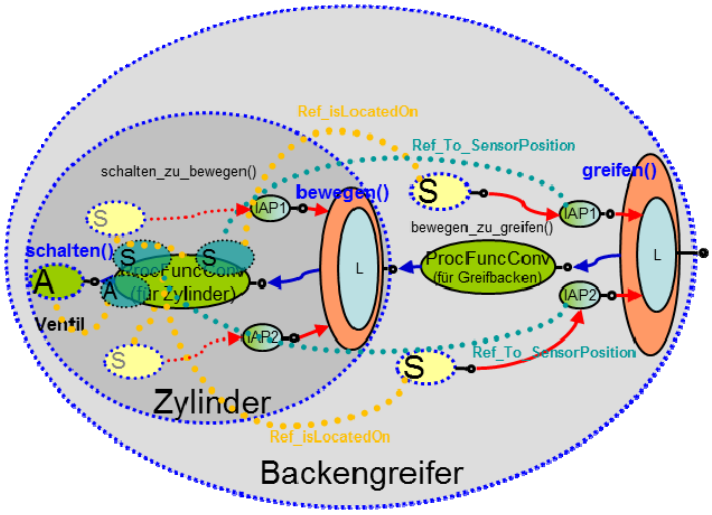
Dies kann durch IAPs zur Festlegung von speziellen Konfigurationsbereichen umgesetzt werden, die einen zulässigen Wertebereich im Sinne des Prozesses beschreiben. Soweit dieser Wertebereich nicht verlassen wird, ist die von den IAPs definierte Bedingung erfüllt.

Für den Betrieb einer Komponente müssen bestimmte Einschaltbedingungen initial oder auch permanent erfüllt sein. Diese können ebenfalls mittels IAPs definiert werden. Ein Einschaltverriegelungs-IAP kann zum Beispiel verwendet werden, um Bedingungen wie „Druckluft ist vorhanden“, „Betriebsmittel vorhanden“, „Schutzgitter geschlossen“ oder „Prozessfreigabe erteilt“ semantisch beschreiben und im weiteren Verlauf des Anlagen-Engineerings mit Komponenten der Automatisierungstechnik verknüpfen zu können.

Neben der Einbindung typisierter Interaktionspunkte sowie deren Verarbeitung implementiert die ControlLogic die Interaktion mit unterlagerten Komponenten. Dies erfolgt über BaseFunction-Aufrufe der unterlagerten Komponenten.

3.5 Bidirektionale Interaktion zwischen Konstruktion und Inbetriebnahme

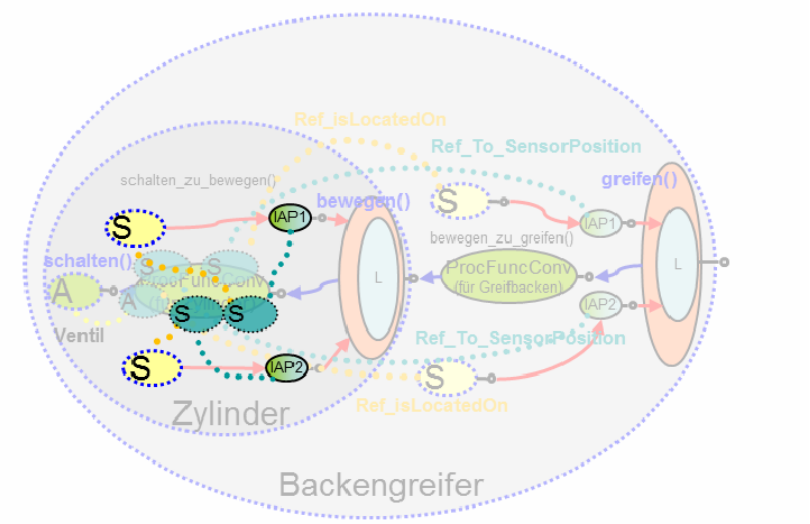
Ein wesentlicher Aspekt des Informationsmodells ist die Interaktionsfähigkeit der verschiedenen Engineering-Phasen. Dies resultiert insbesondere aus der Anforderung an die flexible Anpassbarkeit des Engineerings bezüglich Änderungen der Projektierung. Einen besonderen Fall stellt hier zum Beispiel die Interaktion zwischen Inbetriebnahme und Konstruktion dar. Es kommt bei Projektdurchführungen immer wieder vor, dass im Feld von der Planung auf Grund nicht berücksichtigter Faktoren abgewichen werden muss. Um diese Inkonsistenzen handhabbar zu machen, sind Mechanismen erforderlich, die Änderungen der Anlage während der Inbetriebnahmephase als Anforderung an die Konstruktion rückmelden zu können. Auch hierfür eignet sich das beidseitige Rollenverständnis der Inbetriebnahme und Konstruktion. So können seitens der Inbetriebnahme zwei Rollen, exemplarisch in Abbildung 39 dargestellt, im Informationsmodell neu eingefügt werden. Diese definieren bestimmte funktionale Anforderungen und bereits funktional den IAPs, ProcessFunctionConvertern und ControlLogics zugeordnet sind. In dem in Abbildung 39 dargestellten Beispiel werden zwei neue Rollen für Sensoren im Aggregat des Zylinders erzeugt. Diese sind funktional dem IAP1 und IAP2 der ControlLogic des Zylinders zugeordnet. Die Sensoren detektieren somit das Ein- und Ausfahren des Zylinders.



Legende:	**Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:**
BF: BaseFunction **IAP:** InterAktionsPunkt **L:** Logik **S:** Sensor **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Informationsfluss : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet : gelbe Elemente sind der Sensorik zugeordnet : mechanische Anbauposition eines Aktuators : mechanische Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Aktuators : Referenz zwischen einer Komponente (Sensor/Aktuator) und deren mechanische Anbauposition : Interface : Komponente	Strukturelemente: - ControlLogic - Logik - IAP - BaseFunction - ProcessFunctionConverter - Sensor - Aktuator - Mechanische Anbaupositionen Referenzelemente: - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Sensorik (mech. Realisierung) - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Aktuatorik (mech. Realisierung) - Referenz zwischen einer Komponente und deren mechanische Anbauposition Basisfunktionalitäten: - **schalten()** / Ventil - **bewegen()** / Zylinder - **greifen()** / Backengreifer

Abbildung 39: Einfügen von zusätzlichen Sensoren während der Inbetriebnahme

Die Realisierung der beiden neu eingefügten Rollen erfolgt in der Konstruktion. Diese prüft, ob und auf welche Art und Weise die konstruktive Umsetzung der beiden neu einzufügenden Sensoren erfolgen kann. In der mechanischen Konstruktion wird somit die mechanische Befestigung der gewünschten Sensoren hinterlegt. Sollte keine Lösung in der Konstruktion gefunden werden, kann an die Inbetriebnahme keine konsistente Realisierungsvariante für die beiden neu eingefügten Sensorrollen zurückgegeben werden. Eine solche Inkonsistenz wäre in der Projektierung klar ersichtlich, da jede reale Komponente eine Referenz auf ihre mechanische Anbauposition hat, die in dem geschilderten Fehlerfall nicht existieren würde.



Legende:	Verwendete Sprachelemente der semantischen Aufgabenbeschreibung:
BF: BaseFunction **IAP:** InterAktionsPunkt **L:** Logik **S:** Sensor **A:** Aktuator **ProcFuncConv:** ProcessFunctionConverter : Informationsfluss : Aufruf der Aktuatorik (Aktion) : grüne Elemente sind der Aktuatorik zugeordnet : gelbe Elemente sind der Sensorik zugeordnet A : mechanische Anbauposition eines Aktuators S : mechanische Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Sensors : Referenz zwischen IAP und mech. Anbauposition eines Aktuators : Referenz zwischen einer Komponente (Sensor/Aktor) und deren mechanische Anbauposition : Interface : Komponente	Strukturelemente: - ControlLogic - Logik - IAP - BaseFunction - ProcessFunctionConverter - Sensor - Aktuator - Mechanische Anbaupositionen Referenzelemente: - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Sensorik (mech. Realisierung) - Referenz zwischen IAP (Verwendung semantisch beschriebener Information) und verwendeter Aktuatorik (mech. Realisierung) - Referenz zwischen einer Komponente und deren mechanische Anbauposition Basisfunktionalitäten: - **schalten()** / Ventil - **bewegen()** / Zylinder - **greifen()** / Backengreifer

Abbildung 40: Erfüllung der Anforderung zusätzlicher Sensoren mittels Realisierung in der Konstruktion

Ist eine Realisierung der neuen Anforderungen aus der Inbetriebnahme möglich, so wird an die Inbetriebnahmephase eine konsistente Realisierung der beiden neu eingefügten Sensoren zurückgeliefert (siehe Abbildung 40). Dies bedeutet, dass die „Ref_isLocatedOn“-Referenzen der neu eingefügten Sensorrollen reale Komponenten der mechanischen Konstruktion referenzieren und diese funktional mittels “Ref_To_SensorPosition“-Referenzen den Interaktionspunkten der ControlLogic des Zylinders zugeordnet sind.

3.6 Darstellung des Konzepts am Beispiel einer Handhabungseinheit

Das in den vorhergehenden Kapiteln vorgestellte Konzept soll mit Hilfe eines praxisnahen Beispiels verdeutlicht werden. Hierfür wird eine Handhabungseinheit betrach-

tet, die mit ihrem Endeffektor bestehend aus einem Saug- und einem Backengreifer kleine Plastikdeckel aufnehmen und auf Glasflaschen schrauben kann. Der mechanische Aufbau dieser Einheit, die Konfigurationsmöglichkeiten der einzelnen Komponenten sowie die Prozesspunkte 1 und 2 sind in Abbildung 41 dargestellt. „Prozesspunkt 1“ kennzeichnet eine Materialaufnahmeposition eines Plastikdeckels. Dieser muss über eine Materialzuführung bereitgestellt werden. „Prozesspunkt 2“ definiert eine Materialabgabeposition zum Verschrauben des aufgenommenen Deckels auf einer Glasflasche.

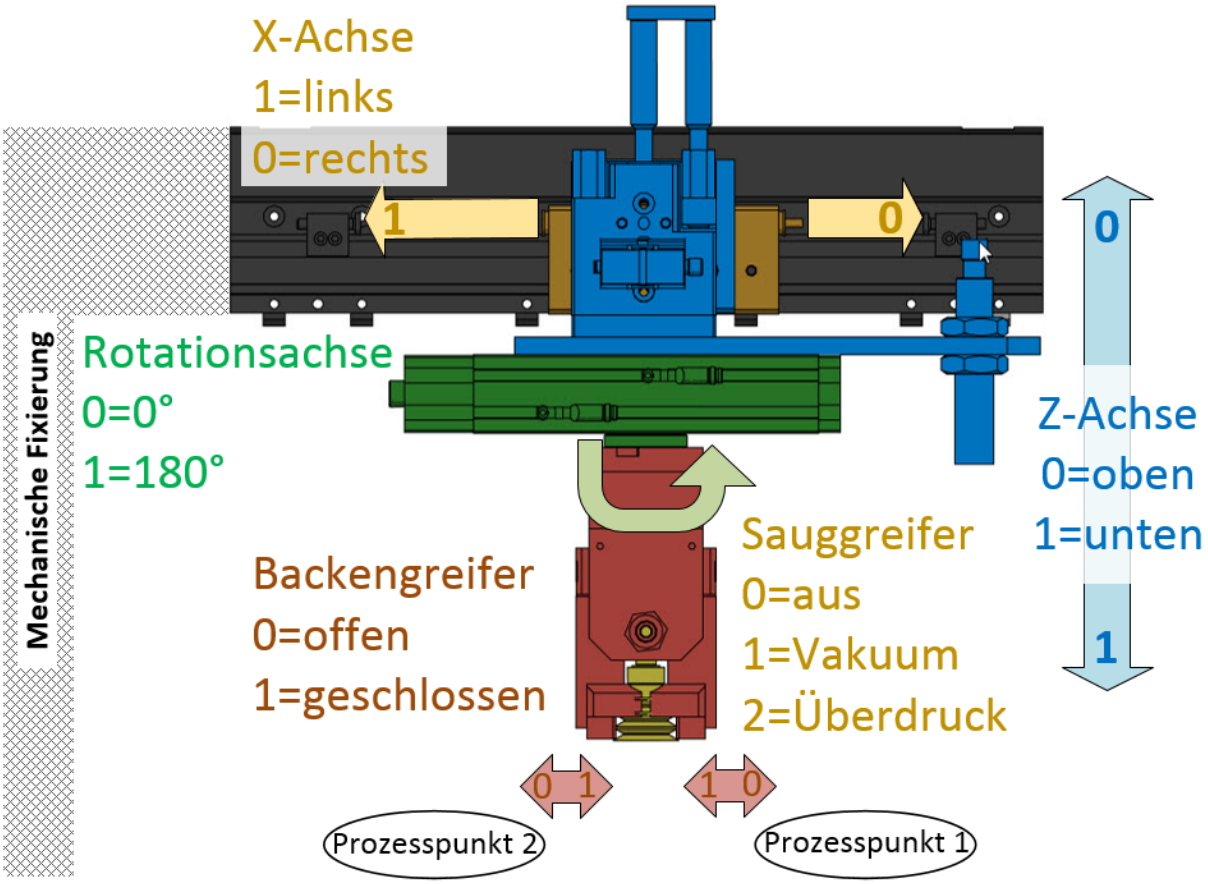


Abbildung 41: Darstellung einer Handhabungseinheit mit eingezeichneten Konfigurationsmöglichkeiten

Der zyklische Prozessablauf der betrachteten Greifeinheit kann mit Hilfe der Zustandsdefinitionen der einzelnen Komponenten in Vektorschreibweise dargestellt werden. Es wird jeder Dimension der Zustandsvektoren eine Komponente zugewiesen. Dadurch enthält ein Vektor die Soll-Konfigurationen aller beteiligten Komponenten und definiert somit einen Zustand einer mechatronischen Einheit. Der Gesamtprozess der Handhabungseinheit setzt sich aus den zwei Teilprozessen „Deckel aufnehmen“ und „Deckel auf Flasche schrauben“ zusammen, die in Abbildung 42 als Sequenz von Zustandsänderungen der einzelnen Komponenten in Vektorschreibweise dargestellt sind. Diese Vorgehensweise zur ganzheitlichen Beschreibung des Ablaufverhaltens eines mechatronischen Systems sowie eine entsprechende pro-

grammiertechnische Umsetzung als Steuerungsprogramm einer SPS stellt die heute übliche Form der SPS-Programmierung dar.

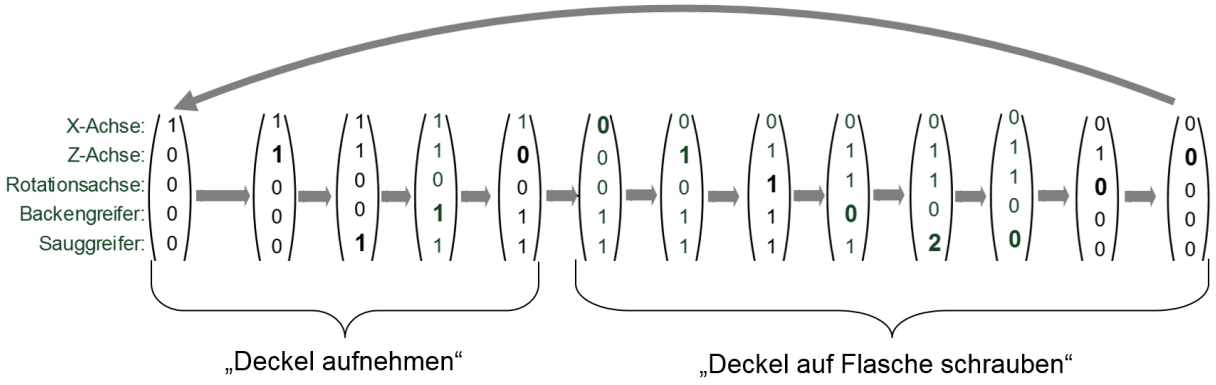


Abbildung 42: Darstellung des Prozessablaufs der Handhabungseinheit in Vektorschreibweise

Das in Kapitel 3.1 beschriebene Konzept geht im Vergleich hierzu mit einem modularen, hierarchischen Ansatz bezüglich der Anforderungsdefinition mit Hilfe von „Rollen“ (vgl. Kapitel 3.1.2) vor. Im Entwicklungsprozess für die Umsetzung eines Fertigungsprozesses haben dessen Anforderungen zentrale Priorität. Diese werden in einer übergeordneten „Rolle“ verankert und können durch unterlagerte Rollen beliebig modular aufgeteilt werden. Das Ziel bei diesem Vorgehen ist die modulare Aufschlüsselung der Anforderungen einer Rolle, bis diese durch eine Automatisierungskomponente erfüllt werden können.

Dieses Vorgehen erinnert an den Modellraum des Konstruierens nach Rude (siehe Kapitel 2.1.3), in dem ebenfalls durch Variation bzw. durch Zerlegung einer Problemstellung in Teilaspekte nach Lösungsmöglichkeiten für bestehende Anforderungen gesucht wird.

Bei dem hier betrachteten Prozess zur Montage eines Plastikdeckels auf einer Glasflasche lassen sich folgende Dinge als Anforderung ableiten:

- Die Deckelaufnahmeposition liegt vertikal neben der Deckelmontageposition. Dadurch ist eine vertikale Bewegung der bisher nicht weiter definierten Handhabungseinheit erforderlich.
- Der Plastikdeckel wird auf die Flasche gesetzt. Unter der Annahme, dass die Flasche aufrecht stehend zur Verfügung gestellt wird, muss der Plastikdeckel durch eine vertikale Bewegung auf die Flasche aufgesetzt werden.
- Die Verschraubung des Deckels auf der Flasche erfordert eine Drehbewegung.
- Der Deckel muss durch einen geeigneten Greifprozess aufgenommen werden.

Aus diesen Anforderungen, insbesondere an die mechanischen Fähigkeiten, kann mindestens eine Lösungsmöglichkeit abgeleitet werden. Die mechanische Strukturierung in Baugruppen sollte den kinematischen Abhängigkeiten folgen, was zum Beispiel bei der Kinematisierung für Simulationsstudien Vorteile mit sich bringt. Des Wei-

teren führt dies zu einer Aggregation von mechatronischen Komponenten zu funktional höherwertigen mechatronischen Einheiten. Dies ist beispielsweise in Abbildung 43 schrittweise dargestellt. Die Greifeinheit (links) wird mit einer Rotationsachse zu einer rotatorischen Greifeinheit aggregiert. Diese wiederum wird mit einer translatorischen Vertikalachse (Z-Achse) zu einer rotatorischen Z-Achs-Greifeinheit aggregiert. Der letzte Schritt zur Aggregation der translatorischen Horizontalachse (X-Achse) erfolgt analog, wird hier jedoch nicht dargestellt.

Nach dieser Strukturierung mittels Aggregation sind die Definition der Funktionsschnittstellen sowie die Festlegung der Zustände (IAP, siehe Kapitel 3.1.8) im Sinne des Prozesses eines jeden Aggregats vorzunehmen. Die Gesamtfunktionalität des Steuerungsprogramms ergibt sich aus der funktionalen Verknüpfung der einzelnen Aggregate.

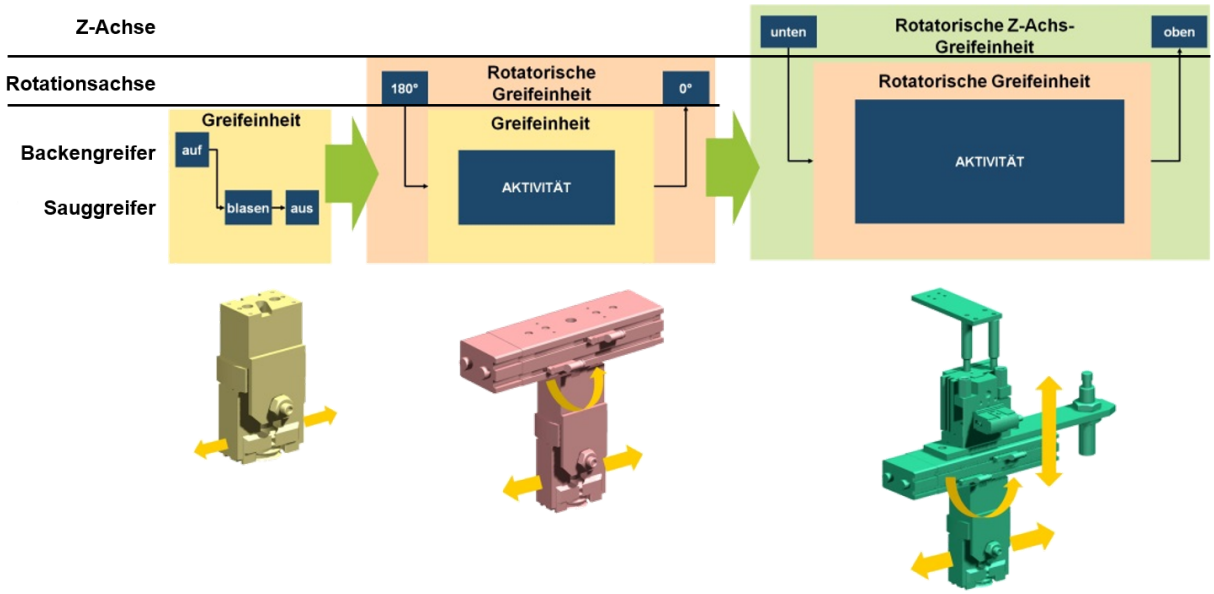


Abbildung 43: Darstellung einer komponentenbasierten, funktionalen Aggregation

Dies bietet den Vorteil, dass in einem Aggregat lediglich die relevanten Parameter sowie die funktionale Aufrufreihenfolge unterlagerter Aggregate hinterlegt sind, was sich vor allem beim funktionalen Komponententausch als vorteilhaft erweist, da hier die Anpassungen auf einer Hierarchieebene durchzuführen sind. Im Gegensatz zur klassischen SPS-Programmierung kann bei einem funktionalen Komponententausch nicht das gesamte Steuerungsprogramm von Änderungen betroffen sein, sondern lediglich das Aggregat, das die getauschte Komponente funktional integriert.

3.7 Klassifizierung von Informationen zur semantischen Beschreibung von Automatisierungssystemen

Zur Darstellung des semantischen Beschreibungsumfangs für automatisierte Produktionssysteme ist eine Strukturierung des Lösungsraums hilfreich. Dies erlaubt eine

sukzessive bedarfsorientierte Erweiterung und soll einen besseren Überblick gewährleisten. Um bereits eine objektorientierte Datenstrukturierung unter Verwendung von Vererbungsmechanismen vornehmen zu können, wurde im Rahmen dieser Arbeit ein UML-Klassenmodell entwickelt, das die wesentlichen Zusammenhänge eines generischen Betrachtungsraums beschreibt und flexibel erweiterbar ist. Das Klassenmodell beinhaltete verschiedene Klassen zur Repräsentation und Beschreibung von Komponenten und Rollen sowie Medien, Grundfunktionen und ein Werte- bzw. Einheitensystem und definiert deren Beziehungen und Abhängigkeiten untereinander. Es ist ein abstraktes Modell, das den semantischen Beschreibungsumfang von Automatisierungssystemen aufspannt und als Metamodell für eine Abbildung auf eine konkrete Implementierungsarchitektur (zum Beispiel OPC UA, siehe Kapitel 2.1.4) dienen kann. Wesentliche Merkmale sind die Erweiterbarkeit und die Anpassung des Informationsmodells an domänenspezifische bzw. kundenspezifische Ansprüche. Die dem Modell zu Grunde liegenden Konzepte sind in der nachfolgenden dargestellt. Diese Art der Strukturierung findet bereits bei OPC UA [37] durch verschiedene erweiterbare *Namespaces* Anwendung, was als grundlegender Strukturierungsansatz übernommen wurde, da der Beschreibungsumfang durch dieses Vorgehen abhängig von der Intension der Verwendung modular erweitert werden kann.

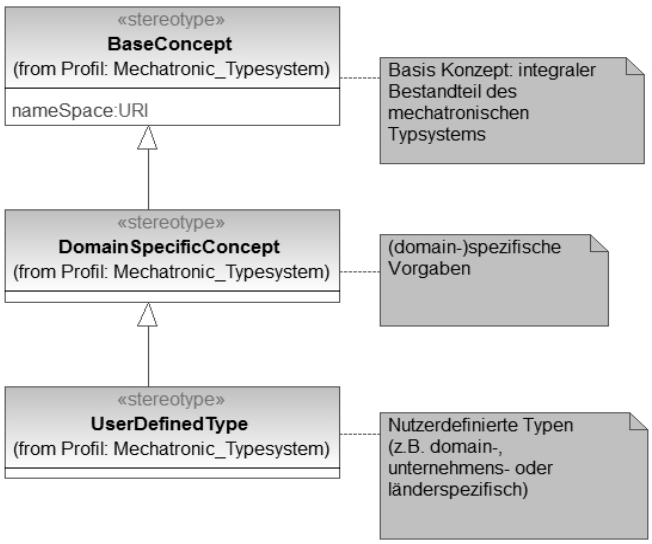


Abbildung 44: Grundlegendes Strukturierungskonzept für das Informationsmodell

Der Stereotyp BaseConcept kennzeichnet den integralen Bestandteil des Informationsmodells. Er definiert Typ- bzw. Klassendeklarationen, welche eine informationsmodellkonforme Komponente mindestens implementieren muss. Ein informationsmodellkonformer OPC UA Client beispielsweise kann diese Deklarationen als festen Bestandteil voraussetzen und somit explizit dagegen programmiert werden.

Das *DomainSpecificConcept* ist eine Erweiterung des *BaseConcept*. In diesem werden domänenspezifische Informationen semantisch abgelegt bzw. definiert. Dies sind beispielsweise Informationen aus den Bereichen *Condition Monitoring*, *Energy Management*, *Diagnostic*, *Safety*, *Security*, etc. Hier werden auch branchendomänen-

und länderspezifische Informationen definiert. Branchendomänen wären beispielsweise die Automobil- oder Chemie-Industrie. Beispiele für länderspezifische Informationen sind die erlaubten Funkfrequenzen bzw. Feldstärken, Verschlüsselungsverfahren etc.

Mit dem UserDefinedType-Konzept kann sowohl das BaseConcept als auch das DomainSpecificConcept erweitert werden. Hier können – basierend auf einem DomainSpecificConcept – weitere *DomainSpecificConcepts* dieses erweitern. So können hier zum Beispiel produkt- oder projektspezifische Informationen abgelegt bzw. definiert werden. Die Verantwortung, dass solche neuen oder erweiterten Elemente des Informationsmodells allen Beteiligten bekannt und damit interpretier- bzw. verarbeitbar sind, liegt beim Erweiterer des Informationsmodells.

3.7.1 Klassifizierung eines rollen- und komponentenbasierten Typ- und Instanzkonzepts

Im nachfolgenden Abschnitt wird ein Konzept zur Informationsstrukturierung von Automatisierungskomponenten beschrieben. Die nachfolgenden Klassendiagramme definieren informations- und datentechnische Strukturen, die genügend flexibel sind, um Automatisierungskomponenten aus informationstechnischer Sicht zu beschreiben. Dabei müssen die bereits vorgestellten Konzepte wie zum Beispiel eine komponentenbasierte Aggregation (vgl. Kapitel 3.1.5), Basisfunktionen oder aber auch Anforderungen bzw. zur Verfügung stehende Rechenleistung dargestellt werden können. Darüber hinaus soll das hier vorgestellte Informationsmodell vom Grundkonzept so strukturiert sein, dass es die Engineering-Phasen vom Requirements Engineering bis hin zur Inbetriebnahme begleiten kann. Deshalb wurde im Rahmen dieser Arbeit ein rollen- und komponentenbasiertes Typ-/Instanzkonzept entwickelt, das über diese erforderliche Dynamik verfügt. In Abbildung 45 sind die Ableitungshierarchien des Klassendiagramms mit Fokus auf die Spezifikation von Rollen und Komponenten dargestellt.

Die Klasse *BaseType* ist eine Basisklasse, die alle wesentlichen Informationen, die für eine Automatisierungskomponente eventuell definiert werden müssen, zusammenfasst. Sie beinhaltet beispielsweise ein digitales Typenschild (*pNumberPlate*), die benötigte Rechenleistung (*pRequiredComputingPower*) sowie eine oder mehrere Basisfunktionen (*BaseFunctions)* der Automatisierungskomponente. Von der Klasse *BaseType* leitet sich die Klasse *InstanceElementType* ab, die eine Beschreibung einer Instanz einer Rolle oder einer Komponente enthält. Von der Klasse *InstanceElementType* (siehe Abbildung 45) leitet sich sowohl die Klasse RoleType als auch die Klasse ComponentType ab. Die Klasse RoleType deklariert eine Rolle im Sinne eines Platzhalters im Engineering, der bestimmte Anforderungen enthält. Eine Rolle kann beliebig viele Rollen aggregieren, wodurch eine hierarchische Modularisierung erreicht werden kann.

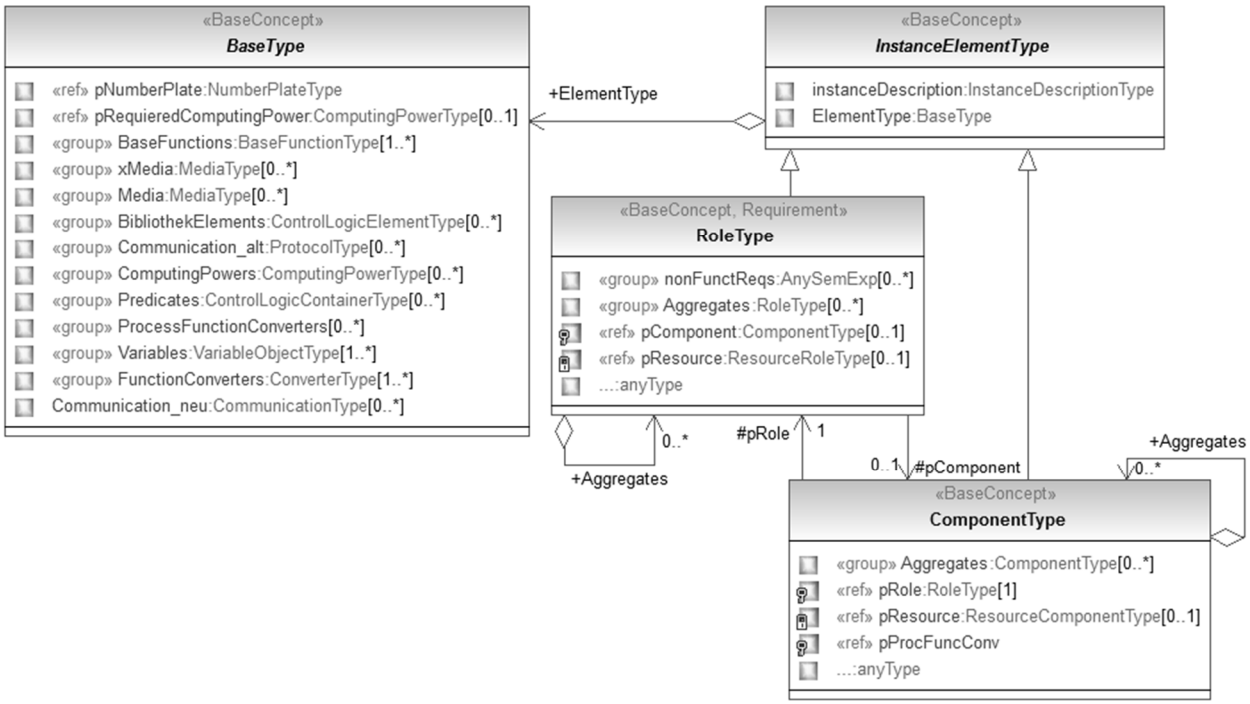


Abbildung 45: Darstellung des Rollen-/Komponentenkonzepts als UML-Klassendiagramm

Ziel dabei ist es, die Anforderungen der Rollen derart zu strukturieren und zu modularisieren, sodass für eine Rolle eine reale Komponente gefunden werden kann, die den Anforderungen der Rolle entspricht. Dieses Vorgehen ist in einer abstrahierten Form an die in Kapitel 2.1.3 beschriebenen Konzepte zur Entwicklung von Lösungsansätzen [19; 21; 23] angelehnt.

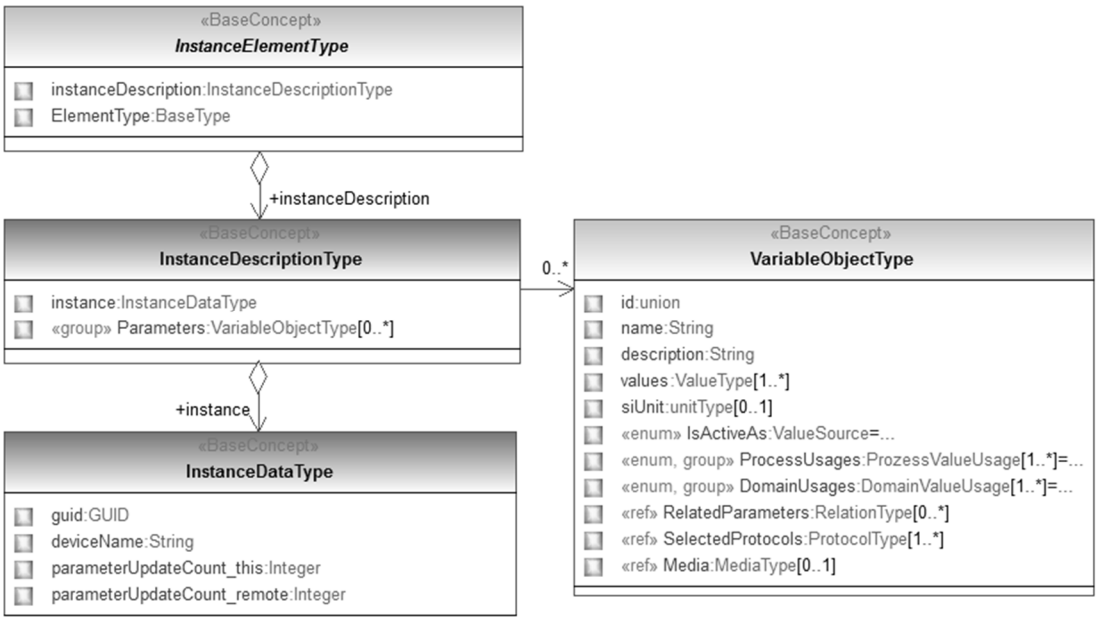


Abbildung 46: Darstellung der Klasse InstanceDescriptionType zur Beschreibung der Instanzen von Rollen oder Komponenten

Die Beschreibung einer Instanz eines *RoleType* oder *ComponentType* wird durch die Klasse *InstanceDescription* (siehe Abbildung 46) realisiert. Durch Verwendung einer beliebigen Anzahl (Kardinalität [0…*]) an Parametern vom Typ *VariableObjectType* bliebt der Umfang der Beschreibung variabel.

3.7.2 Klassifizierung von Variablen einer Automatisierungskomponente im Kontext ihrer Verwendung

Der Typ „VariableObjectType“ hat verschiedene Eigenschaften, die es ermöglichen, sowohl Relationen zu anderen Variablen als auch den für die Automatisierungsaufgabe zugeteilten Anwendungszweck zu hinterlegen. Dies ist erforderlich, um eine Variable semantisch im Kontext Ihrer Verwendung beschreiben zu können. Dies umfasst sowohl die semantische Bedeutung einer Variable und deren zugehörige SI-Einheit [50] als auch die Beschreibung des Kontext der Verwendung der Variablen, wie zum Beispiel die Relation zu einem Medium, dessen Druck beispielsweise gemessen wird, oder aber auch die Festlegung von Kommunikationsbeziehungen, über die diese Variable propagiert werden soll.

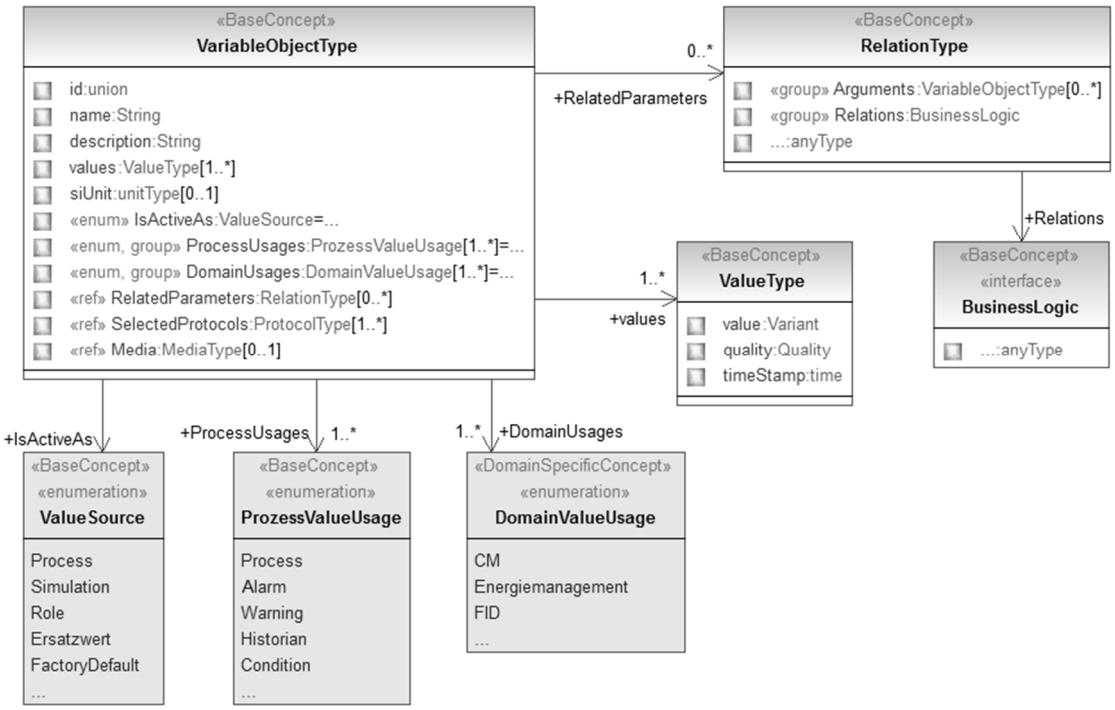


Abbildung 47: Basistypen des VariableObjectType

Die Klasse *VariableObjectType* in Abbildung 47 umfasst verschiedene Attribute, um Variablen generisch beschreiben zu können. Um eine Instanz einer Variablen vom Typ *VariableObjectType* eindeutig identifizieren zu können, werden die Attribute *id, name* und *description* verwendet. Über das Attribut *siUnit* wird die SI-Einheit einer Instanz des *VariableObjectType* definiert. Dem Attribut *values* können Instanzen der Klasse *ValueType* in einer Kardinalität von *[1…*]* zugewiesen werden. Dies ermöglicht die Verknüpfung von einer oder aber auch mehrerer *ValueType*-Instanzen, um

gegebenenfalls einen Werteverlauf darstellen zu können. Darüber hinaus kann der Kontext einer *VariableObjectType*-Instanz durch weitere Referenzen, wie zum Beispiel *RelatedParameters, SelectedProtocols* und *Media* sowie durch Enumerationen, wie zum Beispiel *isActiveAs, ProcessUsages und DomainUsages,* definiert werden.

3.7.3 Klassifizierung der Ausführungsumgebung einer Steuerungslösung

Der Typ „ComputingPower“ spezifiziert Rechenleistung bzw. Speicher von Komponenten. Dabei sollen die zur Verfügung stehenden Speicher- bzw. Rechenkapazitäten sowie Informationen über die Programmierbarkeit und verwendete Programmiersprache abfragbar sein. Wenn eine reale Komponente nicht frei programmierbar ist, kann dort kein beliebiger Programmcode ausgeführt werden. Ein Sensor mit integriertem Mikrokontroller hat zum Beispiel eine bestimmte Rechenleistung, aber keine freie Kapazität und ist nicht programmierbar. Eine Speicherprogrammierbare Steuerung hat freie Rechenleistung und ist frei programmierbar. Ein Informationsserver hingegen hat Rechenleistung, stellt jedoch primär Speicher zur Verfügung.

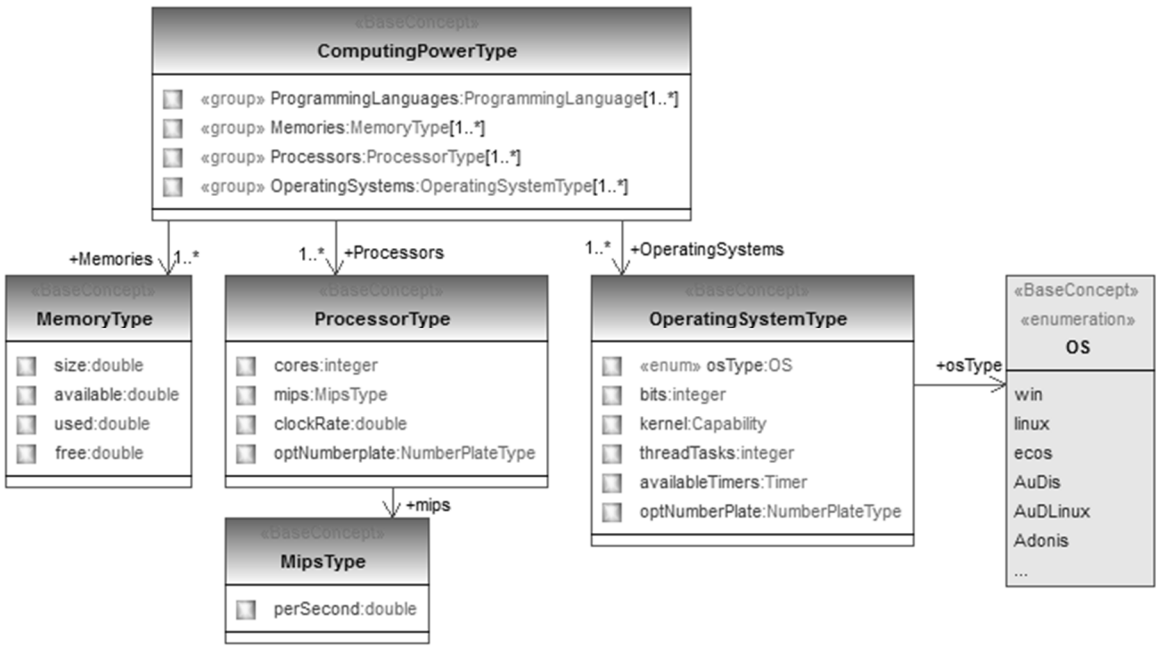


Abbildung 48: Darstellung der Vererbungshierarchie der Klasse ComputingPowerType

3.7.4 Klassifizierung eines digitalen Typenschilds von Automatisierungskomponenten

Wie aus Abbildung 49 ersichtlich, ist mit der *NumberPlateType* eine heute nicht übliche Strukturierung für die Abfrage von statischen Informationen eines digitalen Typenschilds und *factoryDefaults* realisiert. Das *NumberPlate* wird mit seinen Elementen auch für die Spezifizierung der Instanzen von Rollen verwendet. Im *NumberPlate* werden ausschließlich nicht-funktionale Eigenschaften, die jedoch für die Auswahl einer Komponente hilfreich sind, hinterlegt. Als nicht-funktionale Eigenschaft werden

auch Eigenschaften betrachtet, die eine funktionale Implementierung erforderlich machen, wie zum Beispiel Fehlersicherheit, Hochverfügbarkeit und Redundanz. Alle Informationen im *NumberPlate* sind bereits vom Komponentenhersteller hinterlegt und - abgesehen von Firmwareupdates der Komponenten - unveränderlich.

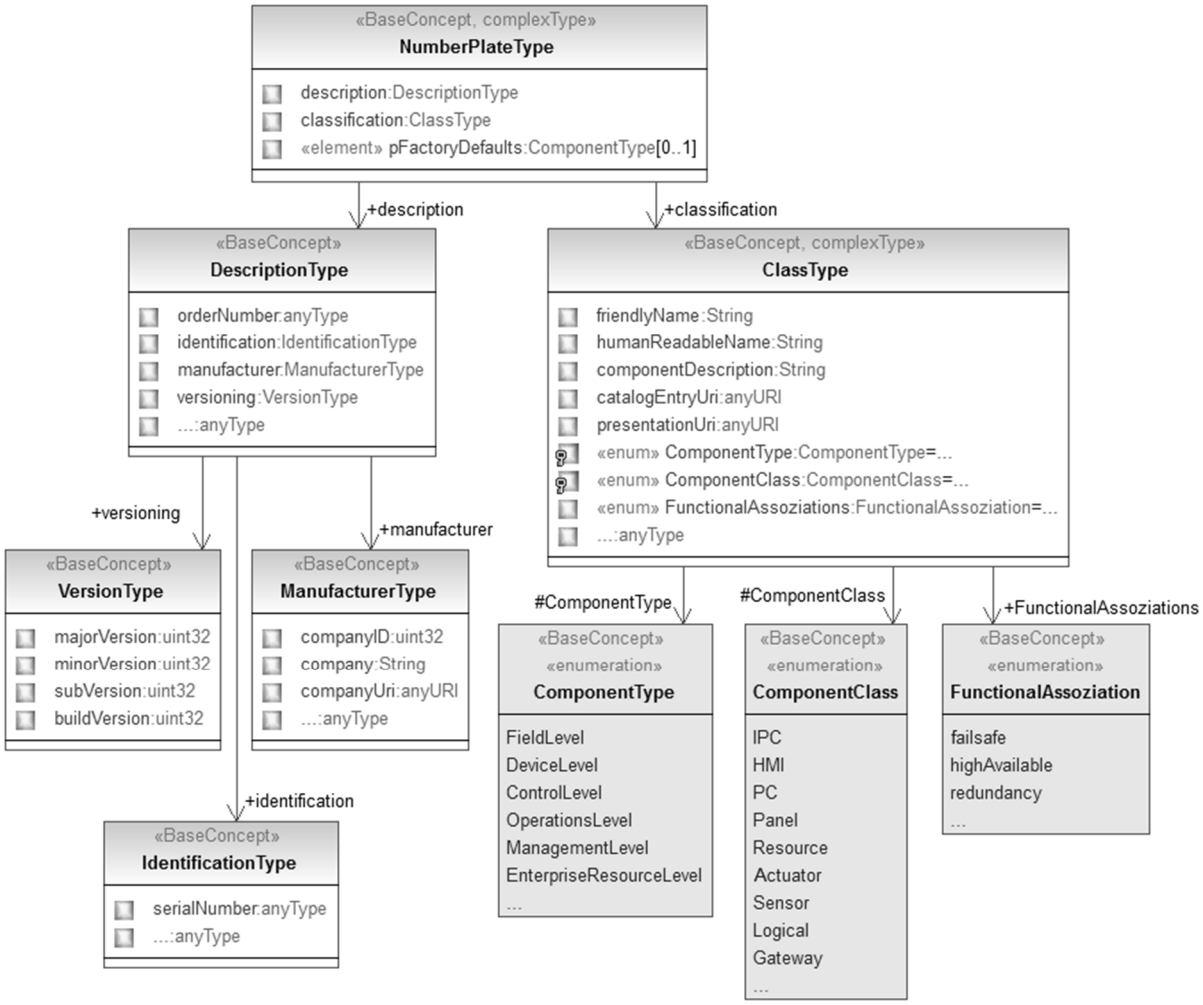


Abbildung 49: Darstellung der Vererbungshierarchie der Klasse NumberPlateType

3.7.5 Klassifizierung von Steuerungsprogrammen

Der Typ „ControlLogicContainerType“ bildet einen Container für die Ausführungslogik einer Komponente. Diese benötigt Rechenleistung zur Ausführung. Eine Steuerungslösung wird durch den Typ *ControlLogicType* spezifiziert. Dieser verwendet Elemente vom Typ *ControlLogicElementType,* Transitionen vom Typ *ControlLogicConnectionType* und klassifiziert die Ablaufsequenz des Steuerungsprogramms durch den Typ *ControlLogicSequenceType*.

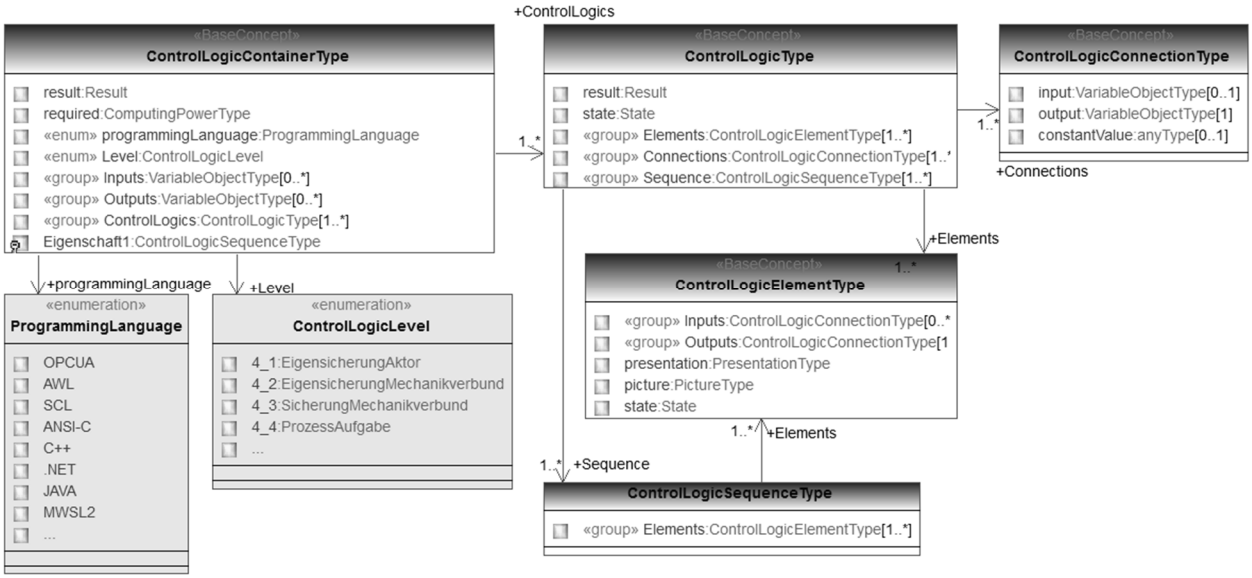


Abbildung 50: Darstellung der Klasse ControlLogicType zur Klassifizierung von Steuerungslösungen

Ziel des hier beschriebenen Teils des Metamodells ist die Referenzierung wesentlicher Elemente einer Steuerungslösung und nicht deren Programmierung. So könnte zum Beispiel ein Steuerungsprogramm nach IEC 61499, bestehend aus einer Vielzahl von Funktionsbausteinen (FB) durch Elemente vom Typ *ControlLogicElementType,* beschrieben werden. Die Beschreibung der Verbindungen zwischen den FBs erfolgt durch Elemente vom Typ *ControlLogicConnectionType*. Die Implementierung bleibt weiterhin in den FBs verborgen, jedoch kann durch dieses Vorgehen jedem FB weitere Semantik angeheftet werden.

3.7.6 Klassifizierung verwendeter Protokolle und Medien einer Komponente

Der Typ „ProtocolType“ spezifiziert den Informationsaustausch von Komponenten. Hierfür müssen physikalische Verbindungen, die verwendeten Protokolle und Ports sowie die benötigten Qualitätsanforderungen definiert werden.

Durch diese Informationen wird eine automatisierte Verteilung des Steuerungsprogramms ermöglicht, da das resultierende Kommunikationsvolumen errechnet bzw. bekannt ist und logische Kommunikationsverbindungen gesucht werden können, die den gegebenen Qualitätsanforderungen entsprechen.

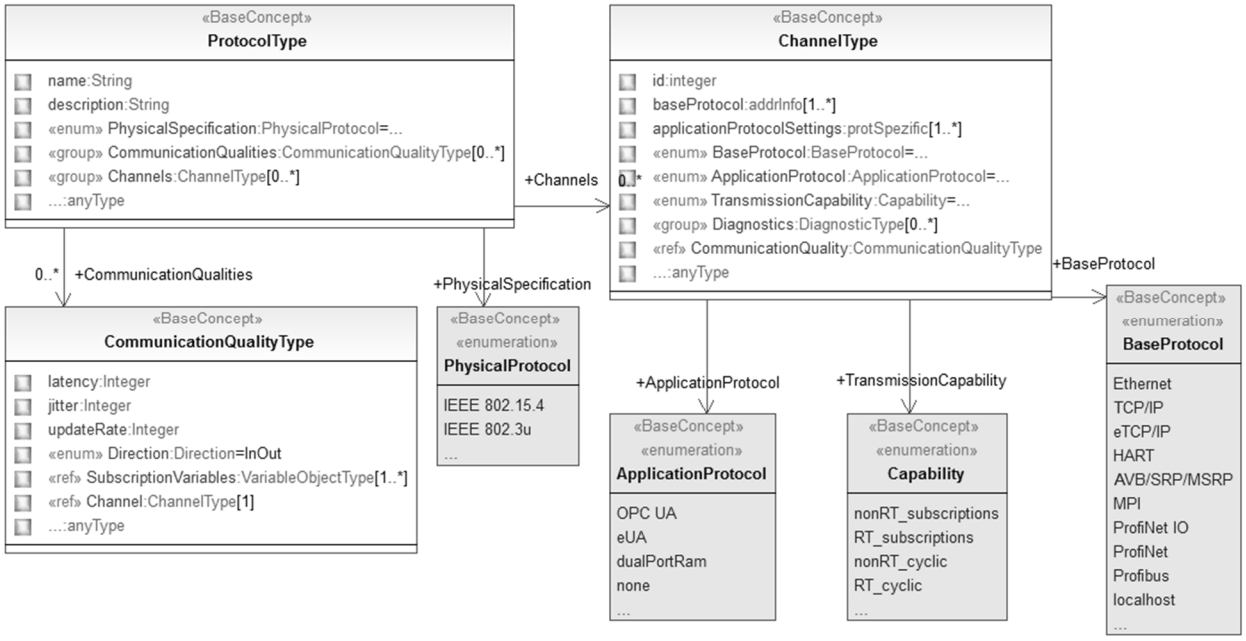


Abbildung 51: Klassifizierung des Übertragungsprotokolls und dessen Qualitätsanforderungen

Der Typ „MediaType“ beschreibt physikalische Medien, die eine Komponente verwendet, beeinflusst oder misst. Die verschiedenen Medien sind hier nur exemplarisch definiert und müssen mit weiteren „Werkstoff“-Konkretisierungen verfeinert werden. Die bereits definierten Grundmedien sind jedoch bereits geeignet, um Medien als spezialisierte Transportwege wie zum Beispiel als Rohr, Band, oder Kupferkabel festlegen zu können.

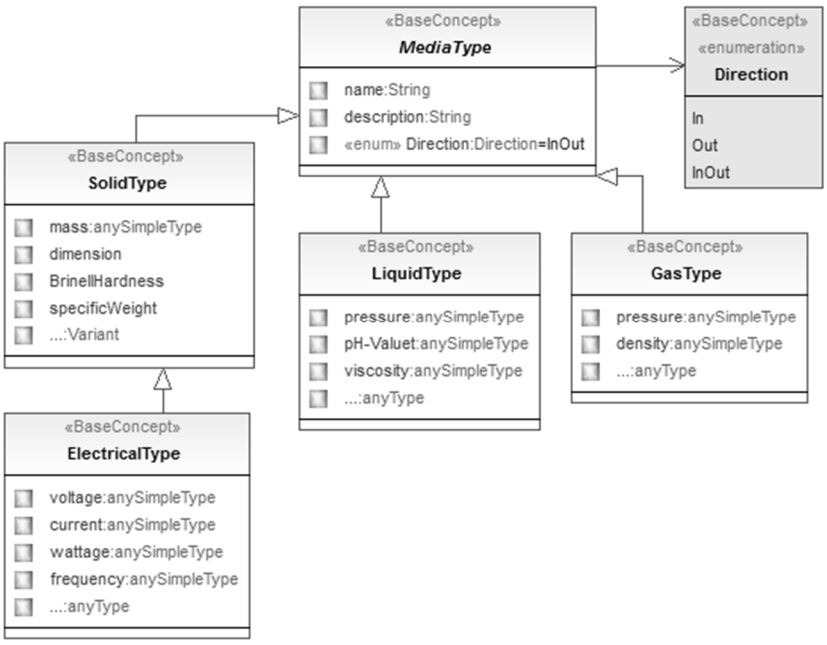


Abbildung 52: Spezialisierung von Medien mit Hilfe des MediaType nach Aggregatszuständen

3.7.7 Klassifizierung der Funktionalität von Komponenten

Der Typ „BaseFunctionType“ gibt Auskunft über die funktionalen Eigenschaften einer Komponente im Sinne der zu lösenden Automatisierungsaufgabe. BaseFunctions können – im Gegensatz zu den Informationen im NumberPlate – erzeugt und gegebenenfalls auch geändert werden.

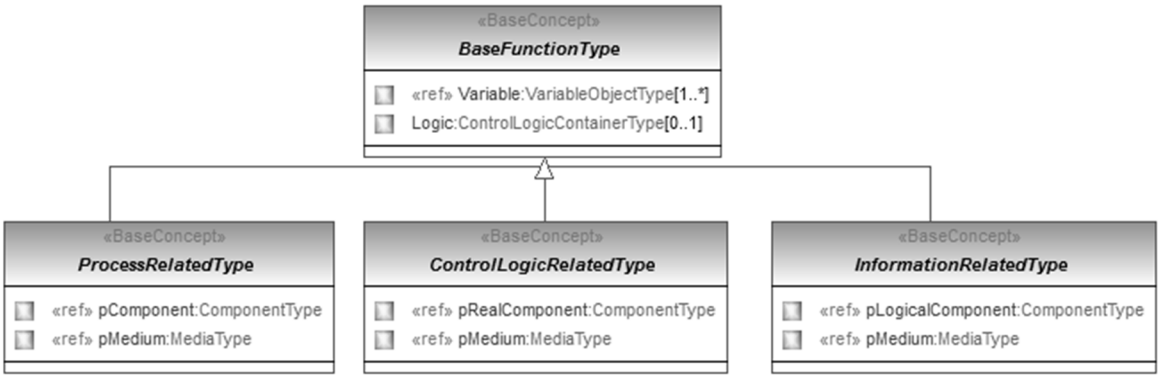


Abbildung 53: Spezialisierung der Grundfunktionalität einer Rolle oder Komponente

Die BaseFunction ist die Interaktionsschnittstelle einer Komponente für deren Automatisierungsaufgabe und kann in prozessbezogene („*ProcessRelatedType*“), steuerungsbezogene („*ControlLogicRelatedType*“) sowie datenverarbeitungsbezogene („*InformationRelatedType*“) Informationen eingeteilt werden. Für diese Typen sind bereits verschiedene Spezialisierungen gemacht worden, wobei in manchen Bereichen auf bereits existierende Standards zur Spezifikation verwiesen werden kann. So kann zum Beispiel der Typ *ProcessRelatedType* im Bereich „Handhaben“ durch die Norm VDI 2860 [88] oder aber auch zu den Bereichen "*Urformen*", "*Umformen*", "*Trennen*", "*Fügen*", "*Beschichten*" und "*Stoffeigenschaft ändern*" durch die Norm DIN 8580 [89] spezialisiert werden.

3.8 Zusammenfassung

In diesem konzeptionellen Kapitel wird zu Beginn die Entwicklung eines semantischen Beschreibungskonzepts dargestellt. Dieses semantische Beschreibungskonzept definiert eine Systemstrukturierung mit Hilfe von Rollen zur Beschreibung von Anforderungen und Komponenten, die die gestellten Anforderungen der Rollen funktional erfüllen. Darüber hinaus wird das Verhalten des mechatronischen Systems beschrieben. Dabei werden semantische Indirektionselemente definiert, die eine Erstellung der Steuerungslogik mit Hilfe dieser semantischen Elemente ermöglicht. Die Indirektionselemente wiederum können mit realer Sensorik bzw. Aktuatorik verbunden werden. Durch dieses Vorgehen wird vermieden, dass in einem Steuerungsprogramm Realparameter individueller Komponenten verwendet werden, da dies ein wesentlicher Hinderungsgrund für einen funktionalen Komponententausch darstellt.

Im weiteren Verlauf wird aufgezeigt, wie aus elementaren Komponenten komplexe, mechatronische Einheiten zusammengesetzt und mittels Funktionsschnittstelle angesprochen werden können.

Im Anschluss wird der semantische Beschreibungsumfang mit Hilfe von UML Klassendiagrammen dargestellt. Diese umfassen die verschiedenen beteiligten Domänen und beschreiben unter Verwendung von objektorientierter Vererbung Typisierungs-, Strukturierungs- und Parametrierungskonzepte.

4 Referenzimplementierung eines Konzepts zur semantischen Beschreibung von Produktionssystemen

Zur Verifizierung der Anwendbarkeit des Konzepts zur semantischen Beschreibung von Automatisierungssystemen wurde eine Referenzimplementierung durchgeführt. Das Ziel dabei war die Adaption des Konzepts an bestehende Engineering-Werkzeuge und die prototypische Umsetzung der Ausleitung der Semantik des Engineerings in ein Informationsmodell, das wiederum von einer Ablaufumgebung (SPS) verwendet werden kann.

4.1 Überblick über das Realisierungskonzept

Das Realisierungskonzept orientiert sich weitestgehend an dem theoretisch beschriebenen Konzept (vgl. Kapitel 3.1), wurde jedoch an technischen Rahmenbedingungen angepasst. Das Realisierungskonzept gliedert sich in die drei Teile

- Engineering,
- Informationsmodell und
- Ablaufumgebung.

Als Engineering Software wird die CAD-Software NX der Firma Siemens mit dem Erweiterungsmodul Mechatronics Concept Designer gewählt. Die Programmierschnittstelle bietet weitreichende Zugriffsmöglichkeiten auf mechanische und kinematische Projektierungsinformationen sowie auf elementare Ablaufinformationen des Prozesses.

Für die Umsetzung der Referenzimplementierung des Informationsmodells wird eine technische Realisierungsmöglichkeit gesucht, die sowohl im Engineering als auch auf Steuerungsebene eingesetzt werden kann. OPC UA bietet umfangreiche semantische Beschreibungsmöglichkeiten und verfügt darüber hinaus über Kommunikations- und Sicherheitsmechanismen (z. B. verschlüsselte Kommunikation bis auf Applikationsebene sowie User-Authentication). Aus diesen genannten Gründen wird OPC UA zur Realisierung des semantischen Informationsmodells verwendet.

Die Ablaufumgebung basiert auf einer bereits bestehenden Hardware- und Betriebssystemarchitektur und wird in den nachfolgenden Kapiteln um die Fähigkeit der Ausführung von Ablaufsequenzen erweitert.

Die jeweiligen Teile der Referenzimplementierung sind in den nachfolgenden Abschnitten detailliert beschrieben.

4.2 Referenzimplementierung als PlugIn der CAD-Software NX

Das vorgestellte Konzept zur mechatronischen Projektierung von automatisierten Produktionssystemen soll anhand einer Referenzimplementierung erprobt und evaluiert werden. Hierfür sollen bereits verfügbare Engineering-Werkzeuge erweitert bzw. angepasst werden. Zur Umsetzung der Referenzimplementierung wird NX der Firma Siemens verwendet, da NX über Erweiterungsmodule verfügt, die eine Kinematisierung und Simulation von Geometriedaten sowie die erforderliche Flexibilität für konzeptbedingte Anpassungen mittels Programmierschnittstelle ermöglicht.

4.2.1 Einsatz von NX als Werkzeug zur Produktentwicklung

Für die Umsetzung der Referenzimplementierung wird NX 10.0 verwendet. NX integriert neben den klassischen M-CAD Funktionalitäten auch Erweiterungsmodule, die ein disziplinübergreifendes Engineering erlauben. Auf der Plattform von NX können die verschiedenen Werkzeuge für Computer Aided Engineering (CAE) untereinander Daten nahtlos austauschen. Als zentrale Informationsdrehscheibe für Produkt- und Prozesswissen kann Teamcenter als zentrale „collaborative Product Development Management“-Lösung (cPDM) eingesetzt werden. In nachfolgender Abbildung 54 ist die Bedienoberfläche von NX 10 dargestellt.

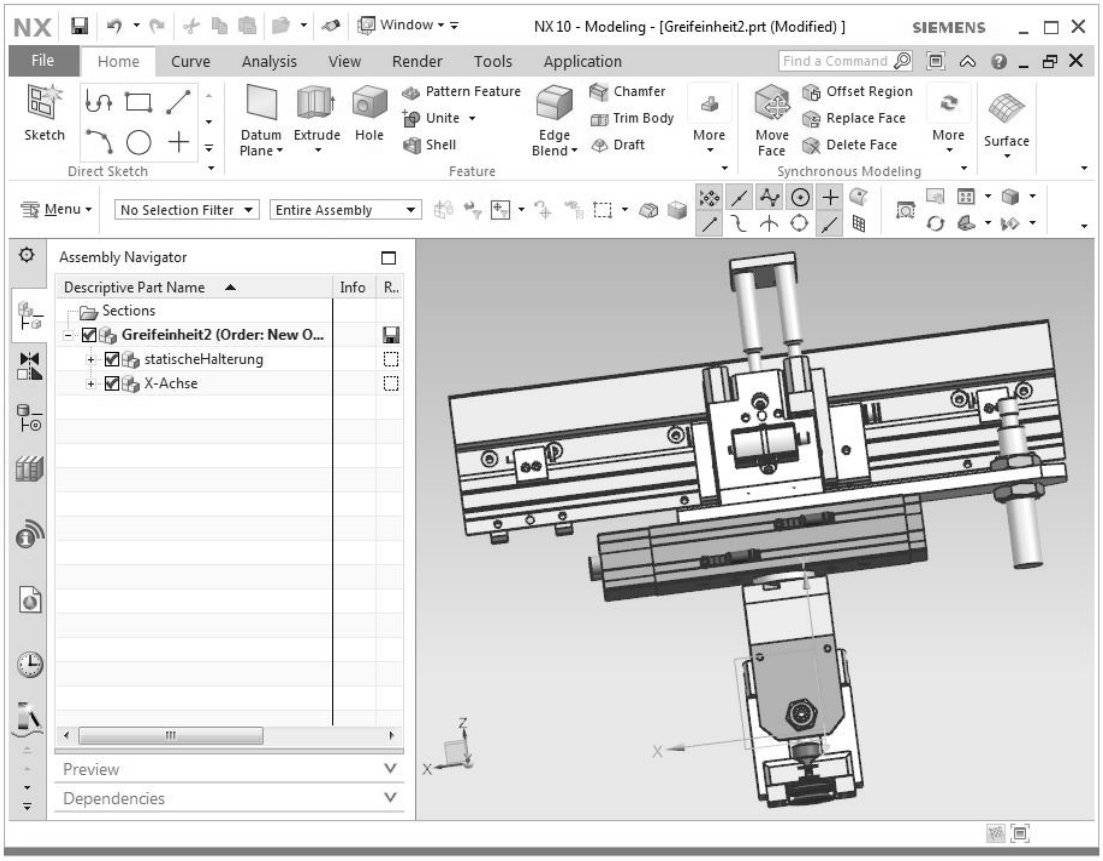


Abbildung 54: Benutzeroberfläche von NX 10

Für die Konstruktion bietet NX spezifische Modellierungswerkzeuge und erlaubt eine integrierte Konstruktion mechanischer, elektronischer sowie elektrischer Bauteile. Diese können in Baugruppen zu komplexen mechatronischen Einheiten aggregiert werden. Neben diesen Fähigkeiten bietet NX umfangreiche Werkzeuge zur Simulation und Berechnung von Strukturen, Strömungen, Wärmeübertragungen, Bewegungen, Optimierungen und für Multiphysikberechnungen. Da diese unterschiedlichen Simulationen in NX als Plattform integriert sind, ist eine schnelle und effiziente Anpassung am Modell in den verschiedenen Domänen möglich.

Die in Abbildung 54 dargestellte Handhabungseinheit verfügt über eine modulare Baugruppenstruktur (siehe Abbildung 55), die sich an den mechanischen Abhängigkeiten und einer funktionalen Aggregation orientiert. Ziel dabei ist es, die Komponentenstruktur so aufzubauen, dass die kinematischen Abhängigkeiten abgebildet werden. So ist zum Beispiel in Abbildung 55 ersichtlich, dass unter der Baugruppe „X-Z-Achs-Rot-GE“ eine Baugruppe „Z-Achs-Rot-GE“ angeordnet ist.

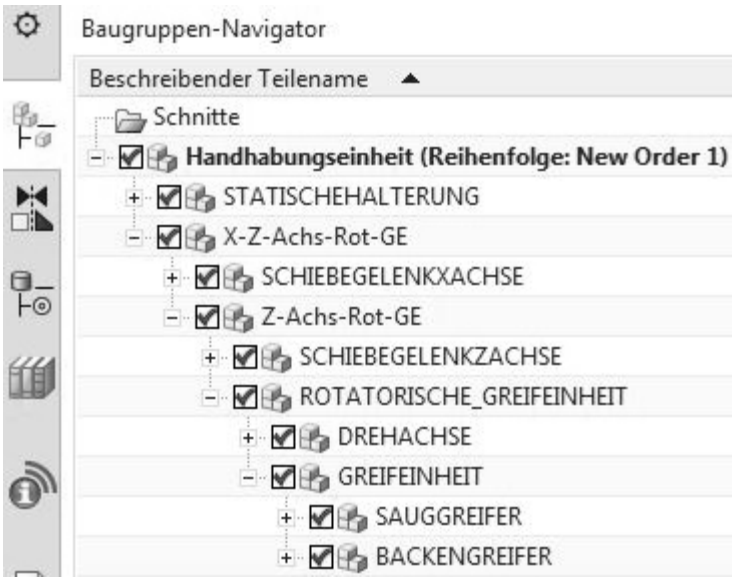


Abbildung 55: Darstellung der hierarchischen Baugruppenstruktur im Baugruppen-Navigator von NX

Da diese „Z-Achs-Rot-GE“ als Komponente der Baugruppe „X-Z-Achs-Rot-GE“ modelliert wurde, ist ersichtlich, dass die „Z-Achs-Rot-GE“ von der „X-Z-Achs-Rot-GE“ mechanisch abhängig ist. Der Vorteil dieser Modellierung ist eine strikte Modularisierung nach funktionalen Aspekten. Dies unterstützt zum Beispiel Ansätze des funktionalen Engineerings sowie des funktionalen Komponententauschs.

4.2.2 Verwendung des Erweiterungsmoduls „Mechatronics Concept Designer“

Die mechanische Konstruktion wird in NX unter anderem durch das Modul „Mechatronics Concept Designer“ (MCD) erweitert. Ziel des MCD ist es, in einer frühen Konzeptphase bereits die Mechanik und Kinematik im Sinne des Prozesses evaluieren zu können. So erlaubt der MCD bereits in einer frühen Konzeptphase eine Kinematisierung und Simulation der Geometriedaten aus der Konstruktion unter Verwendung vereinfachter Sensorik und Aktuatorik der Automatisierungstechnik.

Wie in Abbildung 56 dargestellt, können beliebige Geometriekörper aus der mechanischen Konstruktion zu physikalischen Körpern (❶) erweitert werden. Diese physikalischen Körper werden durch die im MCD integrierte Physics-Engine berücksichtigt. So wirken auf die physikalischen Körper zum Beispiel Schwerkraft, Gleit- und Haftreibung, die bei der Simulation berücksichtigt werden.

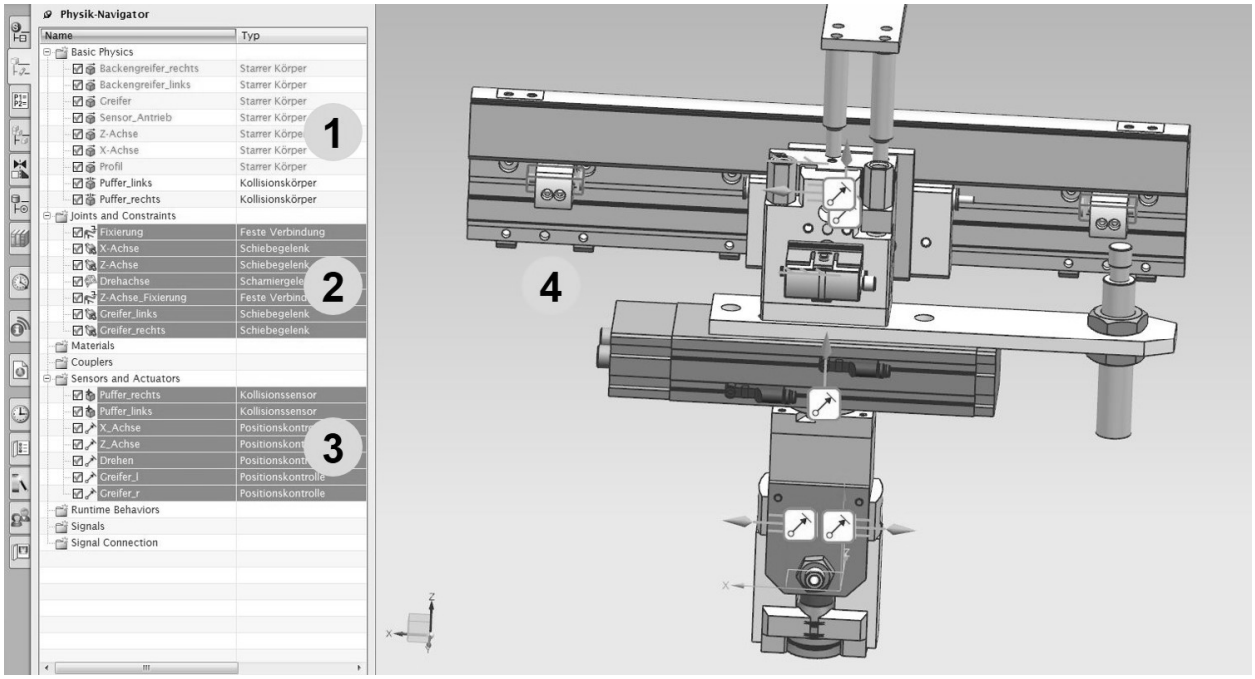


Abbildung 56: Darstellung der Benutzeroberfläche des Mechatronics Concept Designers mit den physikalischen (❶) und kinematischen (❷) Objekten, Sensorik und Aktuatorik (❸) im Physik-Navigator sowie der 3D-Geometriedarstellung (❹).

Die physikalischen Körper werden mittels Gelenke (❷) verknüpft, wodurch das Geometriemodell nachträglich kinematisiert werden kann. Hier stehen folgende verschiedenen Gelenkarten zur Verfügung:

- Scharniergelenk
- Schiebegelenk

- Zylindrisches Gelenk
- Kugelgelenk

Mit dem Kinematikmodell können Positions- oder Geschwindigkeitskontrollen und Kollisionssensoren (❸) verknüpft werden. Im Sinne der Automatisierungstechnik repräsentieren diese im MCD Aktuatorik bzw. Sensorik, da sie auf Position bzw. Geschwindigkeit der Gelenke Einfluss nehmen. Positions- oder Geschwindigkeitskontrollen interpolieren Vorgabewerte auf das verknüpfte Gelenk, sodass eine animierte Bewegung entsteht. Dabei können folgende Parameter berücksichtigt werden:

- Maximales Drehmoment (rotatorisches Gelenk)
- Maximal wirkende Kraft (translatorisches Gelenk)
- Maximale Beschleunigung (rotatorisches oder translatorisches Gelenk)
- Maximale Geschwindigkeit (rotatorisches oder translatorisches Gelenk)

Der MCD integriert einen Sequenz-Editor (siehe Abbildung 57), der verschiedene Operationen nebenläufig oder auch sequenziell verknüpfen kann.

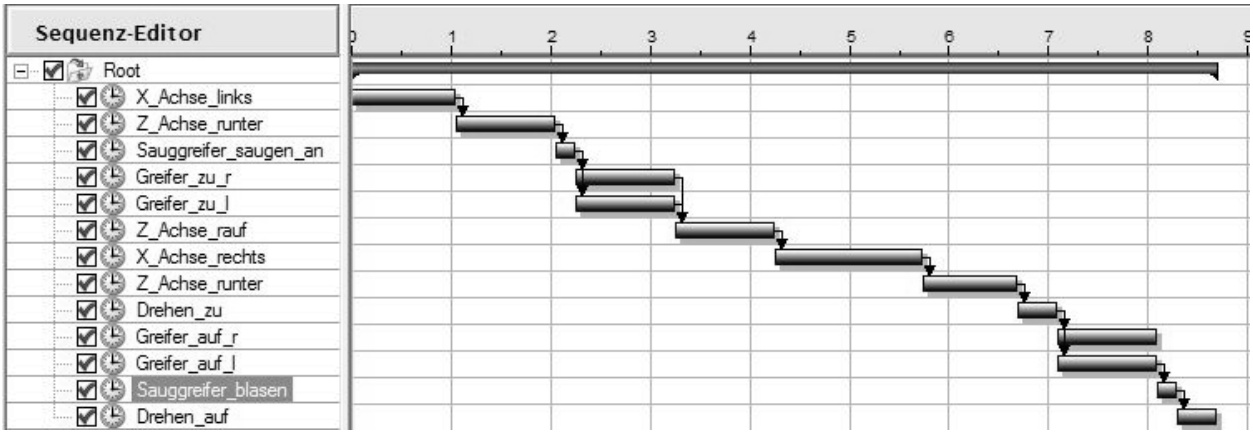


Abbildung 57: Gantt-Chart im Sequenz-Editor des MCD mit Operationen (Balken) und Übergängen (Pfeile)

Mittels Gantt-Chart wird eine Ablaufsequenz für das Kinematik-Modell definiert. Eine Operation (blauer Balken) enthält dabei Vorgabewerte für Positions- oder Geschwindigkeitskontrollen des MCD. Die Balkenlänge entspricht der benötigten Zeit, die für die Ausführung einer Operation erforderlich ist. Durch die Verknüpfung verschiedener Operationen wird der Gesamtablauf des Prozesses realisiert.

4.2.3 Erweiterung der Funktionalität von NX und dem „Mechatronics Concept Designer“

Um das in Kapitel 3.1 beschriebene Konzept in ausgewählten Bereichen mit Hilfe von NX sowie dem Mechatronics Concept Designer umzusetzen, müssen bestimmte Funk-

tionalitäten neu implementiert werden. Solche benutzerspezifischen Anpassungen werden von NX sowohl durch das Benutzeroberflächenwerkzeug BlockStyler als auch durch die Programmier-API NXOpen unterstützt. Diese werden nachfolgend kurz beschrieben.

Gestalten von Benutzeroberflächen mit dem BlockStyler

Zur Erstellung von benutzerdefinierten Oberflächen steht in NX der sogenannte BlockStyler zur Verfügung. Dieser ermöglicht die visuelle Gestaltung von Benutzeroberflächen, die im Stil von NX blockbasiert angelegt werden können. Wie in Abbildung 58 dargestellt, kann ein Dialogfenster (rot umrandet) aus mehreren Blöcken (grün umrandet) zusammengesetzt werden.

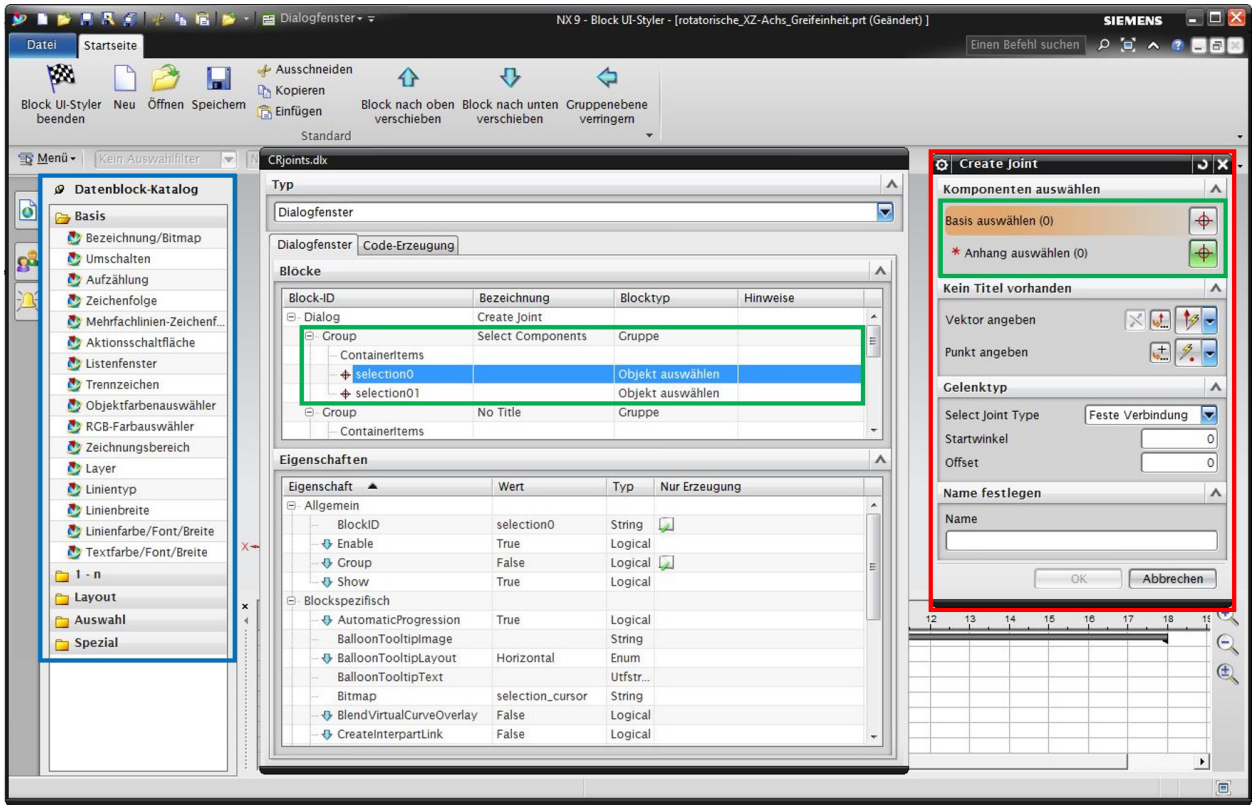


Abbildung 58: Benutzeroberfläche des Blockstylers

Die einzelnen Elemente eines Blocks zum Beispiel zur Auswahl bestimmter Objekttypen in NX stehen bereits im Datenblock-Katalog (blau umrandet) vorgefertigt zur Verfügung. Dies bietet den Vorteil, dass im benutzerdefinierten Dialog über die Datenblock-Bausteine gezielt bestimmte Objekttypen in NX selektiert werden können, die dann mittels benutzerspezifischer Algorithmen des Dialogs benutzerspezifisch verarbeitet werden können.

Nach Erstellung des benutzerspezifischen Dialogs im BlockStyler (in Abbildung 58 rot umrandet) kann dieser als Datei mit der Dateiendung.*dlx* abgespeichert und anschließend durch die Programmier-API NXOpen verwendet werden.

Implementierung von benutzerdefinierten Funktionen mit der Programmier-API NXOpen von NX

Die Programmier-API NXOpen erlaubt die benutzerspezifische Implementierung von Routinen und Funktionen, die dann in NX ausgeführt werden können. Bei der API handelt es sich um die sogenannte Common API von NX, welche die Programmiersprachen C++, C#, VB .NET und Java umfasst. Zur funktionalen Erweiterung von NX wird die NXOpen C++-API verwendet, die weitreichende Zugriffsmöglichkeiten auf die Objektstrukturen von NX und integrierter Module wie zum Beispiel dem MCD bietet. Somit können mittels benutzerdefinierter Funktionen Objekte in NX bzw. dem MCD angelegt, gelöscht, geändert sowie deren Parameter ausgelesen oder neu gesetzt werden. Die Interaktion mit dem Benutzer wird dabei über Dialoge realisiert, die mittels BlockStyler individuell erzeugt und verwendet werden können. Die Programmier-API NXOpen und die im BlockStyler erstellen benutzerdefinierten Dialoge dienen als technische Grundlage, um die bereits vorhandenen Funktionen in NX und dem MCD zu nutzen sowie die Funktionalität so zu erweitern, dass das vorgestellte Konzept (siehe Kapitel 3.1) in ausgewählten Bereichen auf NX und den MCD adaptiert werden kann.

Umsetzung einer komponentenbasierten, funktionalen Strukturierung im Mechatronics Concept Designer

Ein Grundgedanke des in Kapitel 3.1 dargestellten Konzepts zur semantischen Beschreibung von automatisierten Produktionssystemen ist eine komponentenbasierte Modularisierung. Darunter wird verstanden, dass sich eine Maschine zum Beispiel aus verschiedenen mechatronischen Komponenten zusammensetzt. Dabei soll die Komplexität dadurch reduziert werden, dass Teilfunktionen der Gesamtfunktionalität einer Maschine in unterlagerten Komponenten realisiert und über eine Funktionsschnittstelle funktional angeboten werden können. Somit kann eine überlagerte Komponente die Funktionen ihrer unterlagerten Komponenten bei Bedarf aufrufen. Durch dieses Aggregationsprinzip können komplexe Funktionalitäten mechatronischer Komponenten durch Aggregation untergeordneter Komponenten realisiert werden. Dies kann auch als funktionale Aggregation bezeichnet werden. Diese Art der Strukturierung unterstützt einen funktionellen Komponententausch, da die ausgetauschte Komponente lediglich die gleiche Funktionsschnittstelle anbieten muss, damit die Aufrufstruktur der Steuerungslogik erhalten bleibt. Darüber hinaus muss das Verhalten der neuen Komponente dem der alten Komponente entsprechen. Die technische Realisierung kann sich dabei in der neuen Komponente ändern. Diese komponentenbasierte Sichtweise sowie eine funktionale Aggregation sind aktuell seitens des Mechatronics Concept Designers nicht direkt

vorgesehen. Projektierungsinformationen werden auf einer Ebene ohne Aggregation oder Gliederung in mechatronische Komponenten vorgenommen. Da die Objekte des Mechatronics Concept Designers mit den Geometriedaten von NX in Relation steht, kann die Baugruppenstruktur von NX verwendet werden. Die Abhängigkeiten zwischen den Objekten im Mechatronics Concept Designer bis hin zu NX stellt sich wie folgt dar:

1. Es existiert in NX ein Bauteil oder eine Baugruppe.
2. Im MCD werden aus einem oder mehreren Bauteilen (bzw. Baugruppen) physikalische Körper (siehe Abbildung 56; ❶) erstellt.
3. Im MCD werden mehrere physikalische Körper mittels Gelenkobjekte (siehe Abbildung 56; ❷) zu einem kinematisierten Modell verknüpft.
4. Im MCD wird mit den Gelenkobjekten jeweils eine Geschwindigkeits- oder Positionskontrolle (siehe Abbildung 56; ❸) verknüpft. Diese ist in der Lage, Positionswerte oder aber auch Geschwindigkeitswerte über ein bestimmtes Zeitintervall hinweg zu interpolieren, sodass sich in der Simulation eine gleichförmige Bewegung über dieses Zeitintervall hinweg ergibt.
5. Im MCD werden Geschwindigkeits- oder Positionskontrollen über die Operationen des Gantt-Charts (siehe Abbildung 57) parametriert. Dadurch ergibt sich aus der Ablaufsequenz des Gantt-Charts das Bewegungsverhalten der betrachteten Maschine.

Über die in den Punkten 1. bis 5. dargestellten Abhängigkeiten lässt sich eine Zuordnung von Operationen des Gantt-Charts im Mechatronics Concept Designer zu bestimmten Bauteilen bzw. Baugruppen vornehmen. Durch die mechanische Strukturierung und durch die Aufrufreihenfolge des Gantt-Charts können funktionale Schnittstellen für die einzelnen Komponenten abgeleitet werden. Diese Funktionsschnittstellen beschreiben semantisch die gewünschte Funktionalität, abstrahieren jedoch die Realisierung bzw. Implementierung im Gegensatz zu Operationen des Gantt-Charts im MCD. Als Beispiel kann hier ein einfaches Modell eines Backengreifsystems mit zwei beweglichen Greifbacken genannt werden. Um den Greifer in der Simulation des MCDs zu schließen, sind zwei Operationen zur Realisierung der Schließbewegung beider Greifbacken erforderlich. Die eigentliche singuläre Intention „Greifer schließen“ geht aus funktionaler Sicht dabei verloren. Die Verwendung von Komponenten mit Funktionsschnittstellen ist ein wesentlicher Schritt zur Umsetzung eines funktionalen Komponententauschs.

Wie in Abbildung 59 dargestellt, definiert die Funktion $f_{E,1}$ eine Ablaufsequenz der unterlagerten Funktionen $f_{C,x}$ sowie $f_{D,y}$. Durch funktionalen Komponententausch der Komponente *A*, die durch die Komponente *Z* ersetzt wurde, ändert sich lediglich die Ab-

laufsequenz der direkt überlagerten Komponente *C*. Die Funktionsaufrufstruktur der Komponente *E* bleibt unbeeinflusst.

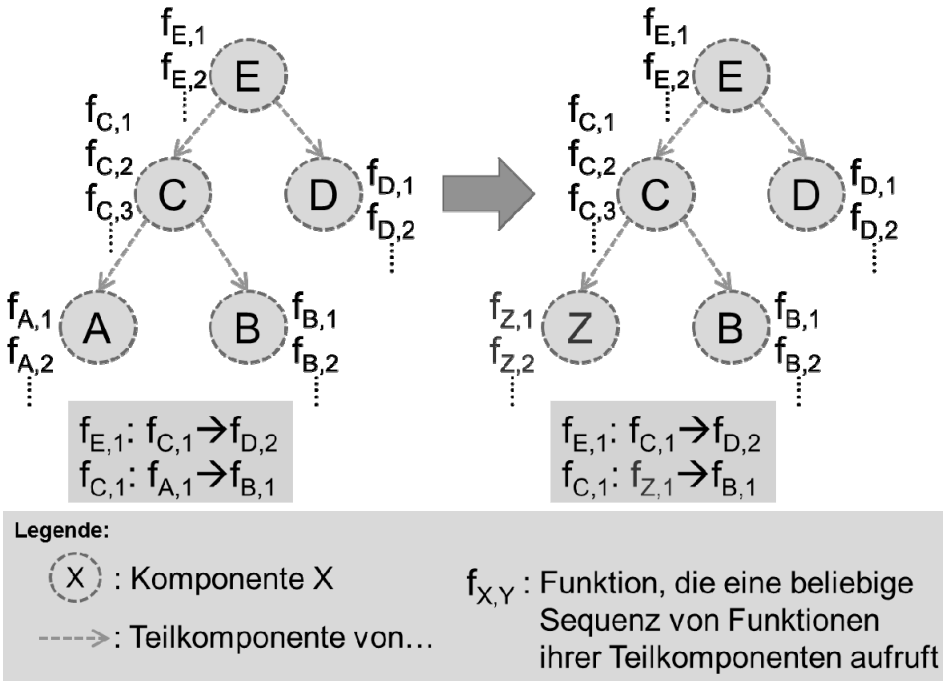


Abbildung 59: Darstellung einer Komponentenhierarchie mit komponentenspezifischen Funktionen sowie resultierenden Änderungen bei funktionalem Komponententausch

4.2.4 Anbindung eines OPC UA Informationsmodells an NX

Um die Anbindung eines semantischen Informationsmodells an Engineering-Werkzeugen zu realisieren, wurde als Standard OPC UA (IEC 62541) zur technischen Umsetzung gewählt, da OPC UA die bereits in Kapitel 2.1.4 dargelegten vorteilhaften Eigenschaften besitzt. Ziel ist also eine möglichst generische Anbindung von Engineering-Werkzeugen an OPC UA Informationsmodelle. Hierbei gibt es zwei mögliche Vorgehensweisen. Erstens ist es mittels Anbindung eines OPC UA Servers an ein Engineering-Werkzeug möglich, dass dieses ein Informationsmodell selbst zur Verfügung stellt, mit dem sich andere Clients verbinden können. Zweitens soll es einem Engineering-Werkzeug auch möglich sein, sich mittels Clientmechanismen mit einem bestehenden OPC UA Informationsmodell zu konnektieren, um dieses kollaborativ zu erweitern. Bei der Referenzimplementierung wurde exemplarisch die Software NX von Siemens inklusive des Erweiterungsmoduls Mechatronics Concept Designer eingesetzt und ein OPC UA Server angebunden.

In Abbildung 60 ist ein Überblick über das Konzept zur generischen Anbindung eines OPC UA Servers oder Clients an ein Engineering-Tool dargestellt.

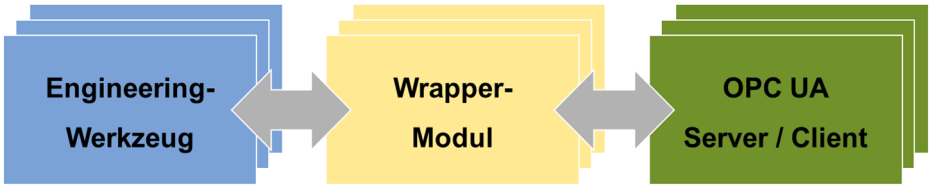


Abbildung 60: Überblick über ein generisches Konzept zur Anbindung eines OPC UA Servers an Engineering-Werkzeuge

In der Mitte zwischen Engineering-Werkzeug und OPC UA Server bzw. Client ist ein Wrappermodul dargestellt. Dieses Wrappermodul hat die Aufgabe, spezifische Aufrufe des Engineering-Werkzeugs mit spezialisierten Datentypen in einen funktionalen Aufruf des OPC UA Servers oder Clients unter Berücksichtigung der OPC UA Datentypen zu übersetzen.

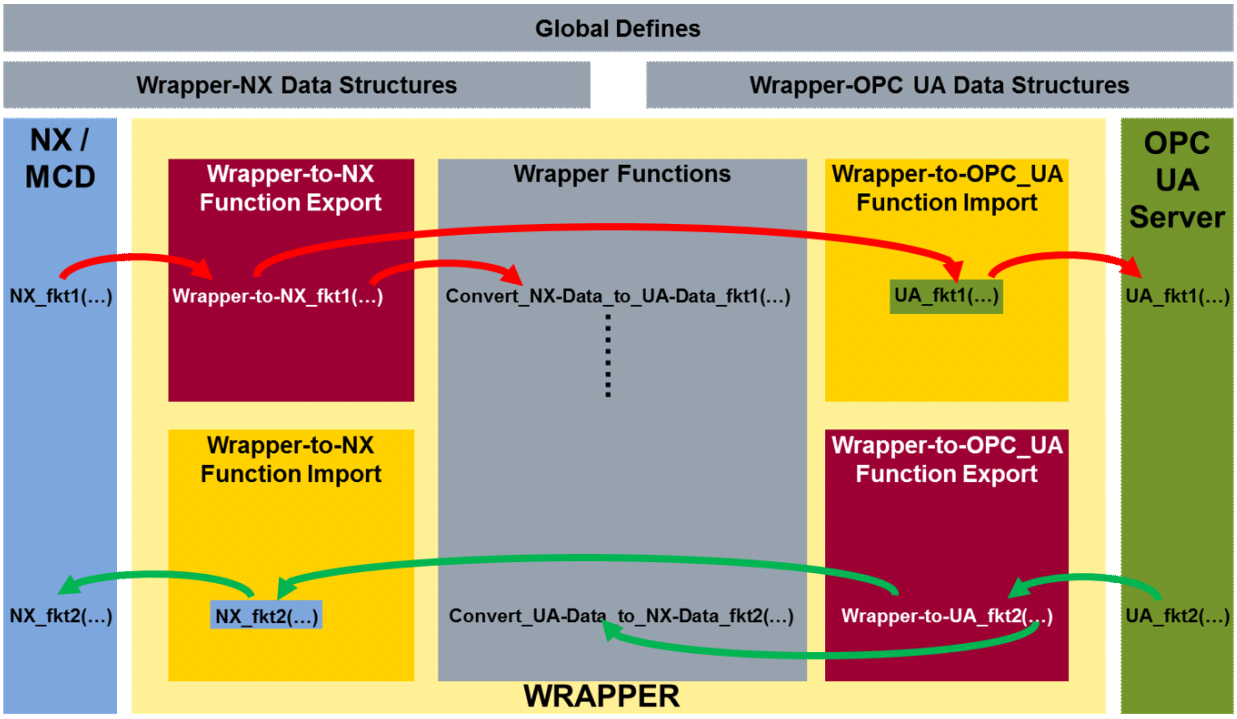


Abbildung 61: Architektur-Übersicht des Wrappers zwischen NX/MCD und einem OPC UA Server

Somit steht seitens des Wrapper-Moduls eine Funktionsschnittstelle in Richtung Engineering-Werkzeug zur Verfügung, die keine Kenntnisse des OPC UA Standards erfordern. Das Wrapper-Modul muss dafür an jedes Engineering-Werkzeug individuell angepasst werden, da im Wrapper-Modul zum Beispiel die Datenkonvertierung zwischen Engineering und OPC UA einmalig festgelegt und implementiert werden muss.

In Abbildung 61 ist die Architekturübersicht des implementierten Wrapper-Moduls für NX und dem Mechatronics Concept Designer dargestellt, auf die an dieser Stelle kurz eingegangen werden soll. Das Wrapper-Modul umfasst zwei Funktionsschnittstellen, wobei die eine dem OPC UA Server, die andere der Engineering-Applikation mittels Funktionsexport Wrapperfunktionen zur Verfügung stellt. Des Weiteren verwendet das Wrappermodul Funktionen mittels Funktionsimport, die seitens des OPC UA Servers und seitens der Engineering-Applikation exportiert werden.

Über diese funktionalen Schnittstellen erfolgt die bidirektionale Nutzung des OPC UA Servers zur Speicherung bzw. zum Abruf von Informationen durch die Engineering-Applikation. Diese ist für Start und Stopp sowie für die semantisch korrekte Generierung des Instanzraums des OPC UA Servers verantwortlich. So stellt der OPC UA Server ein vordefiniertes, spezialisiertes Typsystem zur Verfügung, das mittels der Funktionsschnittstelle zur dynamischen Generierung des Instanzraums verwendet wird. Das Typsystem umfasst spezialisierte Objekt-, Referenz- und Datentypen und wird in Kapitel 4.3 näher erläutert. Die dynamische Erstellung eines Informationsmodells im OPC UA Server muss seitens der Engineering-Applikation mit dem gleichen semantischen Modellverständnis erfolgen, da sonst eine Diskrepanz zwischen Intension der semantischen Beschreibungselemente des OPC UA Typsystems und deren Verwendung resultieren würde. Somit muss die semantisch korrekte Verwendung der Funktionsschnittstelle seitens der Engineering-Applikation gewährleistet werden.

4.3 Spezifikation des Typsystems des OPC UA Informationsmodells

Das Informationsmodell definiert die Elemente zur semantischen Beschreibung von Automatisierungssystemen. Dies erfolgt mittels spezialisierter Objekttypen, Referenzen und Datentypen, die in den nachfolgenden Abschnitten näher erläutert werden.

4.3.1 Spezialisierte Objekttypen des OPC UA Informationsmodells

Für die Abbildung der Objekte im Engineering (zum Beispiel Bauteile/Baugruppen in NX, Gelenke und physikalische Körper im MCD) wurden spezialisierte Objekttypen in einem eigenen OPC UA Namensraum definiert. Mittels dieser Objekttypen ist eine Abbildung der Projektierungsinformationen aus NX/MCD in das OPC UA Informationsmodell möglich. Die hierfür definierten Objekttypen werden nachfolgend vorgestellt.

Definition eines Objekttyps zur Festlegung von physikalischen Körpern als Grundlage für eine Kinematisierung

Vom Objekttyp „BodyType“ leiten sich zwei spezialisierte Objekttypen ab. Der „CollisionBodyType“ dient zur Typisierung von Kollisionskörpern, der RigidBodyType wird für die Typisierung von starren Körpern verwendet, die beide im MCD zur Simulation verwendet werden.

Definition eines Objekttyps zur Festlegung der Bewegungsinterpolation eines Gelenks

Der „ControlType“ generalisiert die spezialisierten Typen „PositionControlType“ und „SpeedControlType“. Diese werden für die Abbildung von Positions- bzw. Geschwindigkeitskontrollen des MCD verwendet. Die Positions- bzw. Geschwindigkeitskontrollen interpolieren Vorgabewerte auf die einzelnen Gelenke, sodass eine kontinuierliche Bewegung im Rahmen der voreingestellten Bedingungen der Physics-Engine resultiert.

Definition eines Objekttyps zur Festlegung des Bewegungsprofils für ein gekoppeltes Bewegungsverhalten von Automatisierungskomponenten

Der „CouplerProfileType“ definiert den Typ für die Abbildung von mechanischen Kopplungsprofilen im MCD. Diese Kopplungs-Profile dienen der Festlegung von dynamischen Kopplungsverhältnissen gemäß dem definierten Profil und sind mit einer Kurvenscheibe vergleichbar.

Definition eines Objekttyps zur Festlegung eines gekoppelten Bewegungsverhaltens von Automatisierungskomponenten

Der in Abbildung 62 abgebildete „CouplerType“ dient der Abbildung eines Kopplungselements des MCD in das OPC UA Informationsmodell. Davon abgeleitet gibt es drei spezialisierte Coupler-Typen:

- *ControlCoupler*: Bildet das Kopplungselement „Elektronische Kurvenscheibe“ in Abhängigkeit eines Antriebs ab
- *ElectricalCoupler*: Bildet das Kopplungselement „Elektronische Kurvenscheibe“ in Abhängigkeit zur Laufzeit ab

MechanicalCoupler: Bildet das Kopplungselement „Mechanische Kurvenscheibe“ ab

Die drei spezialisierten Typen erben bei Instanziierung die unter dem CouplerType definierten Variablen Offset_Master, Offset_Slave, Scale_Master und Scale_Slave. Diese haben folgende Bedeutung:

- *Offset_Master:* Definiert den Offset-Wert des Master-Gelenks
- *Offset_Slave:* Definiert den Offset-Wert des Slave-Gelenks
- *Scale_Master / Scale_Slave:* Definiert das Kopplungsverhältnis zwischen Master und Slave im Verhältnis Scale_Master zu Scale_Slave

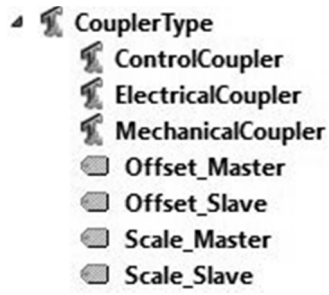


Abbildung 62: Darstellung des Objekttyps "CouplerType"

Definition eines Objekttyps zur Festlegung der Funktionsschnittstelle von Automatisierungskomponenten

Zur Typisierung der funktionalen Schnittstelle von Komponenten dient der Typ *„FunctionType"*. Diese enthält die Variable *trigger* zur Auslösung der spezifizierten Funktionalität sowie die Variable *callback*, die eine vollständige Ausführung der Funktion signalisiert. Die Variablen *isActive, Reset* und *Emergency_Stop* werden zur Verknüpfung mit dem INIT-Servicebaustein in der Ablaufumgebung benötigt (siehe Abbildung 74).

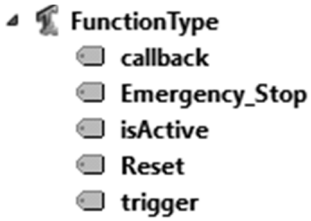


Abbildung 63: Darstellung des Objekttyps "FunctionType"

Definition eines Objekttyps zur Festlegung möglicher Systemzustände von Automatisierungskomponenten

Der Objekttyp *„IAPType"* charakterisiert die Abbildung des Interaktionspunkts in das OPC UA Informationsmodell. Eine Instanz des IAPType trägt eine bestimmte semantische Bedeutung, die erfüllt oder nicht erfüllt sein kann. Dies wird über die Variable *Condition_Fulfilled* ausgedrückt. Für die Abbildung des MCD auf das IAP-Konzept ist eine Spezialisierung des *IAPType* erforderlich, um die Abbildung von Kollisionssensoren und Positionssensoren zu ermöglichen. Der spezialisierte Typ *CollisionSensorIAPType* erbt die Variable *Condition_Fulfilled* und besitzt zwei Variablen (*CenterPoint* und *Radius*), um den CenterPoint und den Radius eines Kollisionssensors festzulegen.

Der spezialisierte Typ *PositionSensorIAPType* erbt die Variable *Condition_Fulfilled* und besitzt zwei Variablen (*Position* und Variance), um die Position im Raum für einen Sensor sowie die erlaubte Abweichung für die Erkennung der Positionserreichung festzulegen.

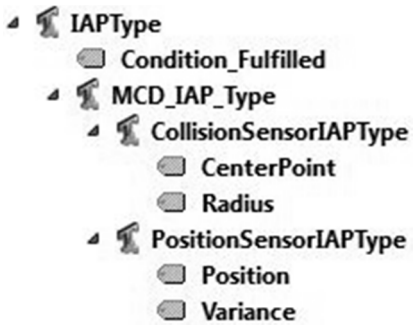


Abbildung 64: Darstellung des Objekttyps "IAPType"

Definition eines Objekttyps zur Festlegung der kinematischen Abhängigkeiten mittels spezialisierter Gelenktypen

Der Objekttyp *JointType* generalisiert verschiedene Gelenktypen, die im MCD zur Erstellung eines kinematisierten Modells verwendet werden.

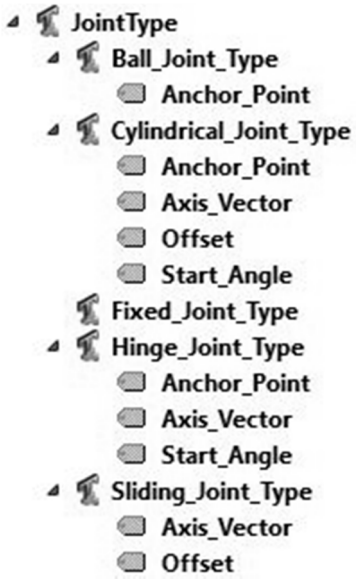


Abbildung 65: Darstellung des Objekttyps "JointType"

Folgende Gelenktypen spezialisieren sich vom *JointType*:

- Ball_Joint_Type (Kugelgelenk): Für die Festlegung des Kugelgelenks wird ein Drehpunkt (Anchor_Point) benötigt, der in diesem spezialisierten Gelenktyp deklariert ist.
- Cylindrical_Joint_Type (Zylindrisches Gelenk): Für die Festlegung des zylindrischen Gelenks werden ein Referenzpunkt (Anchor_Point), ein Achsvektor (Axis_Vector), ein Offset ausgehend vom Ausgangspunkt entlang des Achsvektors sowie ein initialer Drehwinkel des Zylinders benötigt, die in diesem spezialisierten Gelenktyp deklariert sind.
- Fixed_Joint_Type (Feste Verbindung): Feste Verbindung zur Fixierung von Körpern im 3D-Koordinatensystem des MCD

- Hinge_Joint_Type (Scharniergelenk): Das Scharniergelenk wird durch einen Ankerpunkt, einen Achsvektor ausgehend vom Ursprung des lokalen Koordinatensystems der Baugruppe und einem Startwinkel definiert. Das Scharniergelenk dreht um den angegebenen Achsvektor ausgehend vom definierten Ankerpunkt.
- Sliding_Joint_Type (Schiebegelenk): Das Schiebegelenk wird durch einen Richtungsvektor (Axis_Vector) und einen Offset definiert. Der Offset gibt die Abweichung zur initialen Konfiguration an.

Definition eines Objekttyps zur Abbildung der Operationen von Gantt-Diagrammen

Der Typ *„OperationType“* definiert die Abbildung von Operationen des MCDs in das OPC UA Informationsmodell. Diesem Typ sind die Variablen *endposition, invert_movement, speed* und *startposition* zugeordnet. Dabei gibt *start-* bzw. *endposition* die Start- bzw. Endposition einer Bewegungsoperation an. Mittels invert_movement kann die Bewegungsrichtung umgekehrt werden, falls dies erforderlich sein sollte. Durch die Variable *speed* kann die Geschwindigkeit, durch die Variable *duration* die Dauer der Bewegungsoperation im MCD eingestellt werden.

Abbildung 66: Darstellung des Objekttyps “OperationType“

Definition eines Objekttyps zur Abstrahierung logischer Verknüpfungselemente

Der *LogicalConditionType* dient zur Typisierung der logischen Verknüpfungselemente *AND*, *OR* und *NOT*.

Abbildung 67: Darstellung des Objekttyps “LogicalConditionType“

Typisierung von CAD- bzw. mechatronischen Komponenten im OPC UA Informationsmodell

Für die Abbildung der mechanischen Baugruppenstruktur aus CAD-Systemen im OPC UA Informationsmodell ist eine spezielle Typisierung sinnvoll. Wie in Tabelle 2 (a-d) dargestellt, existieren vier verschiedene Domänentypen, die als Typinstanzen vom Objekttyp MechatronicComponenttype (siehe e)) verwendet werden.

Tabelle 2: Überblick über die verschiedenen Domänentypen (a-d) des MechatronicComponentType (e)

<table>
<tr>
<td>a)</td>
<td>AutomationDomainType
Actuators
Sensors</td>
<td rowspan="4">e)</td>
<td rowspan="4">CADComponentType
MechatronicComponentType
AutomationDomain
Actuators
Sensors
ElectricalDomain
EPlan
Requirements
LogicalDomain
Functions
IAPs
MCDOperations
Operations
MechanicalDomain
Bodies
Components
Couplers
Joints
Transformation</td>
</tr>
<tr>
<td>b)</td>
<td>ElectricalDomainType
EPlan
Requirements</td>
</tr>
<tr>
<td>c)</td>
<td>LogicalDomainType
Functions
IAPs
MCDOperations
Operations</td>
</tr>
<tr>
<td>d)</td>
<td>MechanicalDomainType
Bodies
Components
Couplers
Joints</td>
</tr>
</table>

Dieser ist eine Spezialisierung des Objekttyps CADComponentType. Dabei dient der Objekttyp CADComponentType zur Abbildung von CAD-Bauteilen im Informationsmodell, der Objekttyp MechatronicComponentType zur Abbildung von mechatronischen Baugruppen.

4.3.2 Spezialisierte Referenzen des OPC UA Informationsmodells

Die Art der Abhängigkeit zwischen zwei Entitäten im Informationsmodell wird mittels Referenzen ausgedrückt. Dabei wird zwischen zwei Ausprägungen der Referenzen unterschieden:

- Hierarchische Referenzen: Hierarchische Referenzen werden als Baustruktur dargestellt und werden zur Strukturierung von übergeordneten und untergeordneten Entitäten verwendet.
- Nicht-hierarchische Referenzen: Nicht-hierarchische Referenzen werden verwendet, um eine bestimmte Art der Relation zwischen zwei Entitäten auszudrücken, wobei hier keine Rangfolge ausgedrückt werden soll.

Hierarchische Abhängigkeiten können mit Hilfe von benutzerdefinierten hierarchischen Referenzen dargestellt werden

Hierarchische Referenzen werden verwendet, um Baumstrukturen aufzubauen. Der Ausgangsknoten einer hierarchischen Referenz ist dabei der in einer hierarchischen Ordnungsstruktur übergeordnete Knoten, der Zielknoten entspricht dem hierarchisch

untergeordneten Knoten. Die verschiedenen typisierten, hierarchischen Referenzen sind nachfolgend aufgeführt.

Tabelle 3: Tabelle der zur Verfügung stehenden hierarchischen Referenzen

a)	HasMechanicalDomain	d)	HasAutomationDomain
b)	HasLogicalDomain	e)	HasCondition
c)	HasElectricalDomain	f)	HasMechanicalComponent

Abhängigkeiten ohne hierarchische Beziehung können mit Hilfe von benutzerdefinierten nicht-hierarchischen Referenzen abgebildet werden

Nicht-hierarchische Referenzen werden verwendet, um einem Knoten mittels Referenzen weitere semantische Informationen anzuhängen, ohne dabei jedoch eine hierarchische Ordnungsstruktur zu verwenden. Dadurch lässt sich der Kontext eines Knotens erweitern, indem typisierte Referenzen zwischen Ursprungsknoten und Zielentitäten erstellt werden.

Tabelle 4: Tabelle der zur Verfügung stehenden nicht-hierarchischen Referenzen

a)	HasMechatronicComponent	m)	HasBase
b)	HasJoint	n)	HasAttachment
c)	HasJointBase	o)	HasControl
d)	HasJointAttachement	p)	HasCouplerProfile
e)	HasCoupler	q)	HasState
f)	HasCouplerBase	r)	HasIAP
g)	HasCouplerAttachement	s)	HasStartConditionIAP
h)	HasRigidBody	t)	HasCallbackIAP
j)	HasCollisionBody	u)	HasNextState
k)	HasMCDOperation	v)	HasFunction

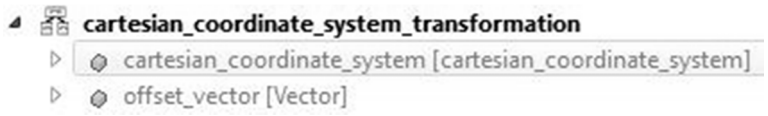
So kann zum Beispiel die Instanz eines *Sliding_Joint_Type* durch die Referenz *HasJointBase* mit der Gelenkbasis, durch die Referenz *HasJointAttachment* mit dem Gelenkanhang verbunden werden. Die verschiedenen typisierten, nicht-hierarchischen Referenzen sind in Tabelle 4 aufgeführt.

4.3.3 Spezialisierte Datentypen des OPC UA Informationsmodells

Im OPC UA Typsystem können spezialisierte Datentypen definiert werden, um komplexe Daten zu beschreiben. Dies wird nachfolgend anhand von Datentypen zur Abbildung von Koordinatensystemen wie zum Beispiel dem absoluten Koordinatensystem (CSYS)

von NX als auch den Arbeitskoordinatensystemen (WSYS) der einzelnen Bauteile oder Baugruppen dargestellt.

Tabelle 5: Darstellung spezialisierter Datentypen des OPC UA Informationsmodells

❶ 	❷ 	❸ 
❹ 		

Um die Koordinate einer Position (siehe Tabelle 5; ❶) im Raum beschreiben zu können, existiert der Datentyp *Position*. Dieser setzt sich aus drei Properties (*X*, *Y*, *Z*) des Datentyps *Double* zusammen. Der Datentyp Vector (siehe Tabelle 5; ❷) setzt sich aus einem Property *tie_point* des Datentyps *Position* (siehe Tabelle 5; ❶) zusammen. Der *tie_point* definiert einen Scheitelpunkt eines Vektors, der sich zwischen dem Ursprung und dem *tie_point* aufspannt. Der Datentyp *cartesian_coordinate_system* (siehe Tabelle 5; ❸) setzt sich aus den drei Properties *XAxis*, *YAxis* und *ZAxis*, jeweils vom Datentyp *Vector* (siehe Tabelle 5; ❷), zusammen. Durch diese drei Vektoren wird das Koordinatensystem aufgespannt. Der Datentyp *cartesian_coordinate_system_-transformation* (siehe Tabelle 5; ❹) setzt sich aus dem Property *cartesian_coordinate_-system* mit dem Datentyp *cartesian_coordinate_system* (siehe Tabelle 5; ❸) und dem Property *offset_vector* mit dem Datentyp *Vector* (siehe Tabelle 5; ❷) zusammen. Mit Hilfe des Datentyps *cartesian_coordinate_system_transformation* kann die Transformation zwischen zwei verschiedenen Koordinatensystemen beschrieben werden, wie es zum Beispiel bei der Erstellung von Baugruppen aus Bauteilen im CAD-Bereich erforderlich ist. So besitzt jedes Bauteil ein eigenes lokales Koordinatensystem. In einer Baugruppe werden Bauteile in dem Koordinatensystem der Baugruppe neu orientiert und positioniert, was durch Angabe des Koordinatensystems und des Offsetvektors beschrieben werden kann.

Der in Abbildung 68 dargestellte Datentyp *OperationParameter* dient zur Bereitstellung eines generischen Objekts, das beliebige Daten beinhalten kann. Dieses Objekt setzt sich aus einem Property *Namen* mit dem Datentyp String, einem Enumerator *ParamValue_Enum* mit dem Datentyp *Operation_Parameter_DataType* sowie den Properties *ParamValue_0* mit dem Datentyp Double*, ParamValue_1* mit dem Datentyp Float*, ParamValue_2* mit dem Datentyp UInt32 und *ParamValue_3* mit dem Datentyp String zusammen.

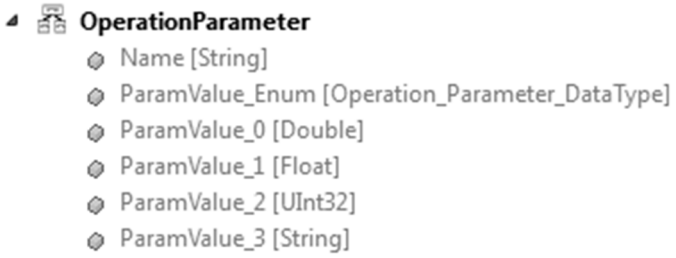


Abbildung 68: Darstellung des Datentyps "OperationParameter"

Bei Verwendung des Datentyps *OperationParameter* gibt der Enumerator *ParamValue_Enum* an, welches der Properties ParamValue_X gültig ist und den eigentlichen Datenwert beinhaltet, sodass beliebige Datenwerte definiert und dynamisch mittels Switch-Case-Anweisung verarbeitet werden können.

4.3.4 Abbildung der mechanischen Struktur und des Ablaufverhaltens in einem OPC UA Informationsmodell

Das in Kapitel 3.6 eingeführte Beispiel einer Handhabungseinheit besitzt eine Baugruppenstrukturierung, die bereits in Abbildung 55 dargestellt ist. Mit Hilfe der in Kapitel 4.3 definierten Objekt- und Referenztypen kann diese Strukturierung in ein OPC UA Informationsmodell übertragen werden.

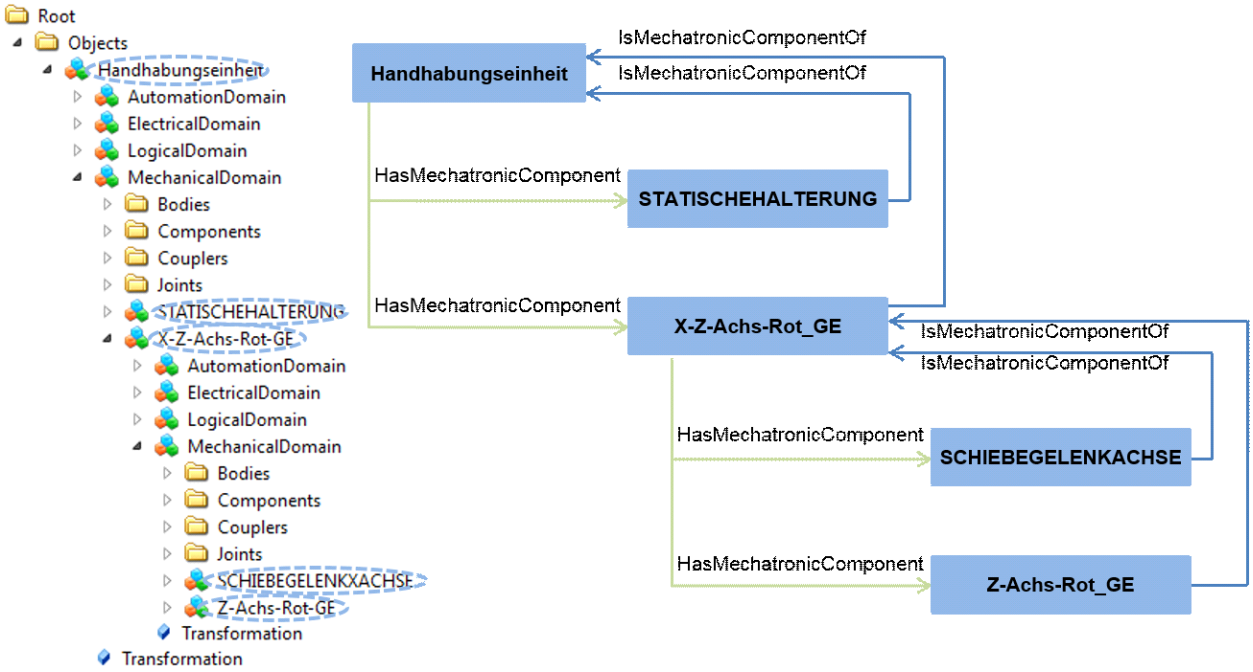


Abbildung 69: Darstellung einer CAD-Baugruppenstruktur, die in ein OPC UA Informationsmodell exportiert wurde

In Abbildung 69 ist links die resultierende Baumstruktur im Instanzraum des OPC UA Informationsmodells dargestellt. Die CAD-Baugruppen „Handhabungseinheit“, „STATISCHEHALTERUNG“, „X-Z-Achs-Rot-GE“, „SCHIEBEGELENKXACHSE“ sowie „Z-Achs-Rot-GE“ sind im Informationsmodell durch Instanzen des Objekttyps *MechatronicComponentType* (siehe Tabelle 2, Seite 99) repräsentiert. Auf der rechten Seite ist die hierarchische Referenzierung zwischen den *MechatronicComponentType-Instanzen* unter Verwendung der hierarchischen Referenz *HasMechatronicComponent* (grün) abgebildet. Diese kann auch gegenläufig der Referenzrichtung verwendet werden, was durch inverse Namen der Referenzen (*IsMechatronicComponentOf*) unterstützt wird. Diese hierarchische Struktur im Informationsmodell analog zur Baugruppenstrukturierung im CAD-System dient als Gerüst, mit dem weitere Informationen unterschiedlicher Domänen verknüpft werden können. Ein Beispiel hierfür ist die Abbildung des Ablaufverhaltens des MCD in das OPC UA Informationsmodell (siehe Abbildung 70).

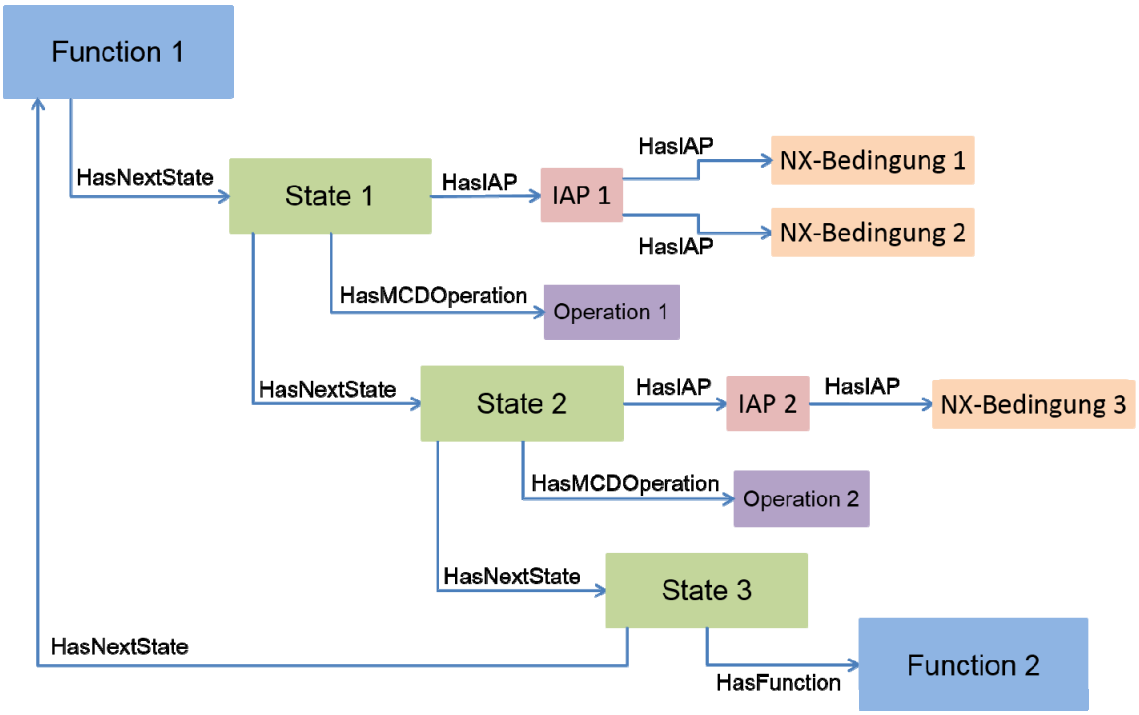


Abbildung 70: Darstellung der Referenzhierarchie zur Abbildung des Gantt-Charts des Mechatronics Concept Designers im OPC UA Informationsmodell

Function 1 ist eine komponentenspezifische Funktionalität und wird in die Unterstrukturen der LogicalDomain einer MechatronicComponentType-Instanz eingeordnet. Function 1 referenziert mit Hilfe der HasNextState-Referenz auf State 1. Mit Hilfe von State-Objekten kann eine Zustandsabfolge erstellt werden, die auch mit Hilfe von Divergenz- und Konvergenzobjekten eine Beschreibung von optionalen bzw. nebenläufigen Ablaufsequenzen ermöglicht. State-Objekte können Interaktionspunkte, Operationen und Funktionen referenzieren. Die Interaktionspunkte aggregieren die Bedingungen, die für

die Ausführung einer Operation gegeben sein müssen. Durch Einbindung weiterer Funktionen (z.B. Function 2) ist eine Verteilung des Gantt-Charts über mehrere Komponenten möglich. Dies unterstützt somit Ansätze des funktionalen Komponententauschs, da die Aufrufstruktur erhalten bleibt und sich lediglich die konkrete Ausprägung einer Funktion ändert.

4.4 Referenzimplementierung einer Ablaufumgebung zur Ausführung von Ablaufsequenzen

Für die Referenzimplementierung der Ablaufumgebung auf Steuerungsebene wurde eine bereits existierende Hardware-Architektur verwendet, die auf einem Sitara Cortex-A8-Prozessor (ARM-Architektur) basiert. Als Betriebssystem wird auf dieser Hardwareplattform eine echtzeitfähige Linux-Distribution eingesetzt. Darauf aufsetzend können verschiedene Applikationen ausgeführt werden.

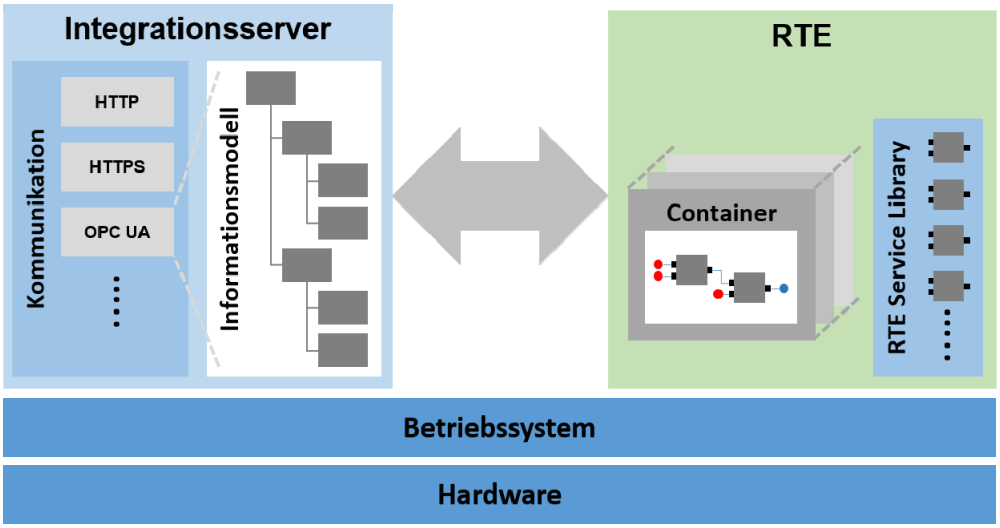


Abbildung 71: Strukturelle Darstellung des Aufbaus der Ablaufumgebung

Wie in Abbildung 71 dargestellt, besteht die Applikationsebene aus den Elementen „Integrationsserver (IS)“ und „RunTimeEngine (RTE)“. Die bidirektionale Interaktion erfolgt über Funktionsschnittstellen.

Das OPC UA Informationsmodell wird mit Hilfe eines Integrationsservers (IS) zur Verfügung gestellt

Der verwendete Integrationsserver integriert verschiedene Kommunikationsprotokolle wie zum Beispiel *http, https, Modbus TCP* sowie *OPC UA* und ist in der Lage, weitere lokale Applikationen anzubinden. Hierfür werden die über beliebige Protokolle von extern eingehenden Daten in ein internes Datenformat überführt, welches dann für eine System-interne Informationsverteilung z. B. über IPC-Mechanismen genutzt wird.

Wie bereits erwähnt, integriert der IS unter anderem einen OPC UA Server. Dies ist insbesondere deshalb von Vorteil, da OPC UA ebenfalls zur Abbildung der Informationen aus dem Engineering (vgl. Kapitel 4.2) verwendet wurde. Dadurch wird derselbe Kommunikations- und Beschreibungsstandard sowohl im Engineering als auch im Umfeld der Runtime eingesetzt. Somit ist eine direkte Verwendung der OPC UA Abbildung aus dem Engineering seitens der Ausführungsumgebung auf Steuerungsebene möglich. Der IS stellt ein Informationsmodell online zur Verfügung, bei dem alle Informationen sowohl clientseitig als auch von der RTE mittels IPC-Mechanismen zugreifbar sind.

Die Ausführung der Steuerungslogik wird durch eine RunTimeEngine (RTE) realisiert

Die RunTimeEngine realisiert die dynamische Ausführung von Steuerungslogik zur Laufzeit. Dies umfasst die Fähigkeit, dass die Steuerungslogik flexibel erstellt, Wertebe-rechnungen und Konvertierungen durchgeführt sowie logische Verknüpfungen (Boole-schen Algebra) definiert werden können. Die RTE ist somit in der Lage, als Ablaufum-gebung für das Verhalten einer oder mehrerer mechatronischer Komponenten eingesetzt zu werden.

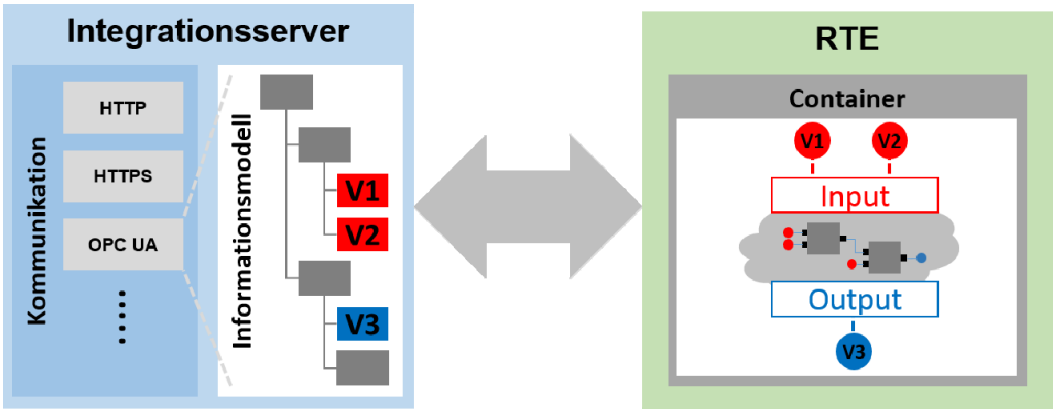


Abbildung 72: Schematische Verknüpfung von Ein- und Ausgangsvariablen mit einer Berechnungslogik in der RunTimeEngine

Die RTE führt nur die Berechnung einer definierten Steuerungslogik durch, speichert jedoch keine Werte, sondern verwendet Variablen des Informationsmodells des Integra-tionsservers als Ein- bzw. Ausgangswerte (vgl. Abbildung 72; Eingangsvariablen „*Var1*“ und „*Var2*“ sowie Ausgangsvariable „*Var3*“).

4.4.1 Entwicklung von Servicebausteinen für eine Servicebibliothek der RunTimeEngine

Das Grundkonzept der RunTimeEngine beruht auf der Verwendung von Servicebau-steinen, die mit Hilfe einer Servicebibliothek dem Anwender zur Verfügung gestellt wer-

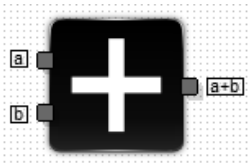
den. Der Anwender nimmt die Projektierung bzw. Programmierung des Zielsystems durch Verknüpfung von Services aus der Servicebibliothek vor. Diese Servicebibliothek kann erweitert werden, indem individuell implementierte Services hinzugefügt werden

Abbildung 73: Darstellung eines Servicebausteins der RTE

Wie in Abbildung 73 gezeigt, kann jeder Servicebaustein eine spezifische Darstellung besitzen und wird durch eine C++-Klasse implementiert. Jeder Service liegt in kompilierter Form als separates „*shared object*“ vor und besitzt jeweils die gleiche Funktionsschnittstelle. Dies ist wichtig, da jeder Service dadurch funktional auf die gleiche Art angesprochen bzw. aufgerufen werden kann.

Einige elementare arithmetische Servicebausteine sind in Tabelle 6 veranschaulicht. Diese können zur Werteberechnung eingesetzt werden.

Tabelle 6: Darstellung der arithmetischen Servicebausteine „Addition“, „Subtraktion“ und „Multiplikation“

Service	Symbol	Operation
Addition		Ausgang = a+b
Subtraktion		Ausgang = a-b
Multiplikation		Ausgang = a*b

Neben den arithmetischen Servicebausteinen existieren noch weitere Typen von Servicebausteinen wie zum Beispiel zur Konvertierung von Datentypen und zur Realisierung von Logikgattern. Das verwendete Architekturkonzept der Ausführungsumgebung der Servicebausteine erlaubt eine modulare Erweiterung der Servicebibliothek um benutzerdefinierte Services. Darüber hinaus wurde es um die Fähigkeit der Ausführung von zustandsbasierten Ablaufsequenzen erweitert. Hierfür war die Anpassung des Abarbeitungskonzepts der Container sowie die Erstellung neuer Sequenzservices erforderlich, worauf im folgenden Kapitel näher eingegangen wird.

4.4.2 Entwicklung spezialisierter Servicebausteine der RunTimeEngine zur Realisierung von Ablaufsequenzen

Um zustandsbasierte Ablaufsequenzen im Rahmen der RunTimeEngine ausführen zu können, wurden zusätzliche Services (SequenceServices) implementiert und das Aufrufverhalten für diese SequenceServices angepasst.

Die zentralen Servicebausteine, die eine Ablaufsequenz bei der hier vorgestellten Realisierung immer kennzeichnen, sind der INIT- und der END-Service (siehe Abbildung 74, links). Der INIT-Service dient als zustandsloser Start-Knoten eines jeden Ablaufsequenzgraphen. Des Weiteren implementiert er die Ausführungslogik der Ablaufsequenz und ist als eine Art Handler-Objekt für die Ausführung von Zustandsübergängen sowie für den funktionalen Aufruf der Berechnungslogik einzelner Operationen zuständig. Dies stellt einen wesentlichen Unterschied zu der bisher vorgestellten Implementierung der RunTimeEngine dar, bei der die Logik des Containers für den funktionalen Aufruf der Berechnungslogik einzelner Servicebausteine zuständig ist. Bei Verwendung von SequenceServices ruft der Container ausschließlich nur noch den INIT-Service auf und dieser übernimmt die weitere Ausführung der Aufrufsequenz.

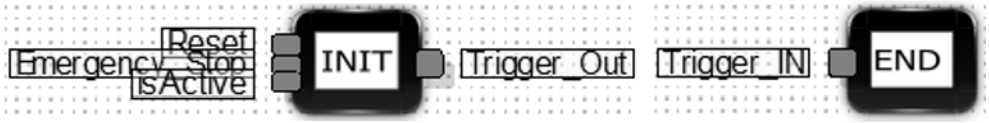


Abbildung 74: Darstellung der Sequenceservices „INIT“ (links) und „END“ (rechts)

Das Gegenstück zum INIT-Service ist der END-Service (siehe Abbildung 74, rechts), der als Ringschluss zum INIT-Service zu verstehen ist. Die projektierte Verschaltungslogik zwischen INIT- und END-Service wird zyklisch wiederholt und ist vergleichbar mit einem Steuerungsprogramm einer klassischen SPS. Weitere zentrale Servicebausteine zur Erstellung von Verschaltungsnetzwerken sind die Transitions- und Operationsservices. Der Transitionsservice (siehe Abbildung 75, links) erlaubt einen Zustandsübergang einer vorausgehenden Operation, die mit seinem Eingang „Transition_In“ verknüpft ist, zu einer nachfolgenden Operation, die mit dem „Transition_Out“-Ausgang

verknüpft ist. Dabei hängt die Transitionsfähigkeit von der Transitionsbedingung ab, die mit dem Eingang „Transition_Condition“ verbunden ist.

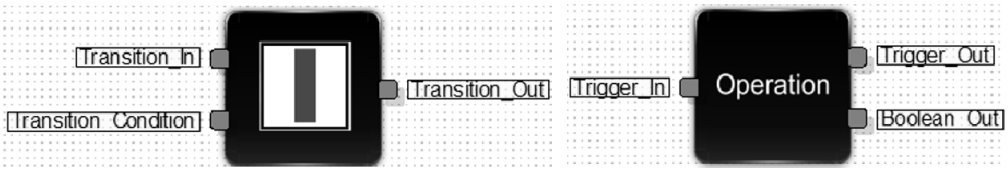


Abbildung 75: Darstellung der Sequenceservices „Transition“ (links) und „Operation“ (rechts)

Der Operationsservice entspricht einem stationären Zustand der Ablaufsequenz, der so lange besteht, bis eine Zustandsänderung aufgrund einer möglichen Transition erfolgt. Solange ein Operationsservice aktiv ist, wird der Ausgang „Boolean_Out“ auf *true* gesetzt.

Das Aufteilen in bedingte bzw. nebenläufige Teilsequenzen lässt sich durch Verwendung von Divergenzservicebausteine realisieren. Wie in Abbildung 76 dargestellt, steht hierfür ein „OR-divergence“-Service (links) zur Aufteilung in eine beliebige Anzahl an bedingten Teilsequenzen bzw. ein „AND-divergence“-Service (rechts) zur Aufteilung in nebenläufige Teilsequenzen zur Verfügung.

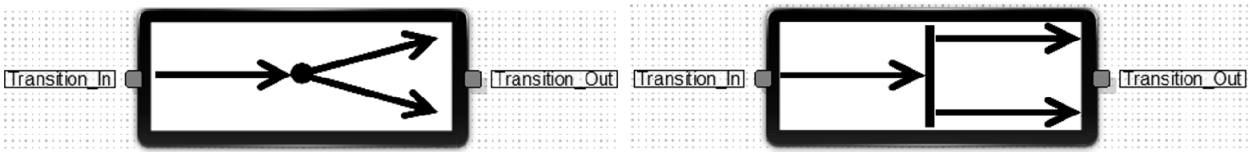


Abbildung 76: Darstellung der Sequenceservices „OR-divergence“ (links) und „AND-divergence“ (rechts)

Um verschiedene Stränge von Teilsequenzen wieder zusammenführen zu können, gibt es analog zur den Divergenzservices auch Konvergenzservices (siehe Abbildung 77), die bedingte Teilsequenzstränge („OR-convergence“) bzw. nebenläufige Teilsequenzstränge („AND-convergence“) wieder zusammenführen. Mit Hilfe des „SequenceStop“-Services kann die Ausführung eines Teilsequenzstrangs auch beendet werden.

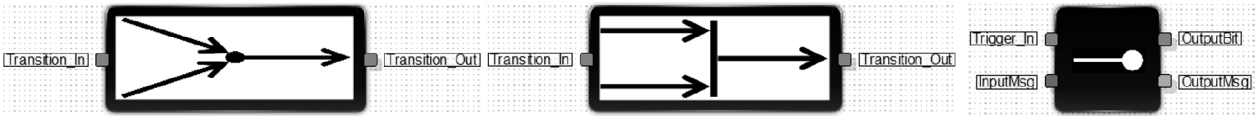


Abbildung 77: Darstellung der Sequenceservices „OR-convergence“ (links), „AND-convergence“ (Mitte) und „SequenceStop“ (rechts)

Mit Hilfe der Divergenz- und Konvergenzservices können komplexe Verschaltungsnetzwerke erstellt werden. Für die Ausführung der Ablaufsequenz ist deshalb ein Algorithmus erforderlich, der ausgehend von einem aktiven Operationsservice ermittelt, ob und zu welchen nachfolgenden Operationsservices eine Transition durchgeführt werden kann.

Ein einfaches Beispiel ist in Abbildung 78 dargestellt. Soll zum Beispiel Operation ① ermitteln, ob eine Transition möglich ist, hängt dies davon ab, ob Operation ② transitionsbereit ist, da sonst ein gemeinsamer Übergang mit Hilfe des „AND-convergence"-Service ③ nicht möglich ist. Dieser überprüft nach initialem Aufruf durch Operationsservice ① rückwärtsgerichtet rekursiv, ob alle vorhergehenden Operationsservices transitionsbereit sind. Fällt diese Abfrage positiv aus, wird der Funktionsaufruf zur Ermittlung transitionsfähiger Operationsservices an den nachfolgenden „OR-divergence"-Service ④ weitergegeben. Dieser prüft vorwärtsgerichtet rekursiv, ob eine der nachfolgenden Transitionen zu den Operationsservices ⑤ bzw. ⑥ übergangsfähig ist und liefert diesen im positiven Fall als Rückgabewert an den initial aufrufenden Operationsservice ① zurück. Dieser kann dann die Transition durchführen, indem er sich deaktiviert und den transitionsfähigen Operationsservice ⑤ bzw. ⑥ aktiviert.

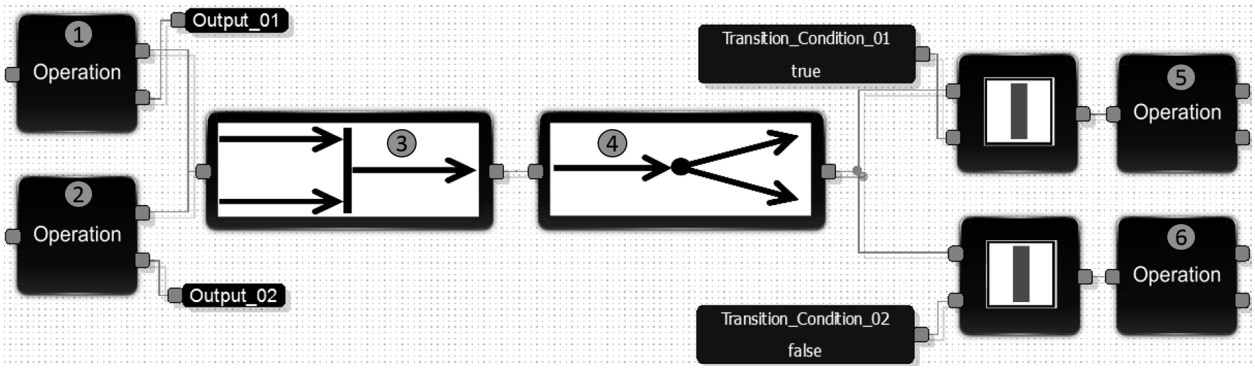


Abbildung 78: Darstellung eines Verschaltungsnetzwerks zur Erläuterung der rekursiven Überprüfung der aktiven Operationen

Der in den SequenceServices implementierte Algorithmus zur Überprüfung der Transitionsfähigkeit ist für beliebige Verschaltungsnetzwerke anwendbar, auch wenn diese zum Beispiel nicht Grafcet-konform erstellt wurden. Mit Hilfe der rekursiven Überprüfung der Transitionsfähigkeit resultiert eine deutlich verbesserte Performance im Vergleich zu vorherigen, wertsetzungsgetriebenen Ansätzen.

4.4.3 Verwendung von RunTimeEngine Containern zur Strukturierung der Steuerungslogik

Die RunTimeEngine verwendet Container als Strukturierungsmittel für Verschaltungsnetzwerke von Servicebausteinen. Ein Container kann dabei beliebig viele untereinan-

der verknüpfte Servicebausteine enthalten und ist für deren Ausführung verantwortlich. Dabei kann zwischen den folgenden zwei Container-Typen unterschieden werden:

- Container zur ereignisdiskreten Ausführung der enthaltenen Services bei Wertänderung der Eingangsvariablen bzw.
- Container mit zyklischem Aufruf zur Ausführung von Ablaufsequenzen.

Container zur ereignisdiskreten Ausführung von Services werden verwendet, um Formeln zu berechnen, deren Ausgangswerte von Wertänderungen der Eingangsvariablen abhängen. Zur Veranschaulichung ist in Abbildung 79 ein einfaches Beispiel dargestellt.

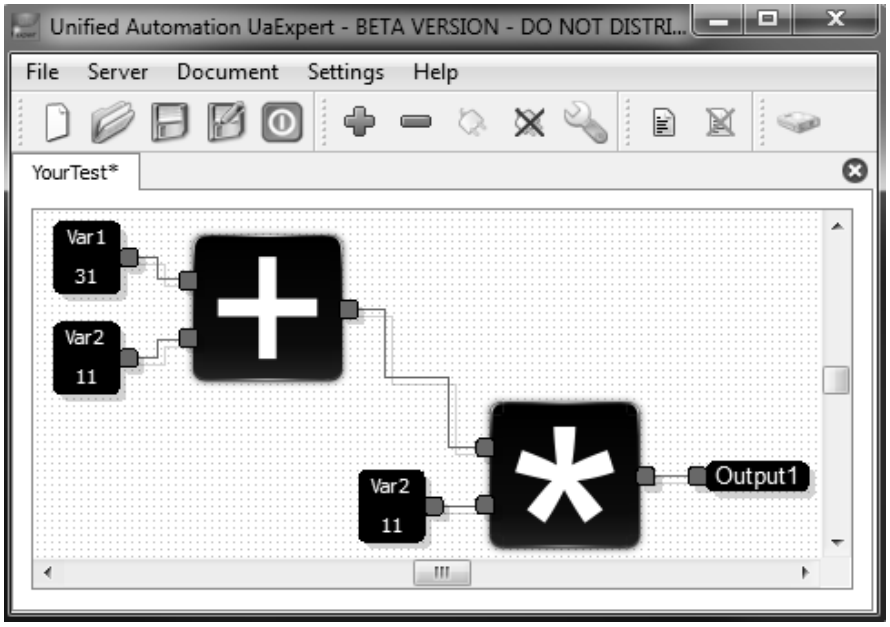


*Abbildung 79: Darstellung eines ereignisdiskreten Containers mit der Berechnungsformel Output1=(Var1+Var2)*Var2*

Im Container „*YourTest*“ befinden sich zwei Instanzen von Servicebausteinen. Bei deren Instanziierung werden zuerst die Eingangs- und Ausgangsvariablen initialisiert (vgl. Abbildung 79; Eingangsvariablen „Var1“ und „Var2“ sowie Ausgangsvariable „Var3“), sodass diese gelesen bzw. beschrieben werden können. Danach kann der instanziierte Service (vgl. Abbildung 79; arithmetische Addition bzw. Multiplikation) verwendet werden. Dies erfolgt durch den Container mittels Aufruf der Funktion compute() eines Servicebausteins, wodurch dessen eigentliche Berechnungslogik unter Verwendung der Eingangswerte (vgl. Abbildung 79; „Var1“ und „Var2“) ausgeführt wird. Bei Betrachtung der Verschaltung der beiden Servicebausteine wird klar, dass die Ausführungsreihenfolge der Services für eine korrekte Berechnung eine entscheidende Rolle spielt, da in diesem Beispiel die Berechnung des Additionsservice abgeschlossen sein muss, bevor dessen Ergebnis als Eingangswert für den Multiplikationsservice genutzt werden kann. Die korrekte Aufrufreihenfolge stellt der Container durch einen Sortieralgorithmus si-

cher, der alle Services eines Containers analysiert und in richtiger Reihenfolge aufruft. Abschließend muss noch das Berechnungsergebnis auf die Ausgangsvariable „*Var3*“ (vgl. Abbildung 79) geschrieben werden.

Container mit zyklischem Aufruf zur Ausführung von Ablaufsequenzen enthalten SequenceServices, die in Kapitel 4.4.2 beschrieben wurden. Der spezialisierte Container wurde so angepasst, dass lediglich „INIT“-Servicebausteine zyklisch aufgerufen werden und diese die weitere Ausführung der Ablaufsequenzen steuern.

4.4.4 Verwendung von Verschaltungsmodellen der RunTimeEngine zur funktionsorientierten Programmierung der Steuerungslogik

Mithilfe des vorgestellten Konzepts sowie dessen Umsetzung kann während der Planungsphase das Verhalten automatisierter Produktionssysteme funktional beschrieben werden (vgl. Kapitel 4.2.3). Dies bietet den Vorteil, dass aus einer bestimmten Sequenz von elementaren Funktionen einzelner Gewerke die Gesamtfunktionalität mechatronischer Einheiten zusammengesetzt werden kann. Aus funktionaler Sicht steht somit die Aufrufreihenfolge der einzelnen Komponenten fest, um die Gesamtfunktionalität im Sinne des Prozesses zu realisieren. Bei der in Kapitel 3.6 dargestellten Handhabungseinheit ist es zum Beispiel erforderlich, dass sich die X-Achse auf einer bestimmten Position befindet, bevor die Z-Achse nach unten fährt, damit der angehängte Endeffektor ein Objekt greifen kann. In Engineering-Werkzeugen wird zur Abbildung des funktionalen Verhaltens einer Maschine oder Anlage das Bewegungsverhalten der Mechanik unter Berücksichtigung der vorhandenen Kinematik beschrieben. Dabei wird oftmals von den zugrundeliegenden Komponenten der Automatisierungstechnik (Sensorik bzw. Aktuatorik) abstrahiert. Die X-Achse der bereits angesprochenen Handhabungseinheit wird pneumatisch betrieben. Der eigentliche Aktuator, der die translatorische Bewegung der X-Achse auslöst, ist ein Ventil, das zwei pneumatische Ausgänge ansprechen kann.

Für die Entwicklung der Steuerungslogik stellt sich somit die Frage, wie eine funktionsorientierte Sicht im Engineering auf Schaltbefehle für die Ausgänge der E/A-Leiste überführt werden können. Ziel ist es, die verwendete Aktuatorik so anzusprechen, dass das im Engineering beschriebene funktionale Verhalten der Mechanik resultiert. Hierfür muss seitens der Steuerungslösung eine Funktionsschnittstelle bereitgestellt werden, die sowohl in Richtung Engineering abgebildet als auch auf Steuerungsebene zur Ausführung der Steuerungslösung verwendet werden kann.

Wie in den vorherigen Kapiteln dargestellt, ist die RunTimeEngine einerseits für eine diskrete Formelberechnung verschalteter Services geeignet, die durch Wertänderung von Eingangsvariablen ausgelöst wird. Andererseits kann durch die Verwendung der Servicebausteine zur Erstellung von Ablaufsequenzen ebenfalls ein sequentielles Ablaufverhalten realisiert werden.

In Abbildung 80 ist ein Konzept zur Überführung eines Funktionsaufrufs auf eine Wertsetzung von Ausgängen der E/A-Leiste sowie die Verknüpfung von Eingängen der E/A-Leiste auf Callbackvariablen der funktionalen Ebene dargestellt. Für eine graphisch verdichtete Darstellung wurden die Transitionsservicebausteine weggelassen und die Transitionsbedingungen direkt links an die Operationsservicebausteine angetragen. Die tatsächliche Darstellung der Referenzimplementierung kann ausschnittsweise Abbildung 82 entnommen werden. Das Konzept differenziert zwischen den drei Bereichen „Function“, „Logic“ sowie „Physics“ und wird in obiger Abbildung mittels der Container „F_Container_1“ und „P_Container_1“ auf der RTE umgesetzt.

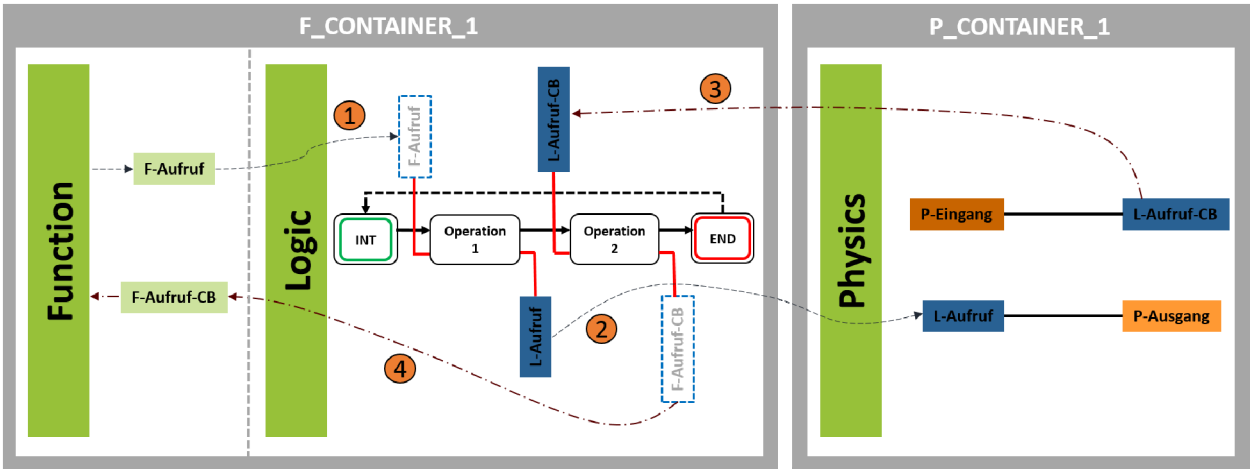


Abbildung 80: Darstellung einer Schnittstelle zum funktionalen Aufruf von Steuerungsprogrammen mit Callback

Im ersten Schritt werden im Engineering alle Funktionen und optional deren entsprechende Callbacks definiert. Diese haben eine typisierte Repräsentation als Variable im OPC UA Informationsmodell. Diese Variablen können im Bereich „Function“ direkt übernommen werden.

Im zweiten Schritt muss eine Funktion mit dem gewünschten Verhalten auf Steuerungsebene hinterlegt werden. So kann es zum Beispiel erforderlich sein, dass bestimmte Konfigurationsparameter (zum Beispiel Soft-Anlauf eines Motors) gesetzt werden müssen, bevor ein Motorstart erfolgen darf. Im Bereich „Logic“ kann dies für beliebige sequentielle und nebenläufige Abläufe umgesetzt werden.

Wie in Abbildung 80 dargestellt, erfolgt der funktionale Aufruf über die Variable „F-Aufruf“ im Bereich „Function“ im Container „F_Container_1“ (①). Diese Variable wird in ein sequentielles Verschaltungsnetzwerk als Eingangsbedingung für „Operation1“ verwendet. Bei Aufruf von „F-Aufruf“ aktiviert die sequentielle Operation „Operation1“ die Ausgangsvariable „L-Aufruf“, die in dem Container „P_Container_1“ als Eingangsvariable (②) verwendet wird. Bei Änderung der Variable „L-Aufruf“ wird der reale Ausgang „P-

Ausgang“ auf den Wert von „L-Aufruf“ gesetzt. Anschließend wird solange gewartet, bis eine Wertänderung am Eingang „P-Eingang“ anliegt. Diese setzt die Ausgangsvariable „L-Aufruf-CB“, die wiederum als Eingangsvariable von „Operation2“ (③) verwendet wird. „Operation2“ wiederum aktiviert die Variable „F-Aufruf-CB“, die auf funktionaler Ebene die erfolgreiche Ausführung des Prozesses signalisiert (④). Auf höheren funktionalen Ebenen ist ein direkter Zugriff auf die E/A-Leiste nicht gewünscht. Vielmehr sollen unterlagerte Funktionen verwendet bzw. aufgerufen werden.

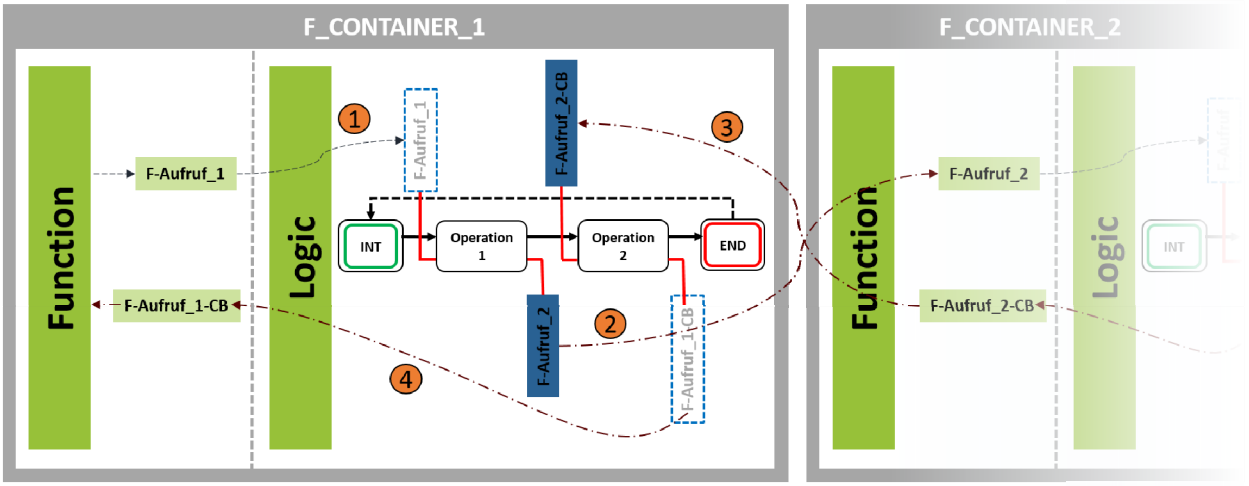


Abbildung 81: Darstellung einer funktionalen Aggregation unter Verwendung von verschalteten Services in der RTE

Abbildung 81 verdeutlicht, dass im Bereich „Logic“ nicht nur unterlagerte Container aus dem Bereich „Physics“ aufgerufen werden können (vgl. Abbildung 80), sondern ebenfalls Funktionen untergeordneter Komponenten. So lässt sich zum Beispiel eine funktionale Aggregation umsetzen, die bei Aufruf einer Funktion eine komplexe Ablaufsequenz aus beliebigen sequentiellen und/oder nebenläufigen Funktionsaufrufen realisiert.

4.5 Abbildung der Verhaltensbeschreibung aus dem OPC UA Informationsmodell auf Servicenetzwerke einer Ablaufumgebung

Die komponentenbasierte, funktionale Strukturierung im Mechatronics Concept Designer (siehe Kapitel 4.2.3) wurde mit Hilfe eines Typsystems bestehend aus Objekt-, Daten- und Referenztypen (siehe Kapitel 4.3) in ein OPC UA Informationsmodell ausgeleitet. Dieses Informationsmodell stellt das Bindeglied zwischen Projektierung im Engineering und der Ablaufumgebung des Steuerungsprogramms auf Steuerungsebene dar. Das verwendete Typsystem stellt dabei eine Metaebene zwischen Engineering und Steuerungsebene dar. An dieser Stelle soll deshalb die Abbildung der Verhaltensbeschreibung aus dem OPC UA Informationsmodell auf Servicenetzwerke der Ablaufumgebung (RTE) erläutert werden. Ausgehend von der Struktur- und Verhaltensbeschreibung in einem OPC UA Informationsmodell wie in Abbildung 69 und Abbildung 70

dargestellt, kann eine Überführung in die Laufzeitumgebung mit Hilfe der folgenden Schritte erfolgen.

- Für jede Komponente, die mindestens eine Funktion beinhaltet, wird ein F_Container (siehe Abbildung 80) erzeugt.
- Für jede Komponente, die mindestens eine Funktion beinhaltet und keine weiteren Komponenten aggregiert, wird ein P_Container (siehe Abbildung 80) erzeugt.
- Jede Funktion wird mit Hilfe einer Ablaufsequenz innerhalb des F_Containers der entsprechenden Komponente realisiert. Somit werden für jede Funktion ein INIT- und ein END-Servicebaustein benötigt. Die Eingangsvariablen des INIT-Servicebausteins werden mit den entsprechenden Variablen der Funktionsinstanz (siehe Abbildung 63) verbunden.
- Der Funktionsaufruf wird in OPC UA mittels Wertsetzung der Variablen trigger der Funktionsinstanz (siehe Abbildung 63) umgesetzt. Das Erreichen des gewünschten Zustands nach Auslösung eines Aktuators kann mit Hilfe der callback-Variablen wiedergegeben werden. Dies ist ein wesentlicher Unterschied zur Abbildung des Bewegungsablaufs im MCD mittels Gantt-Chart, da hier die Reihenfolge des Ablaufs der einzelnen Operationen durch sequentielle Verknüpfung der Operationen mit Hilfe von Linkern sichergestellt werden kann. Bei realen Umsetzungen gibt es drei Ausprägungen, wie eine sequenzielle Ausführung von mehreren Abläufen erreicht werden kann:
  1. Überwachung des IST-Zustands mittels Sensorik (z.B. Absolutwertgeber)
  2. Überprüfung des Zielzustands mittels Sensorik (z.B. induktiver Näherungssensor)
  3. Annahme der Zustandserreichung nach festgelegter Zeit ohne Sensorüberwachung (z.B. zeitgesteuerte Schaltvorgänge mittels Timer-Bausteinen)

  Für jeden State (siehe Abbildung 70) werden in abwechselnder Reihenfolge jeweils zwei Transitions- und zwei Operation-Servicebausteine beginnend mit einem Transitions-Servicebaustein generiert und miteinander verknüpft (siehe Abbildung 82). Die Transitionsbedingung des ersten Transitions-Servicebausteins (Transition_Condition) wird mit der entsprechenden trigger-Variablen (F1_trigger) der Funktionsinstanz verbunden. Der Boolean_Out-Output (siehe Abbildung 75, rechts) des zweiten Operation-Servicebausteins wird mit der entsprechenden callback-Variablen (F1_callback) der Funktionsinstanz verbunden.

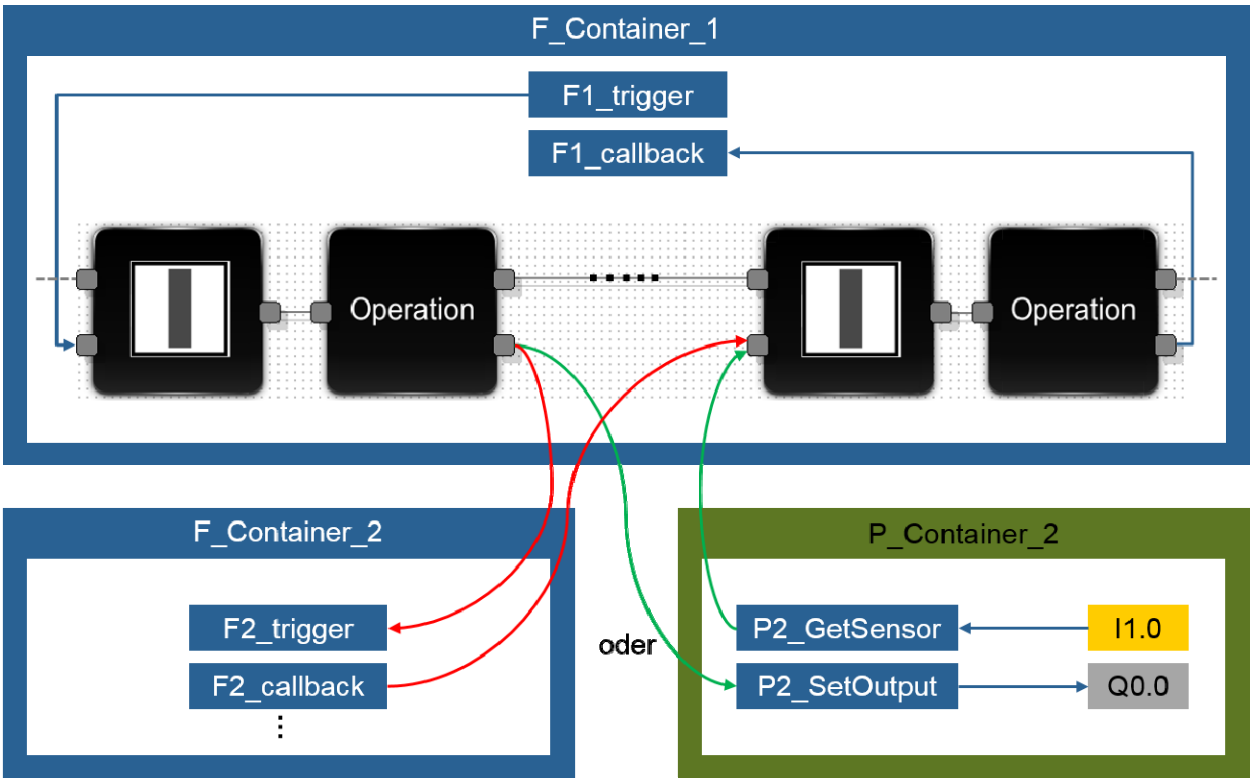


Abbildung 82: Abbildung der Ablaufsequenz des OPC UA Informationsmodells auf ein Functioncall-/Callback- bzw. Wertsetzungs-/Sensorfeedback-Schema

Wie bereits in Kapitel 4.4.4 dargestellt, kann ein Funktions-Container (F_Container_X, der die Funktionen sowie deren sequentielle Abläufe enthält, sowohl unterlagerte Funktions-Container als auch Physics-Container (P_Container_Y) zur Wertsetzung von Ausgängen bzw. Einbindung von Sensorsignalen aufrufen (siehe Abbildung 82, unten links und rechts).

Soll die Funktion *F1_trigger* eine Sequenz von mehreren unterlagerten Funktionen auslösen, würde das in Abbildung 82 dargestellte Schema bestehend aus zwei Transition-Servicebausteinen und zwei Operation-Servicebausteinen mehrfach angewendet werden. Der kombinierte Einsatz von Divergenz- und Konvergenz-Servicebausteinen ist natürlich ebenfalls möglich, wobei die Abbildung hier der Vorgabe des *State*-Graphs (siehe Abbildung 70) folgt.

4.6 Referenzimplementierung einer automatisierten Generierung einer webbasierten Visualisierung automatisierter Produktionssysteme

Während der Entwicklungs- und Planungsphase können automatisierte Produktionssysteme heute im Detail digital abgebildet werden. Wie in den vorausgegangenen Kapiteln erläutert, können Informationen zum Beispiel aus verschiedenen Domänen bzw. Engineering-Tools extrahiert und im Rahmen eines Informationsmodells zusammengeführt

werden. Dabei stellt sich die Frage, welcher Nutzen aus diesen Informationen auf Steuerungsebene im Feld resultieren kann. Ein zu betrachtender Aspekt in diesem Kontext ist zum Beispiel eine generische, webbasierte Visualisierung der Maschine bzw. des betrachteten Produktionssystems. Eine webbasierte Realisierung bietet den Vorteil, dass keine zusätzliche Software installiert werden muss, um zum Beispiel Informationen aus dem Engineering dem Bedien- oder Wartungspersonal zugänglich zu machen.

Wie in Abbildung 83 dargestellt, können aus einem CAD-System (zum Beispiel NX der Firma Siemens AG) über das standardisierte Austauschformat „Jupiter Tesselation (JT)“ zudem Topologie- und Geometriedaten einer Maschine oder Anlage ausgeleitet werden.

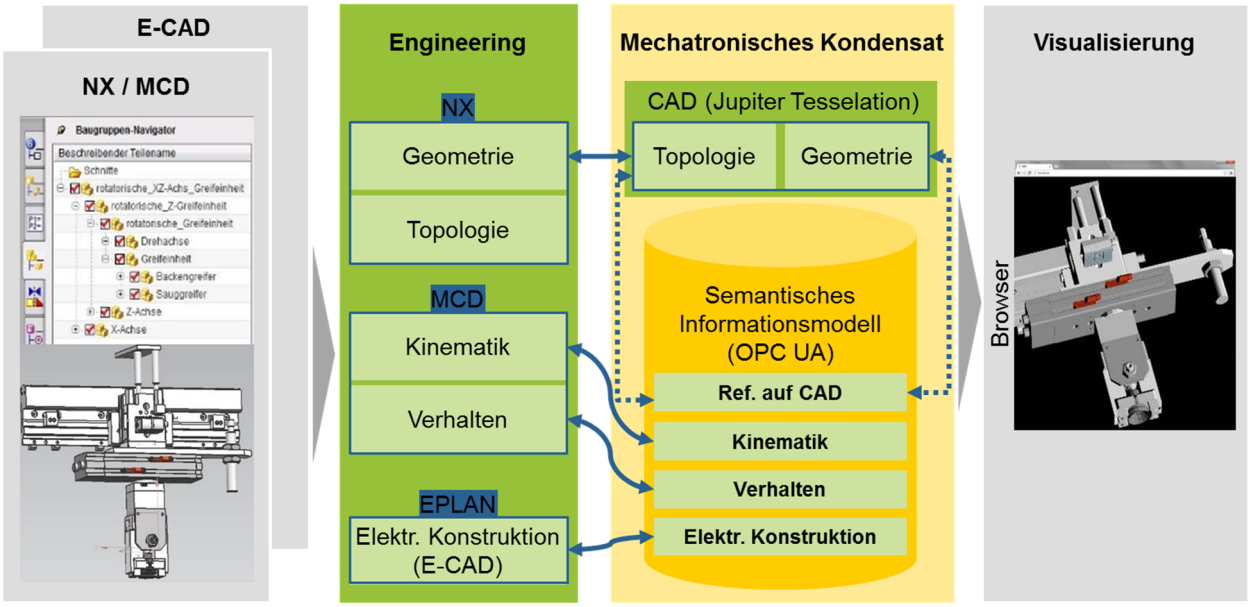


Abbildung 83: Verwendung eines mechatronischen Kondensats als Grundlage zur Realisierung eines webbasierten Visualisierungskonzepts

Darüber hinaus werden Informationen bezüglich Kinematik, Verhalten und elektrischer Konstruktion in ein semantisches Informationsmodell unter Verwendung des Standards *OPC UA* [37] extrahiert (vgl. Kapitel 2.1.4). Dabei referenziert das Informationsmodell auf die Geometrie-Daten, die mittels JT exportiert wurden. Ein kombinierter Datensatz bestehend aus einem JT-Datensatz und einem semantischen Informationsmodell wird als „mechatronisches Kondensat“ bezeichnet. Durch die Verwendung von OPC UA bieten sich zwei grundsätzliche Möglichkeiten der Zurverfügungstellung des semantischen Informationsmodells.

Einerseits kann das OPC UA Informationsmodell in dem XML-basierten Austauschformat „OPC UA XML“ persistiert und offline dateibasiert verwendet werden. So werden

lediglich die Modellierungs- und Abbildungsfähigkeiten von OPC UA genutzt. Die ereignisdiskrete Kommunikationsfähigkeit von OPC UA wird hierbei nicht genutzt.

Andererseits kann das OPC UA Informationsmodell mittels eines Servers interaktiv ausgeführt werden, wobei der Zugriff über Clientmechanismen erfolgen kann. Dies ermöglicht die Interaktion zwischen verschiedenen Beteiligten, die das OPC UA Informationsmodell zum Beispiel im Rahmen eines kollaborativen Engineerings sowohl als Consumer bzw. Provider bestimmter Informationen nutzen können. Da OPC UA bereits Publish&Subscribe-Mechanismen integriert, können die Interessenten bestimmter Informationen bei Änderung aktiv benachrichtigt werden.

Darüber hinaus kann ein OPC UA Informationsmodell in unterschiedlichen Umgebungen genutzt werden. Dies soll an dieser Stelle kurz differenziert dargestellt werden: In den vorausgegangenen Kapiteln (vgl. Kapitel 4.2.4 und 4.3) wurde beschrieben, wie OPC UA als Austauschformat im Engineering durch die Integration in NX/MCD verwendet wird. Ziel dabei ist die Anreicherung eines Informationsmodells, das auf einer Steuerung (SPS) zur Verfügung gestellt werden kann. Während des Engineerings liegt der Hauptfokus bei Verwendung des OPC UA Standards auf der online-fähigen Interkonnektivität zwischen verschiedenen Domänen, Rollen, Benutzern und Softwarewerkzeugen.

Bei Integration des mechatronischen Kondensats auf Steuerungsebene unter Verwendung von OPC UA kann das im Engineering beschriebene Verhalten zur Ausführung des Prozesses auf Steuerungsebene verwendet werden (siehe Abbildung 84).

Somit stehen auf Steuerungsebene alle semantischen Informationen wie zum Beispiel Geometrie/Topologie, Kinematik, elektrische Konstruktion und das projektierte Verhalten aus dem Engineering zur Verfügung. Darüber hinaus können die Realdaten der Steuerung zur betriebsbegleitenden Simulation genutzt werden.

Für die soeben dargestellten Szenarien (mechatronisches Kondensat im Engineering bzw. auf Steuerungsebene) wurde ein Visualisierungskonzept umgesetzt, das folgende Anforderungen erfüllt:

- Das Visualisierungskonzept soll bezüglich der verwendeten Geometriedaten herstellerunabhängig sein.
- Die Visualisierung soll sowohl während des Engineerings als auch zur Laufzeit ausgehend von der Steuerungsebene einsetzbar sein.
- Das Visualisierungskonzept soll Technologien verwenden, die auch für Wide Area Networks (WAN) geeignet sind.
- Zur Darstellung der Visualisierung soll keine weitere Software auf dem Visualisierungsclient installiert werden müssen.
- Die Visualisierung soll auf PCs, Handys und Tablets funktionieren.

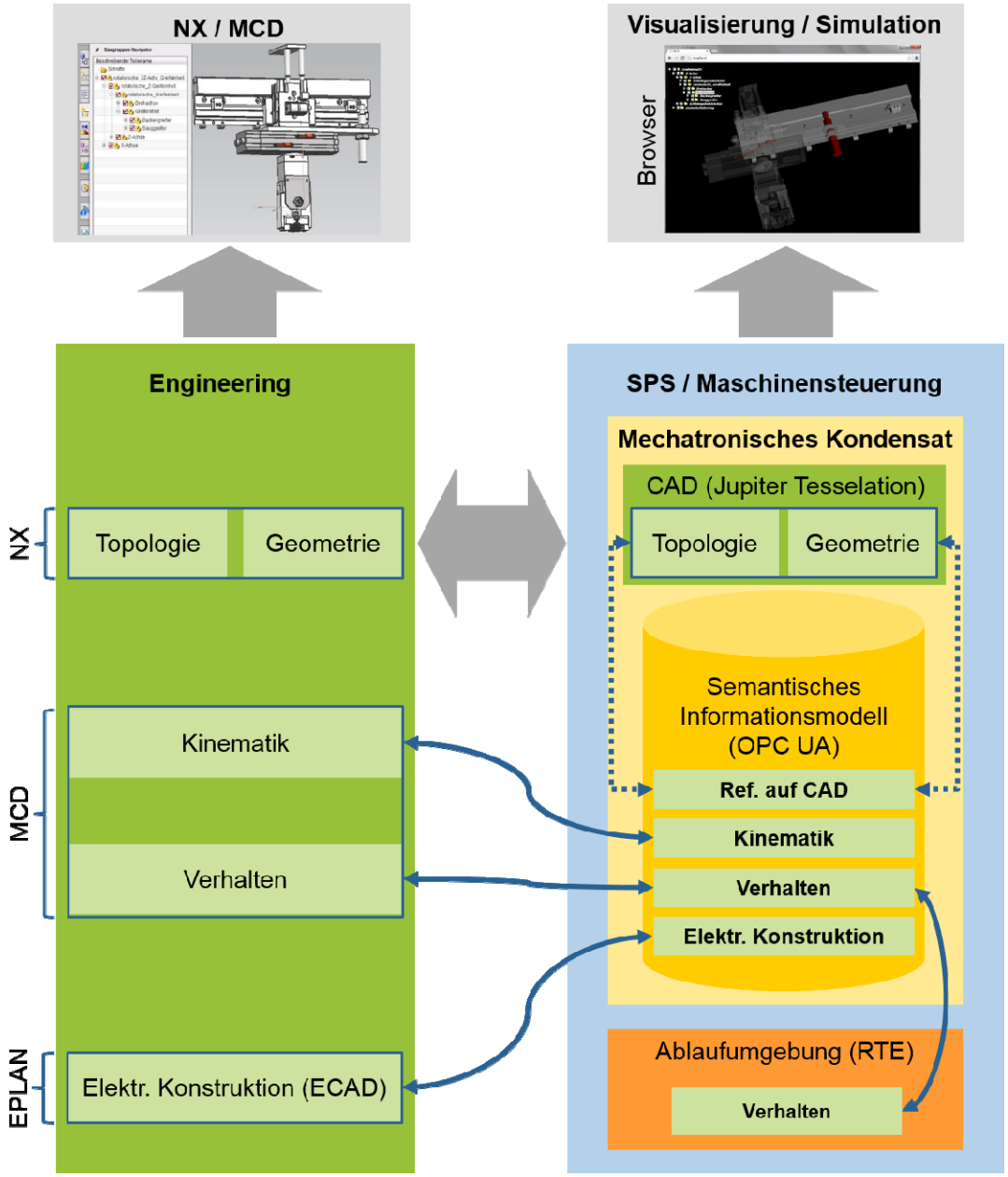


Abbildung 84: Integration des mechatronischen Kondensats in Industriesteuerungen (SPS bzw. MC)

Um die oben genannten Anforderungen zu erfüllen, wurde ein webbasiertes Client/Server- Konzept entwickelt. Wie in Abbildung 85 dargestellt, ist die Ausgangslage eine existierende Konstruktion, die im JT-Format vorliegt. Da das JT-Format ein herstellerübergreifender Standard ist, besteht somit keine Festlegung auf ein bestimmtes CAD-System. Im Web-Browser sollen die Geometriedaten mittels des Web-Standards WebGL graphisch dargestellt werden. Hierfür ist eine Umwandlung der JT-Datei in das Datenformat JSON (**J**ava**S**cript **O**bject **N**otation) erforderlich, um im Browser des Client-Rechners die Geometriedaten mittels JavaScript handhaben zu können.

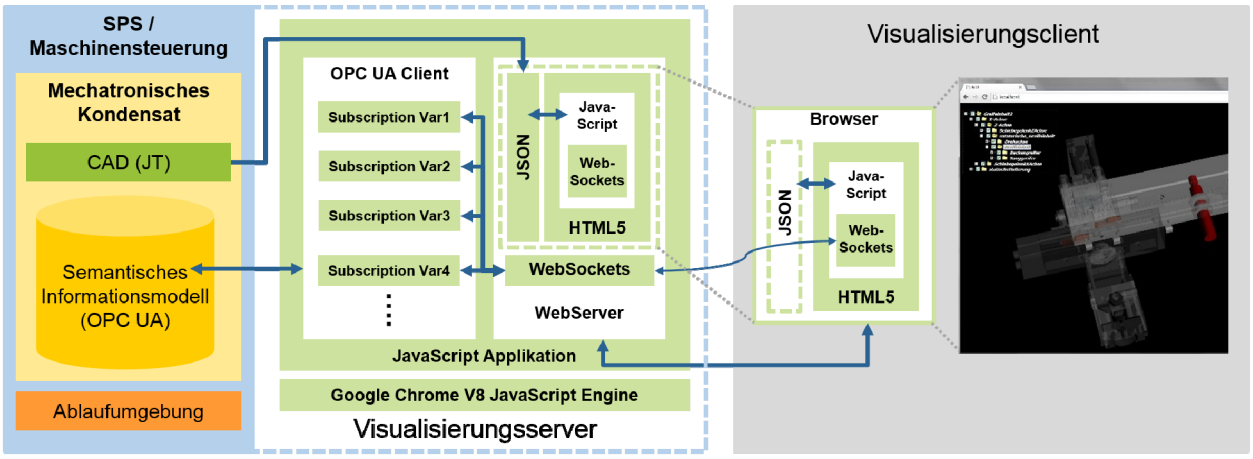


Abbildung 85: Überblick über die Realisierung des Visualisierungskonzepts

Des Weiteren wird von Google Chrome die V8 JavaScript Engine eingesetzt. Diese erlaubt eine serverseitige Ausführung von JavaScript-Applikationen. Da der Source-Code der V8 Engine unter der MIT-Lizenz zur Verfügung steht, ist eine kommerzielle Verwendung auf PC- als auch auf Steuerungsplattformen möglich.

Die verwendete Server-Applikation implementiert einen Web-Server, der über Web-Sockets eine echtzeitfähige, bidirektionale Kommunikation zwischen Web-Client und Web-Server zur Verfügung stellt. Des Weiteren implementiert die Server-Applikation einen OPC UA Client, der sich für die Benachrichtigung bei Wertänderung bestimmter Variablen eines OPC UA Servers registrieren lassen kann (Subscription). Bei Wertänderung einer OPC UA Variable, auf die der OPC UA Client eine Subscription angelegt hat, wird der aktualisierte Wert über Web-Sockets zum Visualisierungsclient propagiert.

Clientseitig erfolgt der Aufruf des Webservers mittels Browser über einen Domänennamen bzw. IP-Adresse sowie den Port des Visualisierungsservers. Der Browser des Clients ruft eine HTML5-Webseite des Web-Servers auf. Mittels des integrierten JavaScript-Codes werden die JSON-Dateien vom Webserver abgerufen und zum dynamischen Aufbau eines WebGL-Modells verwendet. In diesem Modell ist jedes geometrische Objekt individuell manipulierbar, was unter Verwendung der kinematischen Information des Informationsmodells die Kinematisierung des Geometriemodells in WebGL ermöglicht.

Wie in Abbildung 85 dargestellt, besteht zwischen der Website im Browser des Clients und dem Web-Server eine bidirektionale Kommunikationsverbindung mittels Web-Sockets. Diese wird zum Beispiel bei einer betriebsbegleitenden Visualisierung für die Aktualisierung von Gelenkstellungen des visualisierten Modells im Browser verwendet. Darüber hinaus ist natürlich mittels des 3D-Geometriemodells auch eine Visualisierung von vorgegebenen Parametern zum Beispiel durch benutzerdefinierte Eingaben möglich. Dies kann zur Visualisierung von zum Beispiel einzelner kinematischer Zustände

einer Maschine aber auch zur Visualisierung von Bewegungsabfolgen verwendet werden.

Da der Visualisierungsserver direkt auf Steuerungsebene integriert werden kann, erlaubt dies eine signifikante Erweiterung der webbasierten Funktionalitäten, die heutige Steuerungen zur Verfügung stellen. Darüber hinaus ist auch ein eigenständiger Einsatz des Visualisierungsservers möglich, wodurch auch ältere Steuerungsarchitekturen mittels OPC UA Client-Funktionalität des Visualisierungsservers mit diesem Visualisierungskonzept eingebunden werden können.

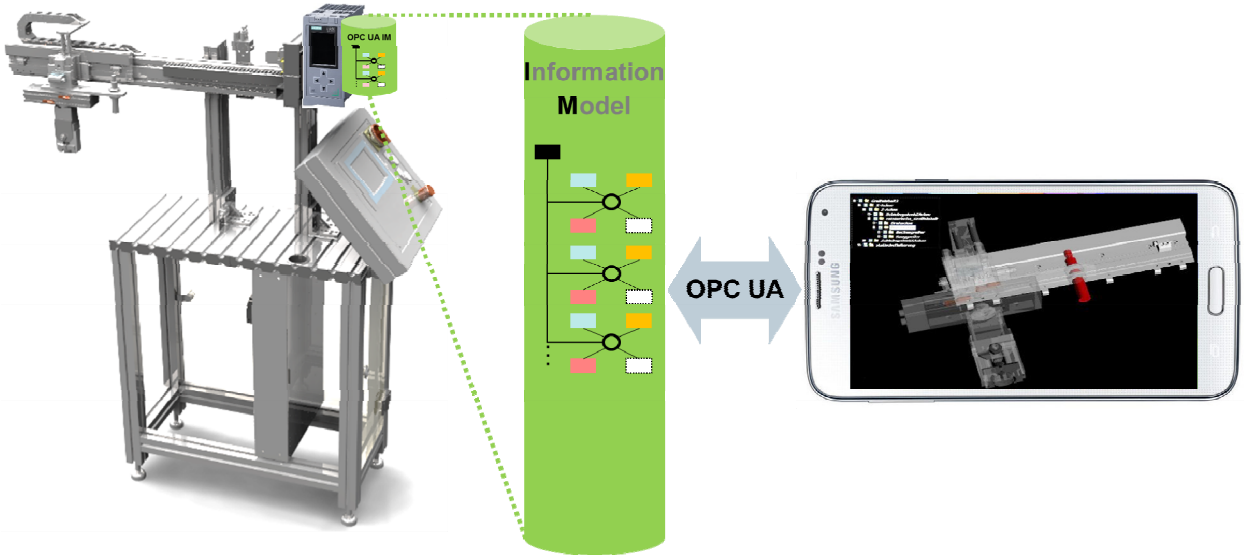


Abbildung 86: Beispielhafte Visualisierung einer Handhabungseinheit im Browser eines Smartphones

Die beispielhafte Darstellung einer visualisierten Handhabungseinheit unter Verwendung des beschriebenen Visualisierungskonzepts ist in Abbildung 86 im Browser eines Smartphones dargestellt und belegt die Einsatzfähigkeit sowohl für Mobilgeräte als auch für normale PCs.

4.7 Zusammenfassung

Dieses Kapitel umfasst die Beschreibung der Referenzimplementierung in den Bereichen „Engineering“, „semantisches Informationsmodell“, „Ausführungsumgebung“ und „webbasierte Visualisierung“.

Für die Umsetzung des Konzepts auf Basis eines existierenden Engineering-Werkzeugs wird die Konstruktionssoftware NX mit dem Erweiterungsmodul „Mechatronics Concept Designer“ gewählt. Mit Hilfe der C++ Programmierschnittstelle NXOpen kann die Funktionalität von NX erweitert werden, um die Realisierbarkeit des semantischen Beschreibungskonzepts darzustellen.

Die Umsetzung des semantischen Informationsmodells erfolgt mit dem Kommunikations- und Beschreibungsstandard OPC UA. Hier ist die Erstellung eines individuellen Namensraums erforderlich, der Objekt-, Daten- und Referenztypen enthält, die zur semantischen Beschreibung automatisierter Produktionssysteme eingesetzt werden können. Dieser Namensraum wird als Bestandteil eines OPC UA Servers an die Engineering-Werkzeuge „NX“ sowie „Process Simulate“ angebunden und über eine Funktionsschnittstelle bedient.

Auf Basis einer bestehenden Hardware- und Betriebssystemarchitektur wird die Umsetzung einer Ablaufumgebung mit Hilfe von Servicebausteinen beschrieben. Diese vorkompilierten, elementaren Servicebausteine erlauben eine Projektierung mittels Verschaltung verschiedener Servicebausteine zu einer Gesamtfunktionalität. Dies ist eine Grundvoraussetzung, um eine Abbildung der Verhaltensbeschreibung aus dem semantischen Informationsmodell auf Verschaltungsnetzwerke der Ablaufumgebung vornehmen zu können.

Abschließend wird erläutert, wie mit Hilfe des semantischen Informationsmodells steuerungsseitig eine webbasierte Visualisierung automatisiert generiert werden kann. Diese ist zum Beispiel zur Prozessvisualisierung oder aber auch zur Diagnose einsetzbar.

5 Diskussion des Potentials des vorgestellten Konzepts und dessen Referenzimplementierung

Das hier vorgestellte Konzept zur semantischen Beschreibung von Automatisierungssystemen sowie dessen prototypische Implementierung untergliedern sich in verschiedene Teile, die nachfolgend differenziert betrachtet und diskutiert werden. Dabei soll darauf eingegangen werden, welche Potentiale und neuen Anwendungsbereiche sich vom vorgestellten Konzept und dessen prototypischer Implementierung ableiten lassen.

5.1 Diskussion des Potentials des Konzepts zur semantischen Beschreibung automatisierter Produktionssysteme

Das dargestellte Konzept zur semantischen Beschreibung von automatisierten Produktionssystemen integriert Konzepte aus dem Engineering wie zum Beispiel *Requirements* und *Functional Engineering* unter Berücksichtigung einer funktionalen, komponentenbasierten Strukturierung der Automatisierungstechnik. Hier lassen sich gewisse Analogien zur Methode der Aufgliederung und Verknüpfung zur Problem- und Systemstrukturierung (siehe Abbildung 3) erkennen, bei der eine Gesamtproblemstellung in Teil- und Einzelprobleme zerlegt wird. Die Einzelprobleme wiederum können durch Einzellösungen (Systemelemente), die Teilprobleme durch Teillösungen erfüllt werden. Dieses Vorgehen lässt sich zum Beispiel auf funktionale Anforderungen des Anlagen-Engineerings übertragen. Funktionale Anforderungen an einen Prozess werden in funktionale Teilanforderungen von Teilprozessen aufgeteilt. Diese Modularisierung wird bis auf eine Ebene fortgeführt, auf der Automatisierungskomponenten mit definierten Funktionalitäten die geforderten funktionalen Anforderungen erfüllen können. Aus funktionalen Lösungen von Teilprozessen mittels Automatisierungskomponenten kann dann eine Gesamtlösung gebildet werden. Das entwickelte Konzept verwendet hierfür die Strukturierungselemente „Rolle“ (siehe Kapitel 3.1.2), „Komponente“ (siehe Kapitel 3.1.3) sowie zur (De-)Komposition das Aggregationsprinzip (siehe 3.1.5). Die Verknüpfung zwischen Komponenten erfolgt mittels einer Funktionsschnittstelle (siehe Kapitel 3.1.6). Diese Art der Strukturierung im Engineering erlaubt eine Abbildung auf die Struktur eines automatisierten Produktionssystems.

Der zweite Teil des Konzepts befasst sich mit der semantischen Beschreibung des Verhaltens automatisierter Produktionssysteme mit dem Ziel, die resultierende Verhaltensbeschreibung zur Projektierung in der Steuerungsebene verwenden zu können. Zentraler Bestandteil dabei ist die funktionsorientierte Sicht auf elementare und aggregierte Automatisierungskomponenten, da jede Komponente mittels Funktionen angesprochen wird, die eine standardisierte Funktionsschnittstelle zur Verfügung stellt.

Weitere essentielle Bestandteile des dargestellten Konzepts sind die semantisch beschriebenen Indirektionselemente „Interaktionspunkt“ und „ProcessFunctionConverter“. Mit Hilfe von Interaktionspunkten können Zustände der Komponenten definiert werden, die von einer realen Implementierung, Parametrierung oder Einbindung realer Sensorik abstrahieren. Das Beschreibungselement „ProcessFunctionConverter“ erlaubt eine semantische Beschreibung des Aktuator-Aufrufs. Durch diese beiden Indirektionselemente ist die Erstellung eines Ablaufprogramms möglich, das die tatsächliche Intension der Automatisierungsaufgabe beschreibt. An einem Pseudocode-Beispiel (siehe Tabelle 7) bestehend aus einer Abfragebedingung der Sensorik (IAP) sowie einem Funktionsaufruf der Aktuatorik (PFC) wird der Vorteil zusätzlicher Semantik ersichtlich.

Tabelle 7: Darstellung der grundlegenden Funktionsweise der semantischen Indirektionselemente „Interaktionspunkt“ und „ProcessFunctionConverter“

Keine Verwendung von semantischen Indirektionselementen	Verwendung von semantischen Indirektionselementen
IF (Näherungssensor_1 = TRUE)	IF („Material_vorhanden(IAP)“)
THEN	THEN
Ventil_1::= TRUE;	„Greifer_schließen(PFC)“
END_IF	END_IF

Die Funktion einer aggregierten Komponente kann aus einer sequentiellen oder aber auch nebenläufigen Ablaufsequenz von Funktionsaufrufen unterlagerter Komponenten bestehen. Diese Ablaufsequenz wird analog zur Struktur eines automatisierten Produktionssystems innerhalb der jeweiligen Komponente gespeichert, wodurch sich automatisch eine Verteilung des gesamten Prozessablaufs über alle beteiligten Komponenten ergibt. Dies bringt mehrere Vorteile mit sich. Einerseits unterstützt eine solche Modularisierung des Steuerungsprogramms zum Beispiel eine verteilte Ausführung unter Verwendung dezentraler Peripherie oder aber auf Multicore-Steuerungen. Andererseits trägt dies zur Realisierung eines funktionalen Komponententauschs bei, da lediglich der Funktionsaufruf ausgehend von der übergeordneten zur getauschten Komponente angepasst werden muss.

Durch die Integration des vorgestellten Konzepts zur semantischen Modellierung automatisierter Produktionssysteme in Engineering-Werkzeuge entsteht kein weiterer Mehraufwand bei der Projektierung von Automatisierungslösungen. Es wird eine technische Möglichkeit aufgezeigt, wie eine semantische Integration verschiedener eigenständiger Engineering-Werkzeuge in ein gemeinsames durchgängiges Informa-

tionsmodell geschaffen werden kann. Dies setzt jedoch einen konsequenten Einsatz von Engineering-Werkzeugen voraus, um die erforderlichen Informationen für ein durchgängiges Engineering zusammentragen zu können. Aus diesem Grund wird sich der Ansatz der semantischen Modellierung vor allem bei Anwendungsbereichen lohnen, die eine hohe Komplexität besitzen und häufig an neue Gegebenheiten adaptiert werden müssen.

Die wesentlichen Einsparpotentiale bei Anwendung des vorgestellten Konzepts liegen in der Verbesserung der vertikalen IT-Integration zwischen Engineering und Steuerungsebene sowie in der horizontalen IT-Integration zwischen Engineering-Werkzeugen. Die vertikale IT-Integration unterstützt den bidirektionalen Informationsaustausch zwischen Engineering und Steuerungsebene. Um dies zu ermöglichen, ist eine semantische Erweiterung der Steuerungsebene bezüglich mechanischer Struktur, Funktion und Verhalten erforderlich. Das bildet die Grundlage für eine bessere Unterstützung der Synchronisierung zwischen automatisiertem Produktionssystem und dem Engineering während des Life-Cycles.

Darüber hinaus bietet die komponentenbasierte Modularisierung der Steuerungslogik den Vorteil einer skalierbaren Verteilung auf dezentrale Peripherie, wobei eine Realisierung mittels zentraler SPS weiterhin möglich ist. Zentrales Ziel dabei ist die Nutzung des funktional spezifizierten Prozessverhaltens der Engineering-Werkzeuge zur Generierung einer Steuerungslösung für das Produktionssystem, was den Programmieraufwand insbesondere während des gesamten Life-Cycles des Produktionssystems deutlich senkt.

Durch die zusätzliche Semantik auf Steuerungsebene ergeben sich neue Einsatz- und Nutzungsmöglichkeiten zum Beispiel im Bereich Wartung und Diagnose. Durch Verwendung der Struktur- und Prozessinformationen einer Maschine können Maschinenparameter bzw. Fehlerszenarien besser interpretiert und bedarfsgerechte Wartung und Instandsetzung geplant werden. Aufgrund der komponentengranularen Modularisierung ist eine Diagnose [90] während der Prozessausführung bis hin zur jeweiligen Automatisierungskomponente möglich.

Der zusätzliche Erstaufwand zur Projektierung wird sich im Vergleich zur Standard-Projektierung wenn überhaupt nur geringfügig erhöhen, da der größte Teil der erforderlichen Informationen im Hintergrund der verwendeten Engineering-Applikationen automatisch verarbeitet werden kann. Bisher nicht erforderliche Eingaben zur Erstellung bzw. Manipulation von Strukturierungen und Hierarchien werden andererseits zu einer Verbesserung der Handhabbarkeit der Komplexität sowie zu einer einfacheren Austauschbarkeit von Teilmodulen des zu projektierenden Produktionssystems führen. Damit dies gelingen kann, ist ein umfangreicher Einsatz von Engineering-Werkzeugen erforderlich, um die erforderlichen Informationen während der verschiedenen Engineering-Phasen erheben zu können. Eine solch umfangreiche Unterstützung des Entwicklungsprozesses mit Hilfe von Engineering-Werkzeugen lohnt sich

überwiegend bei sehr komplexen Systemen mit voraussichtlich vielen Systemänderungen.

Aufgrund der hardwareunabhängigen Beschreibung der Steuerungslösung bietet das hier vorgestellte Konzept den Vorteil, dass der Ressourcen- und Kommunikationsbedarf der Steuerungslösung im Rahmen bestimmter Vorgaben wie zum Beispiel Kommunikationslatenzen, Echtzeitfähigkeit etc. dynamisch skalierbar ist. Somit ist eine Verteilung der Steuerungslösung unter Verwendung von dezentraler Peripherie als auch der Einsatz von zentralen SPSen möglich. Diese Flexibilität spielt insbesondere bei der Wandlungsfähigkeit einer Anlage während des Lebenszykluses eine wesentliche Rolle, da die Planung entsprechend der geänderten Anforderungen dynamisch adaptiert werden kann.

Das hier vorgestellte Konzept basiert auf einer Informationsmodellierung und Abbildung mit OPC UA unter Verwendung benutzerdefinierter Objekt-, Daten- und Referenztypen. Der Beschreibungsstandard AutomationML ist ebenfalls in der Lage, Struktur und Verhalten von automatisierten Produktionssystemen zu beschreiben. Für eine erfolgreiche vertikale IT-Integration zwischen Engineering-Applikationen und Steuerungsebene ist OPC UA erforderlich. Eine Zusammenarbeit zwischen beiden Standards wurde bereits vereinbart [91]. Die Arbeiten an der Spezifikation der Abbildungsvorschriften (Companion Specification) sind aktuell noch nicht abgeschlossen.

5.2 Diskussion des Potentials der Referenzimplementierung

Die prototypische Umsetzung des Konzepts zur semantischen Beschreibung von automatisierten Produktionssystemen unterteilt sich in die Arbeitsbereiche

- Prototypische Realisierung eines semantischen Informationsmodells,
- Prototypische Umsetzung in Engineering-Werkzeugen,
- Prototypische Implementierung einer Ablaufumgebung und
- Prototypische Implementierung einer webbasierten Visualisierung.

Diese sollen nachfolgend einzeln betrachtet werden.

Grundlage für die semantische Beschreibung von automatisierten Produktionssystemen sind die zur Verfügung stehenden Beschreibungselemente, die zur Erzeugung eines semantischen Informationsmodells genutzt werden können. Technisch wurde dies mit Hilfe von OPC UA umgesetzt, was verschiedene Vorteile bietet. So kann das Typsystem des semantischen Informationsmodells bei Bedarf angepasst werden. Durch Verwendung eines eigenständig lauffähigen OPC UA Servers wirken sich Anpassungen des Informationsmodells nicht direkt auf die Architektur der Engineering-Werkzeuge aus. Lediglich eine semantisch korrekte Verwendung der zur Verfügung gestellten semantischen Beschreibungselemente des Informationsmodells ist zu gewährleisten. OPC UA lässt sich im Engineering auch ohne Probleme in Kombination bzw. neben PDM-Systemen einsetzten. Somit entspricht dies einer evolutionären

technischen Lösung, die in eine heterogene Tool-Landschaft zur Homogenisierung des Informationsaustauschs bis auf Feldebene genutzt werden kann. Das Informationsmodell ist durch den Einsatz von OPC UA onlinefähig abfragbar und stellt unter anderem Publish&Subscribe-Mechanismen zur Verfügung. Sind verschiedene Softwarewerkzeuge mit einem solchen OPC UA Informationsmodell verbunden, ist im Rahmen eines kollaborativen Engineerings ein sternförmiger Informationsaustausch zwischen allen Engineering-Entitäten bis hin zur Steuerungs- und Feldebene sowohl während der Planungs- und Realisierungsphase als auch während des Betriebs möglich. Dies stellt einen entscheidenden Unterschied zu anderen Standards im Engineering dar, da OPC UA neben den Fähigkeiten zur Modellbildung auch über Kommunikations- und Sicherungsmechanismen verfügt und darüber hinaus auf Laufzeitsysteme der Steuerungs- und Feldebene portiert werden kann. Dies eröffnet aus technischer Sicht einerseits Möglichkeiten der horizontalen Integration von Informationen aus dem Engineering bzw. dem Produktionssystem in ein Informationsmodell. Andererseits ist dadurch auch eine vertikale Integration zwischen Engineering und realer Anlage umsetzbar. Durch das in dieser Arbeit beschriebene Konzept ist so zum Beispiel eine Verbesserung der Verfolgbarkeit von Änderungen während des Anlagen-Life-Cycles möglich, da die Programmierung des Steuerungsprogramms über die Komponentenstruktur der Maschine bzw. Anlage verteilt ist. Änderungen im Ablaufverhalten können dadurch gezielt bestimmten Komponenten zugeordnet werden, was zum Beispiel den Anpassungsaufwand im Bereich der Konstruktion deutlich verringert.

Aus Sicht der Steuerungsebene bietet die Verfügbarkeit der Informationen aus dem Engineering mit Hilfe eines OPC UA Informationsmodells eine Vielzahl von potentiellen Möglichkeiten. So stehen im Informationsmodell zum Beispiel Informationen bezüglich Geometrie und Kinematik zur Verfügung. Auf dieser Datengrundlage lassen sich Berechnungen und Analysen auf einer SPS durchführen, die heute im Bereich der Robotik angesiedelt sind, wie zum Beispiel Erreichbarkeits- und Kollisionsanalysen sowie eine Bahnplanung mittels Ablaufsequenzen diskreter Konfigurationen einer Kinematik. Auch die Abbildung des Arbeitsraums mit Hindernissen ist möglich. So könnte auch ein Lösungsvorschlag zur Überführung der Maschine vom manuellen Modus in den Automatikmodus automatisiert erzeugt werden, was einen Wiederanlauf nach einem Fehlerfall beschleunigen kann. Diese Fähigkeiten würden es einer Steuerung erlauben, die Durchführbarkeit eines Prozesses zum Beispiel unter Berücksichtigung des zur Verfügung stehenden Arbeitsbereichs zu bewerten.

Des Weiteren kann mit Hilfe der zusätzlichen Semantik auf Steuerungsebene ein Bezug hergestellt werden zwischen dem Setzen eines Ausgangs und der beabsichtigten Reaktion. Sollte bei der realen Anlage ein Fehler in der Verkabelung der Ein- und Ausgänge der Steuerung vorliegen, lässt sich so der Fehler schneller eingrenzen. Durch Nutzung der Geometriedaten lassen sich zum Beispiel Distanzen von Verfahrwegen ermitteln. Unter Berücksichtigung der Leistungsdaten bzw. der Konfigura-

tion eines Aktuators können automatisiert Laufzeitüberwachungen für die Teilprozesse generiert werden. Mit Hilfe der in Kapitel 4.6 dargestellten Realisierung einer webbasierten Visualisierung unter Verwendung der Semantik des Informationsmodells (siehe 4.3) lassen sich noch weitere Vorteile erzielen. Aus dem semantischen Informationsmodell lässt sich automatisiert eine webbasierte Visualisierung unter Verwendung der enthaltenen Geometriedaten realisieren. Darauf aufbauend können folgende Visualisierungsaspekte umgesetzt werden:

- Visualisierung von mechanischen Zuständen bzw. Gelenkstellungen
- Visualisierung des Arbeitsraums inkl. Hindernisse
- Simulation der Erreichbarkeit bestimmter Positionen im Raum mittels einer vorgegebenen Kinematik
- Simulation von Bewegungsabfolgen zum Beispiel zur Überführung vom manuellen Modus in den Automatikbetrieb
- Visualisierung der Referenzen zwischen Ein- und Ausgängen der EA-Leiste und den daran angeschlossenen Automatisierungskomponenten
- Visualisierung betroffener und angrenzender Komponenten im Fehlerfall

Die vorgestellten Visualisierungsmöglichkeiten können im Bereich HMI sinnvolle Ergänzungen darstellen ohne den Projektierungsaufwand signifikant zu erhöhen.

5.3 Semantisches Informationsmodell als Bindeglied zwischen Engineering-Werkzeugen und Steuerungsebene

Die Verwendung eines semantischen Informationsmodells dient der Integration von Engineering-Daten über den gesamten Planungsprozess hinweg bis hin zur Realisierung bzw. dem Betrieb eines automatisierten Produktionssystems. Dabei dient das Informationsmodell als semantische Datendrehscheibe horizontal zwischen verschiedenen Engineering-Werkzeugen als auch vertikal während den Planungs- und Entwicklungsphasen bis hin zur Steuerungs- und Feldebene. Dies ist in Abbildung 87 dargestellt und wurde bereits in Teilen prototypisch umgesetzt.

Im Konstruktionswerkzeug NX von Siemens wird die mechanische Konstruktion für Maschinen und Teilanlagenbereiche durchgeführt. Diese Geometriedaten können entweder vom Erweiterungsmodul „Mechatronics Concept Designer“ oder aber auch mittels JT-Export von dem Softwarewerkzeug „Process Simulate“ weiterverarbeitet werden. In beiden Fällen können die Geometriedaten kinematisiert sowie ein Ablaufverhalten erstellt werden. Sowohl „NX/MCD“ als auch „Process Simulate“ unterstützen die Anbindung eines OPC UA Servers mit Hilfe des in Kapitel 4.2.4 beschriebenen Wrapper-Konzepts. Dabei adaptiert der Wrapper von „Process Simulate“ dessen .NET-Programmierschnittstelle an die C++ Dynamic Link Library (DLL) des OPC UA Servers und sorgt für ein korrekte Überführung (Marshalling) der elementaren Datentypen bzw. Datenstrukturen zwischen .NET und C++.

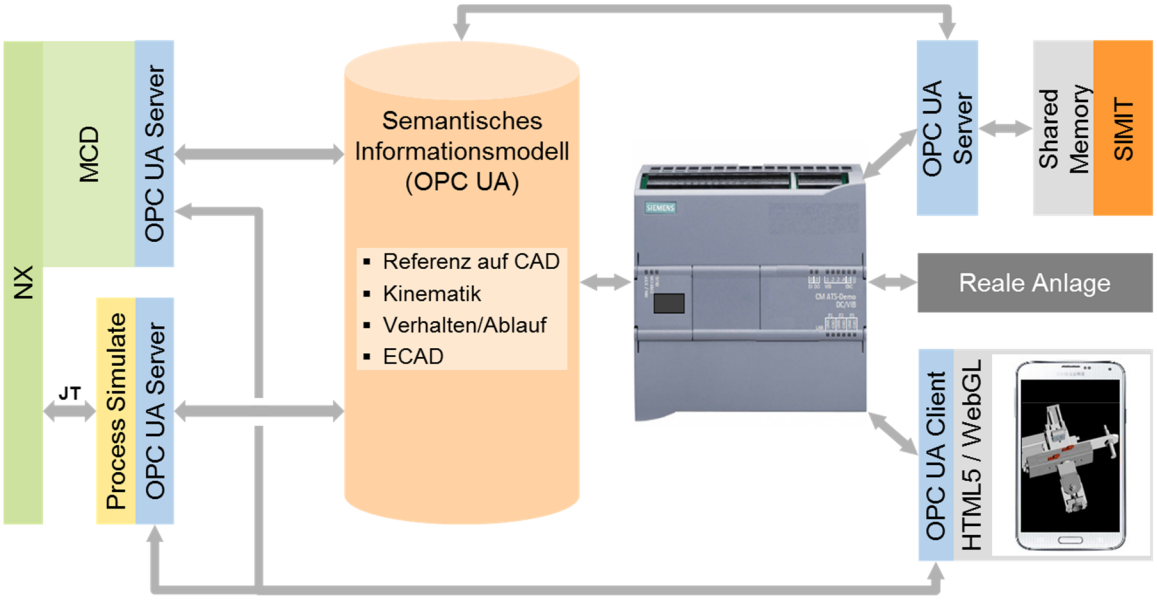


Abbildung 87: Darstellung einer semantischen Kopplung zwischen Engineering-Simulations- und Visualisierungswerkzeugen und der Steuerungsebene

Eine Einbindung von ECAD-Daten ist erforderlich, um eine Automatisierungskomponente mit einer bestimmten Funktionalität mit Hilfe einer drahtgebundenen Verbindung auf definierten Ein- bzw. Ausgängen einer Steuerung aufzulegen. Diese Ein- bzw. Ausgänge dienen dann der Steuerung als elementare, funktionale Schnittstelle zu dieser Komponente. Dies wurde bisher noch nicht prototypisch umgesetzt und stellt jedoch auch keinen Konflikt zum grundlegenden Konzept der semantischen Beschreibung von automatisierten Produktionssystemen dar.

Die dargestellte Steuerung wurde um die Ausführungsfähigkeit für verschaltete Servicebausteinnetzwerke (siehe Kapitel 4.5) erweitert, die aus dem semantischen Informationsmodell abgeleitet werden können. Die Steuerung kann dadurch entweder eine reale Anlage, ein virtuelles Anlagenmodell (hier wurde SIMIT der Firma Siemens eingesetzt) oder aber auch ein webbasiertes Visualisierungsmodell steuern. Für die virtuelle Inbetriebnahme sind alle erforderlichen Informationen im semantischen Informationsmodell vorhanden, sodass ein automatisierter Modellaufbau gelingt. Diese Arbeiten wurden bisher noch nicht implementiert, stehen jedoch konzeptionell in keinem Widerspruch zur beschriebenen Vorgehensweise.

Wie in Abbildung 88 dargestellt, bildet das OPC UA Informationsmodell das zentrale Bindeglied für eine semantische Kopplung bzw. für einen Informationsaustausch zwischen verschiedenen Engineering-Werkzeugen (horizontale IT-Integration) als auch zwischen Engineering-Werkzeugen und Steuerungsebene (vertikale IT-Integration). Da OPC UA als Server-Applikation online lauffähig ist, bietet dieser Standard den Vorteil, dass eine enge Kopplung zwischen den internen Objektmodellen der Engineering-Applikationen und einem OPC UA Objektmodell möglich ist. Dies erlaubt ei-

ne Synchronisierung der Informationen zwischen verschiedenen Engineering-Werkzeugen und ermöglicht eine technische Realisierung von kollaborativen Engineering-Ansätzen, ohne hierfür zentrale PDM-Systeme einsetzen zu müssen. Dies bietet zum Beispiel Einsatzmöglichkeiten während der Inbetriebnahmephase im anlagennahen Umfeld.

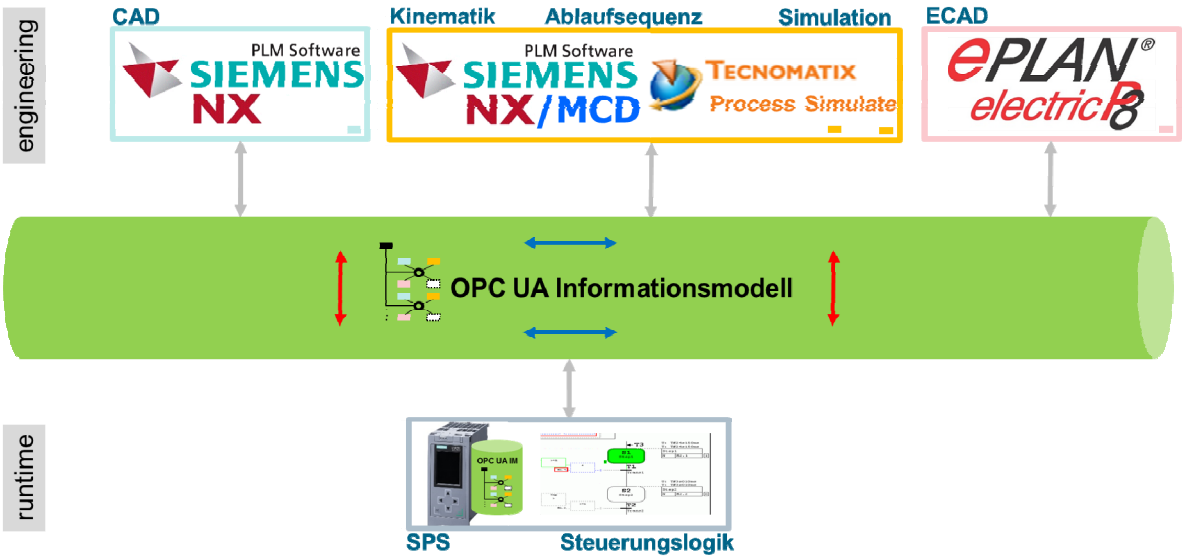


Abbildung 88: Überblick über das Gesamtkonzept zur Verbesserung der horizontalen und vertikalen IT-Integration mit Hilfe von OPC UA

Da OPC UA auch auf speicherprogrammierbaren Steuerungen eingesetzt werden kann, bietet dies den Vorteil, dass Informationsmodelle aus dem Engineering auf einer SPS abgelegt werden können. Darüber hinaus sind diese Informationsmodelle online abfragbar und während der Laufzeit dynamisch adaptierbar. Durch eine Abbildung der internen Objektstruktur einer SPS in das Objektmodell von OPC UA wird die technische Möglichkeit geschaffen, das Ablaufprogramm einer SPS durch ein Objektmodell in OPC UA zu erstellen. Somit bildet OPC UA das Bindeglied zum vertikalen Informationsaustausch zwischen Engineering-Software und der Steuerungsebene.

5.4 Zusammenfassung

In diesem Kapitel wird das Potential des Konzepts zur semantischen Beschreibung von automatisierten Produktionssystemen sowie dessen Referenzimplementierung diskutiert. Dabei werden nochmals die Kernelemente des Konzepts auszugsweise dargestellt und verdeutlicht. Danach wird auf die Realisierung des Konzepts eingegangen, wobei hier ebenfalls die möglichen Potentiale in den einzelnen Arbeitsbereichen der Realisierung (Informationsmodell, Engineering, Steuerung, Visualisierung) im Vordergrund standen. Es zeigt sich, dass zusätzliche Informationen aus dem Engineering auf Steuerungsebene verschiedene Anwendungsmöglichkeiten eröffnen, die heute in dieser Form nicht umsetzbar sind. Im Kapitel 5.3 wird ein Überblick über

eine informationsdurchgängige Architektur mit Hilfe von semantischen Informationsmodellen unter Verwendung von OPC UA dargestellt. Ziel dabei ist die technische Umsetzung einer verbesserten IT-Integration zwischen verschiedenen Engineering-Werkzeugen sowie zwischen Engineering-Werkzeugen und realem Produktionssystem.

6 Zusammenfassung und Ausblick

Zu Beginn der Arbeit werden die äußeren Einflüsse betrachtet, die auf produzierende Unternehmen wirken. Insbesondere im Spannungsfeld der industriellen Produktion in den Hochlohnländern besteht eine permanente Notwendigkeit der Verbesserung bzw. Effizienzsteigerung bei Planung, Realisierung und Betrieb automatisierter Produktionssysteme, um im internationalen Wettbewerb bestehen zu können.

Die theoretischen Grundlagen zur allgemeinen Förderung des Verständnisses des Kontexts, in dem sich diese Arbeit bewegt, werden im zweiten Kapitel behandelt. Hier sind unter anderem Methoden und Konzepte zur Strukturierung und Problemlösung, Grundlagen zur Modellbildung sowie verschiedene Beschreibungs- und Programmierstandards dargestellt, die unter anderem bei der Projektierung heutiger automatisierter Produktionssysteme Anwendung finden. Anschließend wird der Handlungsbedarf dargestellt. So wird zum Beispiel anhand der Kostenstruktur von Investitionskosten im Bereich der Steuerungstechnik (siehe Abbildung 23) deutlich, dass die Projektierung der Hard- und Software einen Kostenanteil von über 50% ausmacht. Bei der Zieldefinition wird aus diesem Grund wird hier angesetzt, um in diesem Bereich Verbesserungen zu erzielen. Als Lösungsansatz soll ein Konzept entwickelt werden, das mittels einer semantischen Beschreibung von Automatisierungssystemen eine Verbesserung des Informationsaustauschs zwischen Engineering und Steuerungsebene sowie zwischen verschiedenen Engineering-Werkzeugen realisiert.

Ausgehend von den theoretischen Grundlagen existierender Standards und Normen wird im dritten Kapitel die Entwicklung eines Konzepts zur semantischen Beschreibung von Automatisierungssystemen dargestellt. Es wird ein Konzept entwickelt, das an bestimmte Rahmenbedingungen zur Umsetzung einer Referenzimplementierung innerhalb einer bestehenden Tool-Landschaft adaptiert wurde. Als Ergebnis dieser konzeptionellen Ausarbeitung resultiert eine Methodik zur Beschreibung der Struktur und des Verhaltens automatisierter Produktionssysteme. Die Verhaltensbeschreibung wird bei diesem Vorgehen über die Baugruppenstruktur des Systems verteilt, woraus besondere Vorteile bezüglich funktionalem Komponententausch und einer verteilten Ausführung des Steuerungsprogramms erzielt werden können. Das Konzept wird ebenfalls in einem UML-Klassendiagramm abgebildet. Dabei stellt das Klassendiagramm bestimmte Strukturierungsmechanismen wie zum Beispiel „Rollen“ (siehe Kapitel 3.1.2) und „Komponenten“ (siehe Kapitel 3.1.3) objektorientiert mit Hilfe von Vererbungshierarchien dar.

Im nachfolgenden vierten Kapitel wird die Umsetzung des Konzepts auf Basis existierender Engineering-Werkzeuge (NX/MCD und Process Simulate) und einer implementierten Architektur zur Service-orientierten Formelberechnung von Verschaltungsnetzwerken (siehe Kapitel 4.4) erläutert. Zur technischen Realisierung eines semantischen Informationsmodells, das definierte Beschreibungselemente (Objekttypen, Datentypen

und Referenzen) zur Verfügung stellt, wurde der IEC-Standard OPC UA (siehe Kapitel 2.1.4) verwendet. Dieser kann sowohl im Engineering als auch auf Steuerungs- und Feldebene eingesetzt werden. Dadurch stehen Informationen aus dem Engineering auf Steuerungsebene zur Verfügung, was neue Einsatz- und Verwendungsmöglichkeiten eröffnet. So kann auf Steuerungsebene eine automatisierte Generierung einer webbasierten Visualisierung des zugrundeliegenden Prozesses umgesetzt werden, was konzeptionell beschrieben sowie prototypisch umgesetzt wurde.

Abschließend sind in Kapitel fünf die resultierenden Potentiale des dargestellten Konzepts sowie der prototypischen Referenzimplementierung beschrieben. Bei grundlegender Betrachtung resultieren diese aus der Verfügbarkeit zusätzlicher Informationen aus dem Engineering auf der Steuerungs- und Feldebene und können zur Umsetzung weiterer Funktionalitäten auf diesen Ebenen verwendet werden. Durch die technische Möglichkeit der Verwendung von OPC UA in Anbindung an Engineering-Werkzeuge sowie auf Steuerungs- und Feldebene ist ein vertikaler, semantisch definierter Informationsaustausch umsetzbar, was eine wesentliche Verbesserung der Synchronisierung zwischen Planungsdaten und Anlagen-Life-Cycle erlaubt.

Erzielte Forschungsergebnisse

Die hier vorliegende Arbeit kombiniert und integriert vorhandene Konzepte, Methoden und Standards zu einer technisch adaptierten Lösung, die eine horizontale Verknüpfung von verschiedenen Engineering-Softwareprodukten als auch die vertikale Verknüpfung zwischen Engineering und Steuerungsebene realisiert. Die erzielten Forschungsergebnisse können anhand der Arbeitsbereiche der Referenzimplementierung differenziert werden. Sie umfassen die Konzeption einer Beschreibungsmethode, die sowohl in Anbindung an Engineering-Werkzeuge als auch auf Steuerungs- und Feldebene realisiert werden kann, was prototypisch nachgewiesen wurde. Die Konzeption der Beschreibungsmethode umfasste die Entwicklung semantischer Beschreibungselemente, die zur Beschreibung von Struktur und Verhalten automatisierter Produktionssysteme verwendet werden können. Die technische Umsetzung koppelt dadurch eine Ablaufumgebung auf Steuerungsebene mit einem CAD-System wie zum Beispiel NX. Dadurch können Abläufe auf Steuerungsebene semantisch der Baugruppen-Struktur des CAD-Systems zugeordnet werden. Durch Verwendung des IEC-Standards OPC UA zur semantischen Abbildung der Informationen des Engineering bzw. der Runtime steht eine technische Realisierung zur Verfügung, die sowohl in Softwarewerkzeuge des Engineerings als auch in Geräte der Steuerungs- und Feldebene evolutionär integriert werden kann. Das dadurch resultierende Potential liegt in der technischen Umsetzbarkeit einer sternförmigen, semantischen Kopplung zwischen Planungs- bzw. Projektierungswerkzeugen des Engineerings und beteiligten Automatisierungskomponenten der Steuerungs- und Feldebene während der Planung, der Inbetriebnahme und dem Betrieb einer Anlage.

Ausblick auf weiterführende Arbeiten

Bei der Ausweitung des in dieser Arbeit vorgestellten Ansatzes ist eine Integration von weiteren Engineering-Werkzeugen in ein gemeinsames OPC UA Informationsmodell geplant. Aufgrund der unterschiedlichen Konzepte, Einsatzziele und Datenrepräsentationen der verschiedenen Engineering-Werkzeuge ist eine sorgfältige Abbildung der Objektrepräsentationen der einzelnen Applikationen in das OPC UA Informationsmodell notwendig. Dabei wird zwischen allgemeingültigen und werkzeugspezifischen Informationen differenziert, wobei werkzeugspezifische Informationen durch Spezialisierung der betroffenen Objekttypen abgelegt werden.

Heutige Engineering-Konzepte mit dem Ziel eines durchgängigen, konsistenten Einsatzes von Software-Werkzeugen basieren in der Regel auf einem zentralen PDM-System, das als Informationsdrehscheibe während des Engineering-Prozesses mechatronischer Produkte verwendet wird. Das in dieser Arbeit vorgestellte Konzept und dessen prototypische Realisierung fokussieren sich auf die Entwicklung automatisierter Produktionssysteme und eignen sich insbesondere für die Interaktion zwischen Steuerungstechnik und Engineering-Software. Vor allem bei der Inbetriebnahme werden im anlagennahen Umfeld Änderungen an den Projektierungsdaten vorgenommen, da bestimmte Randbedingungen während den vorangegangenen Engineering-Phasen nicht bekannt oder übersehen wurden. In diesem Umfeld bieten semantische Informationsmodelle die Möglichkeit, Änderungen an den Projektierungsdaten bis zum laufenden Produktionssystem im Feld begleiten zu können. Nach erfolgreicher IBN ist eine Überführung der semantischen Informationsmodelle in die Engineering-Projektierungsserver bzw. PDM-Systeme möglich. Geeignete Interaktionsmechanismen sind hierfür noch zu erarbeiten.

Für eine weitere Verbesserung der Anwendbarkeit ist die Umsetzung einer Bibliothek für Automatisierungskomponenten an das vorgestellte Konzept zu adaptieren. Dabei müssen die enthaltenen Automatisierungskomponenten bezüglich ihrer geometrischen Struktur, ihrer Funktion im Sinne des Prozesses (mechanisch, elektrisch, thermisch, chemisch, etc.) sowie ihrer elektrischen Funktionsschnittstellen beschrieben sein und in einer Abbildung vorliegen, die eine direkte Verwendung in einem OPC UA Informationsmodell erlaubt. Zugrunde liegt hier der Gedanke eines *Internet of Things*, das alle erforderlichen Informationen für Engineering und Ablaufumgebung bezüglich der Automatisierungskomponenten herstellerübergreifend zur Verfügung stellt.

7 Summary

At the beginning of this thesis external factors are considered which have effect on manufacturing companies in a globalized world. Especially for industrial production in high-wage countries there is a permanent need to improve and increase efficiency in the planning, implementation and operation of automated production systems in order to survive in international competition.

The technical state of the art is described in the second chapter. This contains engineering standards as well as methodology and concepts in the areas of structuring and problem solving, modeling as well as specification and programming standards. After that the need for action is discussed. The cost structure of investment costs in the field of control technology (see figure 23) clearly shows that approximately 50% of the total cost is spent on planning and configuration efforts for hardware and software. Therefore a major goal is to achieve improvements in this area. Hence a concept should be developed and implemented, which improves the exchange of information between engineering tools and control level devices (vertical IT integration) as well as between different engineering tools (horizontal IT integration) by using a semantic description of automated production systems.

Based on existing standards and norms, the development of a concept for the semantic description of automation systems is presented in the third chapter. The concept has to be adapted to existing software tools and hardware architectures. As a result a methodology was defined to describe the structure and behavior of automated production systems. An important advantage is that the description of the behavior of an automated production system can be distributed along the assembly structure of the system. This helps with respect to functional component replacement and a distributed execution of the control program can be achieved. This concept is also modeled as an UML class diagram. The UML class diagram defines specific structures of the concept such as RoleTypes (see section 3.1.2) and ComponentTypes (see chapter 3.1.3) by using object oriented mechanisms such as inheritance hierarchies.

The following fourth chapter describes the implementation of the concept which takes existing engineering tools (NX / MCD and Process Simulate) into account. On the control level the concept was implemented by using a runtime engine which was enhanced to be used as a soft plc (see section 4.4). For the technical realization of a semantic information model, descriptors such as object types, data types and reference types were defined and are provided by the IEC standard OPC UA (see section 2.1.4). OPC UA can be used both in engineering as well as on the control and field level.

As a result engineering information can be deployed to information models on the control and field level which provides new opportunities and uses. Thus a web-based

visualization can be automatically generated by using the OPC UA information model. The concept behind that was described and a prototype was implemented.

The resulting potential of the concept and the prototypical implementation are described in chapter five. Basically the potential is a result of supplying additional information from the engineering to the control and field level which can be used to implement further functionalities. Using OPC UA attached to engineering tools as well as on the control and field level helped to provide a technical homogeneous implementation which integrates vertically between engineering tools and control hardware.

Achieved research results

This thesis shows how to integrate existing concepts, methodologies and standards to a technically-adapted solution, which realizes a horizontal IT integration of various engineering software products as well as the vertical IT integration between engineering software and control level. The obtained research results can be differentiated on the basis of the working areas of the prototype implementation. This includes the specification of a description method that can be implemented both in engineering tools as well as in control and field level devices, which has been demonstrated as a prototype. The developed description methodology includes semantic descriptors for structure and behavior of automated production systems. The prototype implementation couples a runtime environment (control level) with a CAD system such as NX. Therefore processes on the control level can be assigned to the assembly structure of the CAD system. By using the IEC standard OPC UA a technical solution is available for providing an semantic information model which can be integrated both in engineering software tools and in the control and field level devices. Thus the technical feasibility of a star-shaped, semantic linkage between engineering tools for planning and configuration and automation components on the control and field level during the planning, commissioning and operation phase of a plant was demonstrated.

Outlook on further work

The integration of this concept in other engineering tools is planned. Therefore a shared information model has to be developed. Because of the different concepts, Objectives and data representations of various engineering tools a careful mapping of the object representations of the individual applications in the OPC UA information model is necessary. A distinction has to be made between universal and tool-specific information. Tool-specific information has to be stored by specialization of the affected object types.

Today's engineering concepts with the goal of an integrated, consistent use of software tools are usually based on a central PDM system, which is used as an information hub during the engineering process of mechatronic products. The concept presented in this thesis and its prototype implementation are focused on the development of automated production systems and are particularly suitable for the interaction between control hardware and engineering software. Especially during the commissioning phase changes to planning and configuration data have to be made due to unknown or overlooked constraints. In this environment, semantic information models offer the possibility to accompany changes to the configuration data to the current production system in the field. After a successful commissioning the transfer of semantic information models to the engineering project server or PDM systems is possible. Suitable interaction mechanisms have to be developed yet.

For a further improvement of the usability and applicability of the prototype a library for automation components should be implemented. The library should contain automation components which are described with respect to their geometrical structure, their function within the meaning of the process (mechanical, electrical, thermal, chemical, etc.) as well as its electrical function interfaces. Here underlies the idea of Internet of Things, which provides all the information needed for engineering and runtime environment with respect to vendor-independent automation components.

8 Abkürzungsverzeichnis

CAE: Computer-Aided Engineering

cPDM: collaborative Product Development Management

E/A: digitale/analoge Ein- bzw. Ausgänge zum Beispiel einer SPS

E-CAD: Elektrotechnische computergestützte Konstruktion

FA: Fertigungsautomatisierung

FB: Funktionsblock

GUID: Global Unique Identifier

HMI: Human-Machine-Interface

HTTP: HyperText Transfer Protocol

HTTPS: HyperText Transfer Protocol Secure

IAP: Interaktionspunkt

IBN: Inbetriebnahme

IPC: Interprocess communication

M2M: Machine-to-Machine

MC: Motion Control

M-CAD: Mechanische computergestützte Konstruktion

MCD: Mechatronics Concept Designer, ein Erweiterungsmodul von NX

NX: CAD-Programm der Firma Siemens

OLE: Object Linking and Embedding

OPC UA: OLE for Process Control Unified Architecture

PA: Prozessautomatisierung

RTE: Die RunTimeEngine ist eine Ablaufumgebung für verschaltete Servicebausteine mit eine Erweiterung für zyklische, zustandsorientierte Ablaufsequenzen.

SPS: Speicherprogrammierbare Steuerung

TCO: Total Cost of Ownership

TCP/IP: Transmission Control Protocol / Internet Protocol

UDP: User Datagram Protocol

vIBN: virtuelle Inbetriebnahme

9 Eingesetzte Software

- NX/MCD 8.5/9.0/10.0beta/10.0 (Siemens)
- Simatic Net (Siemens)
- SIMATIC STEP 7 PROFESSIONAL V13 (Siemens)
- Simit 8.0 Ultimate (Siemens)
- ProcessSimulate 11.1 (Siemens)
- Visual Studio 2005/2008/2010/2012 (Microsoft)
- OPC UA C++ Server SDK (Unified Automation)
- UAExpert (Unified Automation)
- UAModeler (Unified Automation)
- UModel Professional Edition 2014 sp1 (Altova)
- Google Chrome V8 JavaScript Engine

10 Literatur

[1] ABELE, E.; REINHART, G.: Zukunft der Produktion: Herausforderungen, Forschungsfelder, Chancen. München: Hanser, Carl, 2011

[2] FRANKE, J.; MERHOF, J.; HOPFENSITZ, S.: Einsatz von dezentralen Multiagentensystemen: Komplexitätsbeherrschung bei der Produktionsplanung und -steuerung. In: Zeitschrift für wirtschaftlichen Fabrikbetrieb 105 (2010), Nr. 12, S. 1075–1078

[3] FELDMANN, K.; BROSSOG, M.; MERHOF, J.; MICHL, M.: Dreidimensionale Planung, Simulation und Überwachung von Produktionssystemen mit VRML. In: Zeitschrift für wirtschaftlichen Fabrikbetrieb (2008), Nr. 10, S. 676–681. http://www.zwf-online.de/ZW101342

[4] MERHOF, J.: Adaptive echtzeitfähige und vernetzte Produktion (Innovation Forum Embedded Systems 2012). Konferenzzentrum München, 27.04.2012. http://www.bicc-net.de/workspace/uploads/subfeatures/downloads/franke_merhof-4fab7ca81466e.pdf – Überprüfungsdatum 2015-06-23

[5] BROSSOG, M.; HARTMANN, C.; KOHL, J.; MERHOF, J., et al.: Prozessflexible Steuerung von Industrierobotern: Integration von optischer Sensorik und Energiemesstechnik in eine automatisierte Prüfzelle. In: WT (Werkstattstechnik Online) 104 (2014), Nr. 9, S. 535–540

[6] FRANKE, J.; MERHOF, J.; FISCHER, C.; RISCH, F.: Intelligente Steuerungskonzepte für wandlungsfähige Produktionssysteme. In: Industrie Management 26 (2010), S. 61–64

[7] Norm DIN 2330:07.2013, Begriffe und Benennungen

[8] Norm DIN 2342:08.2011, Begriffe der Terminologielehre

[9] Norm DIN 2331:04.1980, Begriffssysteme und ihre Darstellung

[10] Norm DIN EN 61512-1:01.2000, Chargenorientierte Fahrweise Teil 1: Modelle und Terminologie

[11] Norm DIN EN 62264-1:09.2012, Integration von Unternehmensführungs- und Leitsystemen – Teil 1: Modelle und Terminologie

[12] STACHOWIAK, H.: Allgemeine Modelltheorie. Wien [u.a.]: Springer, 1973

[13] MERHOF, J.: Modellerstellung. In: FELDMANN, K.; SCHÖPPNER, V.; SPUR, G. (Hrsg.): Handbuch Fügen, Handhaben, Montieren. München: Carl Hanser Verlag GmbH & Co. KG, 2014 (Handbuch der Fertigungstechnik), S. 608–610

[14] MERHOF, J.; LEBRECHT, H.; MICHL, M.; FRANKE, J.: DES within semiconductor manufacturing. In: SEMATECH (Hrsg.): ISMI Symposium on Manufacturing Effectiveness. Austin, TX: Curran Associates Inc., 2011, S. 596–606

[15] MERHOF, J.: Using DES for RoI calculations within semiconductor manufacturing (Advanced Equipment Control and Advanced Process Control Symposium Asia (AEC/APC)). National Center of Sciences Building, Tokyo, Japan, 07.11.2011. http://publica.fraunhofer.de/eprints/urn_nbn_de_0011-n-1959572.pdf – Überprüfungsdatum 2015-06-23

[16] Oliver Thomas: Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation. Saarbrücken, 2005 (184)

[17] Susanne Strahringer: Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips, Modellierung '98. In: CEUR Workshop Proceedings.

[18] EHRLENSPIEL, K.; MEERKAMM, H.: Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit. 5., überarb. und erw. Aufl. München: Hanser, 2013

[19] Norm VDI 2221:Mai 1993, Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte

[20] DAENZER, W. F.; BÜCHEL, A.; BECKER, M.; HABERFELLNER, R., et al.: Systems engineering: Methodik und Praxis. 8. Aufl. Zürich: Verl. Industrielle Organisation, 1994

[21] EHRLENSPIEL, K.: Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit. 4. Aufl. München, Wien: Hanser, 2009

[22] RUDE, S.: Wissensbasiertes Konstruieren. Als Ms. gedr. Aachen: Shaker, 1998 (Berichte aus dem Maschinenbau)

[23] PONN, J. C.; LINDEMANN, U.: Konzeptentwicklung und Gestaltung technischer Produkte. 2. Aufl. Berlin, Heidelberg: Springer, 2011 (VDI-Buch)

[24] VDI-Richtlinie 2206:2004-06, Entwicklungsmethodik für mechatronische Systeme

[25] Norm ISO/IEC 19505-2:20.04.2012, Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 2: Superstructure

[26] Norm ISO/IEC 19505-1:20.04.2012, Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 1: Infrastructure

[27] 06/2012, OMG Systems Modeling Language (OMG SysML™)

[28] DRAHT, R.: CAEX 2.0 - Ein Überblick über aktuelle Entwicklungen. In: 5. Symposium Informationstechnologien für Entwicklung und Produktion. Aachen, 03-04. April 2008.

[29] DRATH, R.: Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA. Heidelberg, New York: Springer, 2010

[30] Norm IEC 62424:2008-08, Representation of process control engineering-- requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools

[31] BARNES, M.; FINCH, E. L.: COLLADA – Digital Asset Schema Release 1.5.0: Specification. 04.2008

[32] XML-Schema von COLLADA v1.5.0. https://www.khronos.org/files/collada_schema_1_5 – Überprüfungsdatum 2015-06-18

[33] AutomationML Core Presentation at HMI2009: The Glue for Seamless Automation Engineering. Hannover Messe, 2009. https://www.automationml.org/o.red/uploads/dateien/1317722297-AutomationML_Core_HMI2009.pdf – Überprüfungsdatum 2015-07-04

[34] Whitepaper AutomationML: Part 1 - Automation Architecture. 2010

[35] GARCIA, A. A.; DRATH, R. Dr.: AutomationML verbindet Werkzeuge der Fertigungsplanung: Hintergründe und Ziel. 2007

[36] CZICHOS, H.: Mechatronik: Grundlagen und Anwendungen technischer Systeme; mit 12 Tabellen. 2., aktualisierte und erw. Aufl. Wiesbaden: Vieweg + Teubner, 2008 (Studium)

[37] Norm IEC/TR 62541-1:18.02.2010, OPC Unified Architecture - Part 1: Overview and Concepts

[38] Norm-Entwurf DIN EN 62541-2:2008-12:12-2008, OPC Unified Architecture - Teil 2: Modell für die IT-Sicherheit (IEC 65E/93/CDV:2008); Englische Fassung FprEN 62541-2:2008

[39] Norm-Entwurf DIN EN 62541-3:2014-11:11-2014, OPC Unified Architecture - Teil 3: Adressraummodell (IEC 65E/374/CDV:2014); Englische Fassung FprEN 62541-3:2014

[40] Norm-Entwurf DIN EN 62541-4:2014-11:11-2014, OPC Unified Architecture - Teil 4: Dienste (IEC 65E/375/CDV:2014); Englische Fassung FprEN 62541-4:2014

[41] Norm-Entwurf DIN EN 62541-5:2014-11:11.2014, OPC Unified Architecture - Teil 5: Informationsmodell (IEC 65E/376/CDV:2014); Englische Fassung FprEN 62541-5:2014

[42] Norm-Entwurf DIN EN 62541-6:2014-11:11.2014, OPC Unified Architecture - Teil 6: Protokollabbildungen (IEC 65E/377/CDV:2014); Englische Fassung FprEN 62541-6:2014

[43] Norm-Entwurf DIN EN 62541-7:2014-11:11.2014, OPC Unified Architecture - Teil 7: Profile (IEC 65E/378/CDV:2014); Englische Fassung FprEN 62541-7:2014

[44] Norm-Entwurf DIN EN 62541-8:2014-11:11.2014, OPC Unified Architecture - Teil 8: Zugriff auf Automatisierungsdaten (IEC 65E/381/CDV:2014); Englische Fassung FprEN 62541-8:2014

[45] Norm-Entwurf DIN EN 62541-9:2014-11:11.2014, OPC Unified Architecture - Teil 9: Alarme und Zustände (IEC 65E/382/CDV:2014); Englische Fassung FprEN 62541-9:2014

[46] Norm-Entwurf DIN EN 62541-10:2014-11:11.2014, OPC Unified Architecture - Teil 10: Programme (IEC 65E/383/CDV:2014); Englische Fassung FprEN 62541-10:2014

[47] Norm-Entwurf DIN EN 62541-11:2014-08:08.2014, OPC Unified Architecture - Teil 11: Zugang zu historischen Daten (IEC 65E/380/CDV:2014); Englische Fassung FprEN 62541-11:2014

[48] Norm-Entwurf DIN EN 62541-13:2014-08:08.2014, OPC Unified Architecture - Teil 13: Aggregation von Daten (IEC 65E/379/CDV:2014); Englische Fassung FprEN 62541-13:2014

[49] Unified Automation GmbH: Introduction to OPC UA. http://documentation.unified-automation.com/uasdkcpp/1.0.0/L2OpcUaOverview.html – Überprüfungsdatum 2016-05-05

[50] Norm:2006, The International System of Units (SI)

[51] DAMM, M.; LEITNER, S.-H.; MAHNKE, W.: OPC Unified Architecture. Berlin Heidelberg: Springer-Verlag Berlin Heidelberg, 2009

[52] H. L. Gantt: Organizing for work. New York: Hartcourt, Brace and Howe, 1919

[53] Wallace Clark: The Gantt Chart: A working tool of management. New York: The Ronald Press Company, 1922

[54] NOOSTEN, D.: Netzplantechnik: Grundlagen und Anwendung im Bauprojektmanagement. Wiesbaden: Imprint: Springer Vieweg, 2013

[55] DANGELMAIER, W.: Theorie der Produktionsplanung und -steuerung: Im Sommer keine Kirschpralinen? Dordrecht, New York: Springer, 2009 (VDI-Buch)

[56] GRAFCET specification language for sequential function charts: Langage de spécification GRAFCET pour diagrammes fonctionnels en séquence. Ed. 3.0. Geneva: International Electrotechnical Commission, 2013 (International standard Norme internationale IEC 60848)

[57] GRAFCET. 1. Aufl. Troisdorf: Bildungsverl. EINS, 2008

[58] KOMAR, E.: Digitaltechnik: Teil 2. https://www.fbi.h-da.de/fileadmin/personal/e.komar/public_html/DGT-Skript-Teil2.PDF – Überprüfungsdatum 2015-06-26

[59] Siemens AG: SIMATIC S7-GRAPH: Schrittkettenprogrammierung. http://w3.siemens.com/mcms/simatic-controller-software/de/step7/simatic-s7-graph/seiten/default.aspx – Überprüfungsdatum 2015-06-27

[60] Norm DIN EN 61131-1:01.03.2004, Speicherprogrammierbare Steuerungen Teil 1: Allgemeine Informationen (IEC 611311:2003); Deutsche Fassung EN 611311:2003

[61] Norm DIN EN 61131-2:2008-04; VDE 0411-500:2008-04:04.2008, Programmable controllers - Part 2: Equipment requirements and tests (IEC 61131-2:2007); German version EN 61131-2:2007

[62] Norm DIN EN 61131-3:12.2003, Speicherprogrammierbare Steuerungen - Teil 3: Programmiersprachen

[63] ASPERN, J. v.: SPS-Grundlagen: Aufbau, Programmierung (IEC 61131, S7), Simulation, Internet, Sicherheit. Heidelberg: Hüthig, 2005

[64] Technical Report IEC/TR 61131-4:07.2004, Programmable controllers - Part 4: User guidelines

[65] Norm DIN EN 61131-5:2001-11:11.2001, Programmable controllers - Part 5: Communications (IEC 61131-5:2000); German version EN 61131-5:2001

[66] Norm DIN EN 61508-1:2011-02; VDE 0803-1:2011-02:02.2011, Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements (IEC 61508-1:2010); German version EN 61508-1:2010

[67] Norm DIN EN 62061:2013-09; VDE 0113-50:2013-09:09.2013, Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems (IEC 62061:2005 + A1:2012); German version EN 62061:2005 + A1:2013

[68] Norm DIN EN 61131-6:2013-10; VDE 0411-506:2013-10:10.2013, Programmable controllers - Part 6: Functional safety (IEC 61131-6:2012); German version EN 61131-6:2012

[69] Norm DIN EN 61131-7:2001-11:11.2001, Programmable controllers - Part 7: Fuzzy control programming (IEC 61131-7:2000); German version EN 61131-7:2000

[70] Technical Report IEC/TR 61131-8:29.09.2003, Programmable controllers - Part 8: Guidelines for the application and implementation of programming languages

[71] JOHN, K. H.; TIEGELKAMP, M.: SPS-Programmierung mit IEC 61131-3: Konzepte und Programmiersprachen, Anforderungen an Programmiersysteme, Entscheidungshilfen. 4. Aufl. Berlin, Heidelberg: Springer-Verlag, 2009

[72] GAUSEMEIER, J.: Domänenübergreifende Vorgehensmodelle. http://www.viprosim.de/fileadmin/Fachbeitrag/Beitrag_Vorgehensmodelle_Gausemeier_2008-12-18.pdf – Überprüfungsdatum 2015-06-26

[73] GAUSEMEIER, J.; KAISER, L.; POOK, S.; NYßEN, A.; TERFLOTH, A.: Rechnerunterstützte Modellierung der Prinziplösung mechatronischer Systeme. 2010

[74] GAUSEMEIER, J.; TRÄCHTLER, A.; SCHÄFER, W.: Semantische Technologien im Entwurf mechatronischer Systeme: Effektiver Austausch von Lösungswissen in Branchenwertschöpfungsketten. München: Hanser, 2014

[75] HOLZ, R.: Zustandsmaschinen mit CODESYS: Spracheigenschaften geschickt genutzt (CODESYS Users´ Conference Deutschland/Schweiz 2014). Egerkingen/Schweiz, 17.03.2014. http://www.users-conference.com/fileadmin/UsersConference/Praesentationen/02_Zustandsmaschinen_mit_CODESYS.pdf – Überprüfungsdatum 2015-02-12

[76] HOPPE, S.: eXtended Automation Engineering: C++ zur komfortablen Programmierung nutzen. http://www.polyscope.ch/site/assets/files/20288/ps1611_32_34.pdf – Überprüfungsdatum 2015-06-26

[77] ABU-KHALAF, M.; CHEN, R.: Reglerentwurf am Beispiel des Gelenkvierecks: PID-Entwurf schnell und leicht gemacht. http://www.polyscope.ch/site/assets/files/20283/ps1611_16_19.pdf – Überprüfungsdatum 2015-06-26

[78] MERHOF, J.; MICHL, M.; MAINKA, M.; FRANKE, J.: Investitionsbewertung: Eine Methodik zur Bewertung und zum Vergleich von Investitionen. In: Industrie Management 28 (2012), Nr. 4, S. 20–24

[79] KOITZSCH, M.; MERHOF, J.; MICHL, M.; NOLL, H., et al.: Implementing Virtual Metrology into Semiconductor Production Processes - An Investment Assessment. In: Proceedings of the 2011 Winter Simulation Conference. Phoenix, AZ, 2011, S. 2022–2033

[80] AUTOMATISIERUNGSINITIATIVE DEUTSCHER AUTOMOBILHERSTELLER (Aida): Analyse Investitionskostenstruktur Steuerungstechnik und Robotik am Beispiel Rohbau. 2005

[81] HIRZLE, A.: AutomationML - ein Überblick. In: SPS Magazin (Hrsg.): AutomationML–Fachexperten erklären das Format, S. 3–5

[82] NYHUIS, P. (Hrsg.): Wandlungsfähige Produktionssysteme: Heute die Industrie von morgen gestalten. Garbsen: PZH, Produktionstechn. Zentrum, 2008

[83] WESTKÄMPER, E.: Wandlungsfähige Produktionsunternehmen: Das Stuttgarter Unternehmensmodell. Berlin: Springer, 2009 (SpringerLink: Bücher)

[84] Peter Nyhuis, Tobias Heinen, Michael Brieke: Adequate and economic factory transformability and the effects on logistical performance. In: ZÄH, M. F. (Hrsg.): 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2007), Toronto, Ontario, Canada, 22 - 24 July 2007. Windsor, 2007, S. 286–307

[85] International Standard Organization: ISO/IEC 9126: Information technology - software product evaluation - quality characteristics and guidelines for their use. Geneve, 1991

[86] MERHOF, J.; FRANKE, J.: Cutting-edge engineering - an innovative approach to better support the engineering and the life cycle of automated production systems. In: XU, D. (Hrsg.): Proceedings of Manufacturing Engineering and Technology for Manufacturing Growth 2012, 2012, S. 445–450

[87] Siemens AG: SIMATIC STEP 7 Basic V11.0 SP1: Ausdruck der Onlinehilfe. https://support.industry.siemens.com/cs/attachments/53094945/step_7_basic_V11_0_de_de-DE.pdf?download=true – Überprüfungsdatum 2015-06-27

[88] Norm VDI 2860:1990-05:1990-05, Handhabungsfunktionen, Handhabungseinrichtungen

[89] Norm DIN 8580:2003-09:2003-09, Fertigungsverfahren - Begriffe, Einteilung

[90] MICHL, M.; FISCHER, C.; MERHOF, J.; FRANKE, J.: Comprehensive Support of Technical Diagnosis by Means of Web Technologies. In: 7th International Conference on Digital Enterprise Technology (DET). Athens, 2011, S. 73–82

[91] KROLL, J.: AutomationML und OPC UA: gegenseitiges »Verständnis«. http://www.elektroniknet.de/automation/m2m/artikel/103503/ – Überprüfungsdatum 2015-06-27

[92] Norm DIN EN 61499-1:12.2011, Verteilte Funktionsbausteine für die Automatisierungstechnik – Teil 1: Architektur

[93] VDI-Richtlinie 3695, Blatt 1:2010-11, Engineering von Anlagen - Evaluieren und optimieren des Engineerings - Grundlagen und Vorgehensweise

[94] VDI-Richtlinie 3695, Blatt 2:2010-11, Engineering von Anlagen - Evaluieren und optimieren des Engineerings - Themenfeld Prozesse

[95] VDI-Richtlinie 3695, Blatt 3:2010-12, Engineering von Anlagen - Evaluieren und optimieren des Engineerings - Themenfeld Methoden

[96] VDI-Richtlinie 3695, Blatt 4:2010-12, Engineering von Anlagen - Evaluieren und optimieren des Engineerings - Themenfeld Hilfsmittel

[97] Norm VDI 4499 Blatt1:2008, Digitale Fabrik - Grundlagen

[98] GRAF, R.; MERHOF, J.: Verfahren zur Steuerung einer Maschine und Vorrichtung zur Durchführung des Verfahrens. Siemens AG. Anmeldenr. PCT/EP2015/054816, Europäisches Patentamt. Den Haag. Niederlande

[99] SCHWABL, F.: Statistische Mechanik: Mit 26 Tabellen und 186 Aufgaben. 3., aktualisierte Aufl. Berlin [u.a.]: Springer, 2006 (Springer-Lehrbuch)

[100] SCHWABL, F.: Quantum mechanics. 4th ed. Berlin, New York: Springer, 2007

[101] STRAUMANN, N.: Quantenmechanik: Ein Grundkurs über nichtrelativistische Quantentheorie. 2., überarb. und aktual. Aufl. Berlin: Springer Berlin, 2013 (Springer Lehrbuch)

[102] SCHWINDT, J.-M.: Tutorium Quantenmechanik: Von einem erfahrenen Tutor - für Physik- und Mathematikstudenten. Berlin, Heidelberg: Imprint: Springer Spektrum, 2013 (SpringerLink: Bücher)

11 Anhang

11.1 Auflistung der betrachteten Standards und Normen

Norm	Titel
DIN 2330	Begriffe und Benennungen – Allgemeine Grundsätze [7]
DIN 2331	Begriffssysteme und ihre Darstellung [9]
DIN 2342	Begriffe der Terminologielehre [8]
DIN 28000-3	Chemischer Apparatebau – Dokumentation im Lebensweg von Prozessanlagen – Teil 3: Fließschemata und Anlagenkennzeichnung
DIN EN 61131-3	Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen [62]
DIN EN 61499-1	Funktionsbausteine für industrielle Leitsysteme – Teil 1: Architektur [92]
DIN EN 61512-1	Chargenorientierte Fahrweise Teil 1: Modelle und Terminologie [10]
DIN EN 61512-2	Chargenorientierte Fahrweise Teil 2: Datenstrukturen und Leitfaden für Sprachen
DIN EN 61512-3	Chargenorientierte Fahrweise – Teil 3: Modelle und Darstellungen von Verfahrens- und Werksrezepten
DIN EN 61512-4	Chargenorientierte Fahrweise – Teil 4: Aufzeichnungen zur Chargenproduktion
DIN EN 61850-7-1	Kommunikationsnetze und -systeme in Stationen – Teil 7-1: Grundlegende Kommunikationsstruktur für stations- und feldbezogene Ausrüstung; Grundsätze und Modelle
DIN EN 62264-1	Integration von Unternehmensführungs- und Leitsystemen – Teil 1: Modelle und Terminologie [11]
DIN EN 62264-2	Integration von Unternehmensführungs- und Leitsystemen – Teil 2: Attribute des Objektmodells
DIN EN 62264-3	Integration von Unternehmens-EDV und Leitsystemen – Teil 3: Aktivitätsmodelle für das operative Produktionsmanagement
DIN EN 62264-5	Integration von Unternehmensführungs- und Leitsystemen – Teil 5: Transaktionen zwischen Geschäftsabläufen und Produktionssteuerung

DIN EN 81346-1	Industrielle Systeme, Anlagen und Ausrüstungen und Industrieprodukte – Strukturierungsprinzipien und Referenzkennzeichnung – Teil 1: Allgemeine Regeln
DIN EN 81346-2	Industrielle Systeme, Anlagen und Ausrüstungen und Industrieprodukte – Strukturierungsprinzipien und Referenzkennzeichnung – Teil 2: Klassifizierung von Objekten und Kennbuchstaben von Klassen
IEC TR 62794	Technical Report: Industrial process measurement, control and automation – Reference model for representation of production facilities (digital factory)
ISO 1087-1	Terminology work – Vocabulary – Part 1: Theory and application
ISO TR 9007	Information processing systems; concepts and terminology for the conceptual schema and the information base
ISO 9126	Information technology – software product evaluation – quality characteristics and guidelines for their use
VDI 2206	Entwicklungsmethodik für mechatronische Systeme [24]
VDI 3681	Einordnung und Bewertung von Beschreibungsmitteln aus der Automatisierungstechnik
VDI 3695 Blatt 1	Engineering von Anlagen - Evaluieren und Optimieren des Engineerings - Grundlagen und Vorgehensweise [93]
VDI 3695 Blatt 2	Engineering von Anlagen - Evaluieren und Optimieren des Engineerings - Themenfeld Prozesse [94]
VDI 3695 Blatt 3	Engineering von Anlagen - Evaluieren und Optimieren des Engineerings - Themenfeld Methoden [95]
VDI 3695 Blatt 4	Engineering von Anlagen - Evaluieren und Optimieren des Engineerings - Themenfeld Hilfsmittel [96]
VDI 3695 Blatt 5	Engineering von Anlagen - Evaluieren und Optimieren des Engineerings - Themenfeld Aufbauorganisation
VDI 4499 Blatt 1	Digitale Fabrik – Grundlagen [97]
VDI 4499 Blatt 2	Digitale Fabrik - Digitaler Fabrikbetrieb
VDI 4499 Blatt 4	Digitale Fabrik - Ergonomische Abbildung des Menschen in der Digitalen Fabrik

11.2 Sprachelemente von CAEX

<CAEXFile> ist in einem CAEX-Dokument der Root-Knoten, den es nur einmal geben darf.

<InstanceHierarchy> ist der Root-Knoten für eine beliebige Anzahl an geschachtelten Instanzen des Typs <InternalElement>.

<InternalElement> ist ein Repräsentant eines logischen oder realen Anlagenobjekts, das bestimmte Attribute und Schnittstellen haben kann.

<SystemUnitClass> definiert Bibliotheksobjekte als Typ realer oder logischer Anlagenobjekte und kann vom Aufbau identisch mit einem <InternalElement> abgebildet werden. Somit können Bibliotheksobjekte im Instanzbaum instanziiert bzw. Instanzen als Bibliotheksobjekt übernommen werden.

<RoleClass> abstrahiert die technische Realisierung von Anlagenobjekten und beschreibt grundlegende Funktionalitäten einer Gruppe von Anlagenobjekten.

<InterfaceClass> beschreibt Schnittstellen zwischen Entitäten. Diese Schnittstellen können mechanische Abhängigkeiten (zum Beispiel Flansch verbindet Roboter mit Endeffektor) oder aber auch andere Relationen (zum Beispiel Signal_1 wird zwischen Objekt A und Objekt B ausgetauscht) darstellen.

<SystemUnitClassLib> ist eine Bibliothek mit einer Ansammlung beliebig vieler <SystemUnitClass>-Objekte. In einem CAEX-Dokument darf es beliebig viele <SystemUnitClassLib>-Bibliotheken geben.

<RoleClassLib> ist eine Bibliothek mit einer Ansammlung beliebig vieler <RoleClass>-Objekte. In einem CAEX-Dokument darf es beliebig viele < RoleClassLib >-Bibliotheken geben.

<InterfaceClassLib> ist eine Bibliothek mit einer Ansammlung beliebig vieler <InterfaceClass>-Objekte. In einem CAEX-Dokument darf es beliebig viele < InterfaceClassLib >-Bibliotheken geben.

11.3 Erfindung zur quantenmechanischen Formulierung von Ablaufsequenzen in Automatisierungssystemen

Anknüpfend an die Arbeiten im Bereich der Ablaufumgebung zur Ausführung von Ablaufsequenzen (siehe Kapitel 4.4) wurde ein alternatives Konzept zur Beschreibung von Ablaufsequenzen und deren Ausführung entwickelt und als Patent beim Internationalen Patentamt [98] angemeldet.

Inhalt dieser Erfindung ist die Verwendung von quantenmechanischen Notationen und Berechnungsverfahren zur Beschreibung und Ausführung von Prozessabläufen in einem automatisierten Produktionssystem. Als Ausgangslage dient die Beschreibung des Prozessablaufs in Vektorschreibweise (siehe Abbildung 42) der bereits vorgestellten Handhabungseinheit.

Da die klassische Mechanik Vorgänge in kleinsten Systemen mit einzelnen Teilchen (Atome, Elektronen, …) nicht beschreiben konnte, wurde die Quantenmechanik entwickelt. Weitere Informationen können den Quellen [99; 100; 101; 102] entnommen werden. In der Quantenmechanik haben sich die Wellenmechanik von Schrödinger sowie die Matrizenmechanik von Heisenberg als grundlegende Modelle entwickelt.

Die Wellenmechanik nach Schrödinger sagt aus, dass die Entwicklung des Zustands über die Zeit durch die Schrödinger-Gleichung (zeitabhängige Wellenfunktion) bestimmt ist. Zustandsänderungen werden mit Hilfe von Operatoren ausgedrückt. Operatoren beziehen sich auf die gemessenen Observablen und sind in der Regel zeitunabhängig (abgesehen vom Hamilton-Operator H(t), der als zeitabhängiger Operator Verwendung findet, wenn die zu messende Observable von außen aktiv verändert wird).

Nach Heisenbergscher Betrachtungsweise der Quantenmechanik ändert sich der Zustand eines Systems über die Zeit nicht. Stattdessen wird die Dynamik des Systems mit Hilfe zeitabhängiger Operatoren (Matrizen) dargestellt.

Im Vergleich zu Schrödingers zeitabhängiger Wellenfunktion liegt die Matrizen-Notation von Heisenberg näher an der aktuellen Vektor-Notation des Prozessablaufs der Handhabungseinheit (siehe Abbildung 42). Die Analogie kann durch Gegenüberstellung (siehe Tabelle 8) dargestellt werden.

In beiden Sichtweisen wird ein Zustandsvektor des Systems als Überlagerung (Superposition) aller Eigenzustände verstanden. Da die einzelnen mechanischen Komponenten der betrachteten Handhabungseinheit unabhängig voneinander arbeiten, sind die Eigenvektoren der einzelnen Komponenten orthonormal. Bezüglich dieser Basis an Eigenvektoren ist die Matrixdarstellung eines Operators diagonal, wodurch sich ein Operator nur auf die ihm zugedachte Komponente auswirkt.

Tabelle 8: Gegenüberstellung *der Analogie-Paare zwischen Quantenmechanik nach Heisenberg und vektorbasierte Beschreibung der Steuerungslösung eines automatisierten Produktionssystems*

Quantenmechanik nach Heisenberg	Vektorbasierte Steuerungslösung
Zustandsvektor	Zustandsvektor
Observable (messbare Größe, z.B. Ort)	Beobachtbare Zustandsänderung (z.B. Verfahren eines Aktuators)
Operator	Beschreibung einer Zustandsänderung
Eigenwert	(End-)Position einer Observablen
Eigenzustand	Zustand des zur Observablen gehörigen Anlagenteils

Zustandsvektoren können entweder als echte Vektoren (siehe Abbildung 42) oder aber auch in verkürzter Form nach Dirac-Notation als $|\psi\rangle$ geschrieben werden. Operatoren werden als Großbuchstaben notiert und stellen letztendlich Matrizen dar, die auf Zustandsvektoren angewendet werden.

Eine Schrittkette wie in Abbildung 42 dargestellt, ist eine Aneinanderreihung von Zustandsübergängen (Operatoren) ausgehend von einem Anfangszustand $|\psi_A\rangle$ und endet in einem Endzustand $|\psi_E\rangle$.

Für die einzelnen Komponenten der Handhabungseinheit (X-Achse, Z-Achse, Rotationsachse, Backen- und Sauggreifer) lassen sich die Operatoren X, Z, R, B und S definieren, die noch jeweils unterteilt werden, um zwischen der Bewegungsrichtung zu differenzieren. So beschreibt zum Beispiel der Operator X^- die Bewegung in die rechte Endlage (x_0), der Operator X^+ die Bewegung in die linke Endlage (x_1) der X-Achse. Analog wird mit den anderen Achsen verfahren. Bei der Komponente Backengreifer wird mit den Operationen B^+ und B^- „Greifen“ und „Loslassen“ dargestellt. Der Sauggreifer umfasst drei mögliche Zustände (0=„aus“; 1=„Vakuum“; 2=„Überdruck“). Der Operator S^+ transformiert den Ausgangszustand „aus“ in den Zustand „Vakuum“ oder aber auch den Ausgangszustand „Vakuum“ in den Zustand „Überdruck“. Der Operator S^- hingegen überführt den Ausgangszustand „Überdruck“ in den Zustand „Vakuum“ oder aber auch den Ausgangszustand „Vakuum“ in den Zustand „aus“.

Am Beispiel der X-Achse soll die Notation weiter verdeutlicht werden. Die X-Achse kann zwischen den beiden Positionen x_0 und x_1 bewegt werden. Der Zustandsvektor $|\psi_X\rangle$ kann damit die Eigenvektoren $|x_0\rangle$ und $|x_1\rangle$ haben. Die Operationen X^- und X^+ bewirken die Übergänge wie folgt:

$|x_0\rangle = X^- |x_1\rangle$ und $|x_1\rangle = X^+ |x_0\rangle$

Eine Anwendung der Operatoren auf die jeweils anderen Zustände bewirkt hingegen keine Zustandsänderung:

$$|x_0\rangle = X^- |x_0\rangle \quad \text{und} \quad |x_1\rangle = X^+ |x_1\rangle$$

Nach diesem Prinzip kann die gesamte Prozesskette aus Abbildung 42 beginnend beim Anfangszustand $|\psi_A\rangle$ (Vektor ganz links in der Abbildung) dargestellt werden:

$$|\psi_E\rangle = Z^- \bullet R^- \bullet S^- \bullet S^- \bullet S^+ \bullet B^- \bullet R^+ \bullet Z^+ \bullet X^- \bullet Z^- \bullet B^+ \bullet S^+ \bullet Z^+ |\psi_A\rangle$$

Die Rückführung der Anlage ausgehend vom Endzustand $|\psi_E\rangle$ zum Anfangszustand $|\psi_A\rangle$ stellt sich hingegen deutlich einfacher dar:

$$|\psi_A\rangle = X^+ |\psi_E\rangle$$

Die Initialisierungssequenz ausgehend vom unbekannten Zustand $\psi_U\rangle$ lässt sich wie folgt angegeben:

$$|\psi_A\rangle = X^+ \bullet Z^- \bullet R^- \bullet B^- \bullet S^- \bullet S^- |\psi_U\rangle$$

Diese Sequenz überführt in der definierten Reihenfolge einen beliebigen Zustandsvektor in den definierten Anfangsvektor (siehe Abbildung 42, erster Vektor von links).

Die Matrixdarstellung lässt sich mittels eines Beispiels erläutern. Die Komponente x-Achse hat zwei Endpunkte x_0 und x_1, die auch die möglichen Zustände dieses Teilsystem darstellen. Somit ergibt sich der Zustandsvektor dieser Komponente in Vektordarstellung zu:

$$|\psi_X\rangle = \begin{pmatrix} x_o \\ x_1 \end{pmatrix} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

Die Operatoren X^+ und X^- lassen sich folglich als Matrizen schreiben:

$$X^+ = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad X^- = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

Wendet man diese Operatoren mittels Matrixmultiplikation auf einen gültigen Zustand $|\psi_X\rangle$ an, so wird dieser umgeschaltet oder bleibt unverändert, soweit sich das System bereits in dem geforderten Zustand befindet (wie in Abschnitt 3 beschrieben):

$$X^+ \bullet \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \quad X^+ \bullet \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \quad X^- \bullet \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad X^- \bullet \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix};$$

Analog lassen sich die Zustandsvektoren und Operatormatrizen für die anderen Komponenten erstellen.

Können Operatoren nebenläufig ausgeführt werden, ist eine Zusammenfassung dieser Operatoren in einer Matrix möglich, was eine gleichzeitige Transformation der Zustände nebenläufiger Komponenten erlaubt. In dem hier betrachteten Beispiel der

Handhabungseinheit sind die Operatoren X und Z nebenläufig, wodurch ein aggregierter Operator N wie folgt gebildet werden kann:

$$N^+ = \begin{pmatrix} X^+ & 0 & 0 \\ & 0 & 0 \\ 0 & 0 & Z^+ \\ 0 & 0 & \end{pmatrix}, \quad N^- = \begin{pmatrix} X^- & 0 & 0 \\ & 0 & 0 \\ 0 & 0 & Z^- \\ 0 & 0 & \end{pmatrix}$$

Ist die Ausführung von Operatoren von bestimmten Bedingungen (z.B. Einschaltverriegelungen) abhängig, kann dies ebenfalls mit der Matrixdarstellung umgesetzt werden.

Neben dem Zustandsvektor $\begin{pmatrix} x_o \\ x_1 \end{pmatrix}$ wird ein weiterer Zustandsvektor $\begin{pmatrix} b_o \\ b_1 \end{pmatrix} = \left\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$ eingeführt, wobei dem Zustand $\begin{pmatrix} b_o \\ b_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ die Bedeutung „Bedingung nicht erfüllt“ und dem Zustand $\begin{pmatrix} b_o \\ b_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ die Bedeutung „Bedingung erfüllt“ zugeordnet wird.

Aus den Zustandsvektoren $\begin{pmatrix} x_o \\ x_1 \end{pmatrix}$ sowie $\begin{pmatrix} b_o \\ b_1 \end{pmatrix}$ ergeben sich folgende vier Zustandsvektoren $\begin{pmatrix} x_o \\ x_1 \\ b_o \\ b_1 \end{pmatrix} = \left\{\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}\right\}$. Nun werden die Operatoren X'^+sowie X'^-gesucht, für die gilt:

$$X'^+ \bullet \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad X'^+ \bullet \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad X'^+ \bullet \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}; \; X'^+ \bullet \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix};$$

$$X'^- \bullet \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \; X'^- \bullet \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad X'^- \bullet \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}; \; X'^- \bullet \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix};$$

Um die Lösbarkeit dieser Anforderung immer zu gewährleisten, werden die Zustandsvektoren $\begin{pmatrix} x_o \\ x_1 \\ b_o \\ b_1 \end{pmatrix}$ in ihrer Dimension um einen Korrekturteil (grün eingefärbt) erweitert, wobei die Anzahl der Korrekturdimensionen des Zustandsvektors gleich der Anzahl der Zustandsvektoren ist. Somit ergibt sich ein erweiterter Zustands-

vektor $\begin{pmatrix} x_o \\ x_1 \\ b_o \\ b_1 \\ k_o \\ k_1 \\ k_2 \\ k_3 \end{pmatrix}$, wobei k_a den Korrekturdimensionen entspricht. Es resultieren die vier Zustandsvektoren $\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

Für diese um den Korrekturteil erweiterten Zustandsvektoren lassen sich nun immer Matrizen bilden, die die Anforderungen an den X'^{+} bzw. an den X'^{-} Operator erfüllen. Für das hier vorgestellte Beispiel kann für den Operator X'^{+} folgende Matrix angegeben werden:

$$X'^{+} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Für diese gilt nun:

$$X'^{+} \bullet \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X'^{+} \bullet \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad X'^{+} \bullet \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad X'^{+} \bullet \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Somit ist die Matrix X'^{+} in der Lage, das geforderte Transformationsverhalten in Abhängigkeit des Zustandes sowie unter Berücksichtigung weiterer Bedingungen zu realisieren. Hieraus lassen sich wie folgt Möglichkeiten für die Ausführung von Steuerungslösungen ableiten.

Für Automatisierungskomponenten, die bezüglich ihres eigenen Zustands voneinander unabhängig sind, kann mit Hilfe dieses Vorgehens eine Matrix erstellt werden, die alle Transformationen einer Ablaufsequenz eines Prozesses enthält. Dadurch kann ein System, das sich in einem beliebigen ordnungsgemäßen Zustand im Sinne des Prozessablaufs befindet, durch Anwendung der Matrix (linksseitige Multiplikation auf den Zustandsvektor des Systems) in den korrekten Nachfolgezustand des Systems überführt werden. Aufwändige Abfragen des Systemzustands sind nicht mehr erforderlich, was eine erhebliche Steigerung der Performance ermöglicht.

12 Lebenslauf

Persönliche Angaben

Name:	Jochen Merhof
Adresse:	Eschenweg 12 71131 Jettingen
Geburtsdatum:	08.06.1981
Geburtsort:	Erlangen
Staatsangehörigkeit:	Deutsch
Familienstand:	Verheiratet, 1 Sohn

Schulausbildung

1988 - 1992	Grundschule „Am Lichtenstein“, Pommelsbrunn
1992 - 2001	Paul-Pfinzing-Gymnasium Hersbruck Mathematisch-naturwissenschaftlicher Zweig
2001	Abitur

Studium

Oktober 2001	Beginn des Diplominformatikstudiums an der Friedrich-Alexander-Universität Erlangen-Nürnberg
August 2007	Abschluss des Informatikstudiums mit Auszeichnung

Berufliche Tätigkeit

Sept. bis Dez. 2007	Wissenschaftlicher Hilfsmitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik der Universität Erlangen-Nürnberg
Jan. 2008 bis Mai 2015	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungs-automatisierung und Produktionssystematik der Universität Erlangen-Nürnberg
Okt. 2010 bis Mai 2015	Externer Mitarbeiter in der Vorfeldentwicklung des Industry-Sektors von Siemens
Jan. 2013 bis Mai 2015	Berufung zum Akademischen Rat (auf Zeit)
Seit Juli 2015	Produktmanager bei Siemens Industry Software GmbH