

Fähigkeiten in der Produktionsautomatisierung



Konsolidierung des Konzepts aus Sicht des Maschinen- und Anlagenbaus mit dem Schwerpunkt OPC UA

Fähigkeiten in der Produktionsautomatisierung

**Konsolidierung des Konzepts aus Sicht des Maschinen-
und Anlagenbaus mit dem Schwerpunkt OPC UA**

Inhalt

Editorial	5
Vorwort	6
Management Summary	8
1 Einleitung und Motivation	9
1.1 Einleitung	9
1.2 Vereinfachte Lösungsfindung bei Automatisierungssystemen	11
1.3 Vereinfachtes Engineering durch Fähigkeiten	11
1.4 Fähigkeiten, Skills oder Capabilities?	13
2 Mehrwert durch den Einsatz von Fähigkeiten	15
2.1 Ziele aus Sicht des Maschinen- und Anlagenbaus:	15
2.1.1 Planungsphase	15
2.1.2 Inbetriebsetzung	18
2.2 Ziele aus Sicht der Endanwender	20
3 Modelle zur Beschreibung von Fähigkeiten	22
3.1 Einordnung der Capability- und Skill-Modelle	22
3.2 Allgemeines Metamodell für Capabilities und Skills	24
3.3 Umsetzung von Capabilities als Ontologie in OWL	24
3.4 Implementierung von Skills	28
3.5 Vorgehen beim Einsatz von Fähigkeiten	39
4 Beispiele für die Umsetzung von Skills mithilfe von OPC UA	43
4.1 Umsetzung von Skills am Beispiel der Montagetechnik	43
4.2 Umsetzung von Skills am Beispiel der Fertigungstechnik	46
5 Ausblick	48

Editorial



Johannes Olbort
Projektleiter II4IP,
Forschungskuratorium
Maschinenbau



Andreas Faath
Abteilungsleiter
Machine Information
Interoperability, VDMA

Der Maschinen- und Anlagenbau beschreitet in seiner digitalen Transformation einen Paradigmenwechsel. Qualitativ hochwertige und gleichzeitig höchstproduktive Maschinen und Anlagen zu produzieren, wird nur noch eines von vielen Differenzierungsmerkmalen und Verkaufsargumenten sein. Der bisherige Fokus auf der Maschine und ihrer Qualität sowie Effizienz wird sich auf Funktionalitäten und Dienste erweitern, die direkte Mehrwerte für den Nutzer realisieren. Aus Sicht des Betreibers soll zukünftig oft nur noch für das bezahlt werden, was auch wirklich in Anspruch genommen wird. Die „As-a-services“ (XaaS) Geschäftsmodelle rücken damit in den Fokus des zukunftsgerichteten Maschinen- und Anlagenbaus. Die Vorteile auf Herstellerseite sind unter anderem planbare, wiederkehrende Einnahmequellen und die tieferen Einblicke in das Betreiberverhalten.

Diese Veröffentlichung unterstützt und konsolidiert das Verständnis sowie die Umsetzung des Konzepts der Fähigkeiten in der Produktionsautomatisierung aus Sicht des Maschinen- und Anlagenbaus anhand von Vorgehensbeschreibungen und konkreten Beispielen. Um Effizienz sowie Skalierung und damit Akzeptanz und Profitabilität des Konzepts zu ermöglichen, ist Interoperabilität eine entscheidende Voraussetzung. Durch die beschriebene, mögliche Umsetzung der Fähigkeiten in der Produktionsautomatisierung mit dem Kommunikationsstandard Open Platform Communication Unified Architecture (OPC UA) unter Berücksichtigung des branchenübergreifenden Informationsmodells, der Companion Specification „OPC UA for Machinery“, fügt sich das Konzept nahtlos in die bestehenden Interoperabilitätsaktivitäten des Maschinen- und Anlagenbaus ein.

An dieser Stelle sei den beteiligten Unternehmen, Forschungsinstituten und der Plattform Industrie 4.0 für die erfolgreiche Abstimmung und das gemeinsame Erarbeiten zu danken.

Ausdrücklicher Dank gebührt Prof. Dr.-Ing. Rüdiger Daub stellvertretend für das Fraunhofer-Institut für Gießerei-, Composite- und Verarbeitungstechnik (IGCV) und Patrick Zimmermann, M.Sc. für die wissenschaftliche Erarbeitung dieser Veröffentlichung und Organisation des industriebegleitenden Arbeitskreises.

Wir wünschen eine spannende Lektüre!

Vorwort



Prof. Dr.-Ing. Rüdiger Daub
Fraunhofer IGCV



Patrick Zimmermann
Fraunhofer IGCV

Die steigende Komplexität von Produkten und der Kundenwunsch nach Individualisierung stellt produzierende Unternehmen vor immer größere Herausforderungen. Erschwerend hinzu kommen kürzere Produkt- und Innovationszyklen sowie ein wachsender globaler Wettbewerb. Um dennoch auch in Hochlohnstandorten wie Deutschland eine wirtschaftliche Produktion zu gewährleisten, ist ein hoher Automatisierungsgrad erforderlich. Gleichzeitig verursacht Automatisierung höhere Aufwände in der Planung und Inbetriebnahme, so dass eine wirtschaftliche Produktion oft erst bei großen Stückzahlen möglich ist. Dies liegt vor allem an der meist hohen Komplexität von Automatisierungssystemen, welche zusätzlich Unterschiede bezüglich ihrer Kommunikationsschnittstellen, ihrem Programmcode sowie ihren bereitgestellten Funktionalitäten aufweisen. Diese Problematik zieht sich vom Hersteller kleinster Komponenten über die Maschinen- und Anlagenhersteller bis hin zu großen Endkunden. Besonders davon betroffen sind mittelständische Unternehmen im Maschinen- und Anlagenbau, da sie oft nur in kleineren Stückzahlen produzieren oder sogar ausschließlich Sonderlösungen für ihre Kunden anbieten.

Die Einführung von Lösungsansätzen aus dem Bereich Industrie 4.0 kann Abhilfe schaffen und durch geschickte Kombination verschiedener Konzepte und Technologien zu einer deutlichen Reduktion der Aufwände für die Planung und Inbetriebnahme automatisierter Produktionssysteme führen. Ein Ziel ist dabei die Erhöhung der Interoperabilität zwischen beliebigen Produktionsressourcen. Beliebige Systeme sollen bis hin zu einer vollständig automatisierten Einrichtung (Plug-&-Produce) einfacher miteinander interagieren können, wie es Nutzer von Geräten aus dem Consumer-Bereich gewohnt sind.

Es wird dort mittlerweile als selbstverständlich erachtet, dass Geräte über standardisierte Schnittstellen wie Bluetooth®, Wi-Fi oder USB miteinander kommunizieren können. Der Schlüssel ist dabei, dass die beteiligten Systeme ein gemeinsames Verständnis ihrer vorliegenden Funktionalitäten besitzen und diese einheitlich kommunizieren. Eine Tastatur teilt über USB die Fähigkeit mit, eine Eingabe zu ermöglichen, ein Bluetooth Lautsprecher entsprechend die Fähigkeit einer Audiowiedergabe.

In der vorliegenden Veröffentlichung wird gezeigt, wie dieses Konzept auf den industriellen Anwendungsfall übertragen werden kann. Dabei können den Produktionsressourcen, wie im Consumer-Bereich, solche standardisierten Fähigkeiten (z.B. „Bewegen“) zugeordnet werden. Dies lässt jedoch die Frage nach der Kommunikation und Darstellung dieser Fähigkeiten offen. Nur wenn andere Produktionsressourcen diese Fähigkeit „verstehen“ und darauf reagieren, wird die Interoperabilität erhöht. Hierfür zeigt die Veröffentlichung eine Kombination verschiedenster Technologien wie beispielsweise den Einsatz des Kommunikationsstandards Open Platform Communication Unified Architecture (OPC UA). Dieser hat in den letzten Jahren mit zunehmendem Tempo Anwendung in der Industrie gefunden. Das Wort „Architecture“ ist hierbei entscheidend, denn OPC UA ermöglicht die Bereitstellung unterschiedlicher Kommunikationsmechanismen und Informationen, welche an das Konzept der Fähigkeiten adaptiert werden können. Am Fraunhofer IGCV beschäftigen wir uns dabei bereits seit vielen Jahren mit den verschiedenen

Anwendungsfällen von OPC UA und haben in diesem Zuge auch die Umsetzung von Fähigkeiten intensiv untersucht.

Wir wollen mit dieser Veröffentlichung nun einen umfangreichen Überblick über die Anwendung von Fähigkeiten im Bereich der Planung und Inbetriebnahme sowie die möglichen Technologien zur Umsetzung geben. Dabei möchten wir vor allem den Maschinen- und Anlagenbau ermutigen sich mit diesen neuartigen Konzepten zu beschäftigen, um somit in auf Interoperabilität ausgelegten Produktionssystemen kompatibel zu sein. Diese Fähigkeit ist die Voraussetzung, um Planungsaufwände für automatisierte Produktionssysteme zu reduzieren und auch zukünftig noch eine wirtschaftliche Produktion in Deutschland gewährleisten zu können.

Management Summary

Unternehmen des Maschinen- und Anlagenbaus sowie Betreiber sehen sich heute mit den Anforderungen konfrontiert, neue Geschäftsmodelle sowie tech-nologische Dienstleistungen anzubieten. Sie befinden sich im Spannungsfeld zwischen steigender Komplexität ihrer Produkte, kürzer werdenden Innovationszyklen sowie dem wachsenden globalen Wettbewerb. Um diese Herausforderungen zu adressieren, existieren verschiedenste Forschungsansätze, wobei insbesondere im Bereich der Automatisierung Potentiale durch den Einsatz von Fähigkeiten in der Planung, Inbetriebsetzung und dem Betrieb im Fokus stehen. Ziel dieser Veröffentlichung ist ein konsolidiertes und konsistentes Verständnis sowie eine Orientierungshilfe für das Konzept der Fähigkeiten in der Produktionsautomatisierung zu widerzugeben.

Fähigkeiten beschreiben dabei auf generische Art und Weise, was eine bestimmte Automatisierungsressource „kann“. Eine lineare Antriebseinheit kann dabei „Bewegen“ oder ein Greifer „greifen“ und zwar unabhängig von der spezifischen Bauart oder dem Hersteller. Es handelt sich somit um ein Hersteller- und Ressourcenübergreifendes Konzept, welches die Engineering-Aufwände für die Nutzer dieser Ressourcen drastisch reduzieren kann.

In der Planungsphase werden Fähigkeiten als sogenannte „Capabilities“ genutzt, um eine einheitliche Funktionsbeschreibung zu ermöglichen und somit auf Basis der Anforderungen generisch die richtigen Ressourcen auszuwählen. Sogenannte Ontologien helfen dabei die Zusammenhänge zwischen den Capabilities zu verstehen. Beispiele hierfür wäre der Zusammenhang zwischen „Bewegen“ und der Unterart „linear Bewegen“ aber auch die Tatsache, dass drei „linear Bewegen“ Capabilities zusammen ein dreidimensionales Bewegen ermöglichen. Die Capabilities selbst können in den Verwaltungsschalen der jeweiligen Ressourcen beschrieben werden.

Sollen diese Fähigkeiten nun konkret auf den Ressourcen ausgeführt werden, so kommen die sogenannten „Skills“ ins Spiel. Während es heutzutage üblich ist eine Bit und Byteweise Programmierung von Ressourcen im Steuerungs-code durchzuführen, die sich auch noch von Hersteller zu Hersteller unterscheidet, so verfolgen Skills einen serviceorientierten Ansatz. Über standardisierte Schnittstellen wie z.B. OPC UA können die Fähigkeiten als sich selbst beschreibende und direkt ausführbare Skills dargestellt werden. Dies ermöglicht eine einheitliche Ansprache über alle Ressourcen hinweg, ohne spezifisches Expertenwissen über die Ansteuerung jeder Ressource zu benötigen und somit aus Sicht der Endanwender eine drastische Reduktion der Inbetriebnahmeaufwände.

Doch auch im Betrieb kann eine derartige standardisierte Fähigkeitsbeschreibung helfen Produktionsprozesse zu optimieren. Typische Beispiele wären Kapazitätsschwankungen, welche in der heutigen durch Globalisierung geprägten Wirtschaft häufig vorkommen. Durch die einheitliche Beschreibung sowie Möglichkeit der Ansteuerung ist es möglich aus Planungssicht direkt weichen in der Produktion zu stellen und somit dynamisch und unmittelbar Aufträge zu priorisieren bzw. Produktionskapazitäten zu erhöhen. Es ist somit keine Übersetzung dieser Priorisierungen über die verschiedensten Softwaresysteme hinweg erforderlich, welche auch mit einem deutlichen Zeitverzug einhergehen. Durch diese höhere Produktionsdynamik kann die Produktion somit deutlich unmittelbarer an die Auftragsanforderungen angepasst und Liefertreue sowie Zufriedenheit der Endabnehmer gesteigert werden.

Insgesamt könnten Fähigkeiten somit den gesamten Engineering Prozess im Maschinen- und Anlagenbau, von der frühen Planungsphase bis hin zur Inbetriebsetzung erleichtern sowie die Flexibilität und Reaktionsfähigkeit im Betrieb bei den Endkunden steigern.

1 Einleitung und Motivation

1.1 Einleitung

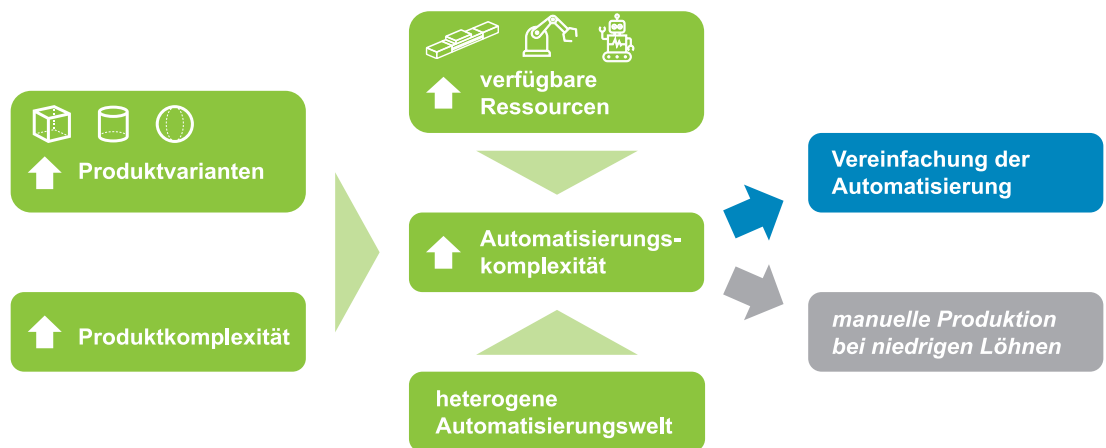
Mit der Vision von Industrie 4.0 sollen produzierende Unternehmen im Grundsatz robuster gegenüber Marktveränderungen werden. Bei diesen Veränderungen handelt es sich vor allem um starke Nachfrageschwankungen durch volatile globale Märkte sowie durch den stets zunehmenden Kundenwunsch nach individualisierten Produkten (LINDEMANN ET AL. 2006). Um letzterem gerecht zu werden, verfolgen Hersteller zum einen eine Diversifizierung ihres Produktportfolios und zum anderen eine Kundenindividualisierung je nach Produkt (WIENDAHL ET AL. 2004). Ein Beispiel hierfür ist die Automobilindustrie, welche über die letzten Jahre verstärkt SUV-, Hybrid- sowie Elektroderivate ihrer gängigsten Modellreihen entwickelt haben und zudem weiterhin eine Individualisierung dieser Fahrzeuge über unzählige Ausstattungsvarianten ermöglicht. Auch im Bereich der Hersteller von Automatisierungskomponenten ist ein Trend hinsichtlich einer derartigen Kundenorientierung bei Massenprodukten anhand der über die letzten Jahre hinweg stark zunehmenden Variantenvielfalt zu beobachten. Ein Beispiel hierfür sind Roboterhersteller, welche nicht nur immer geringere Abstufungen bezüglich Traglast und Reichweite bereitstellen, sondern auch zusätzlich ihr Portfolio vermehrt um kollaborative Roboter erweitern.

Diese Trends stellen die Produktionsautomatisierung vor neue Herausforderungen bezüglich Flexibilität und Wandlungsfähigkeit der Produktionslinien (NYHUIS 2008, KOREN & SHPITALNI 2010). Anforderungen an zukünftige Produktionsumgebungen sind somit die Produktion verschiedenster Produkte bis hin zur Losgröße 1 (Flexibilität) sowie die Anpassungen der Linie, um neu entwickelte Produkte herstellen zu können (Wandlungsfähigkeit). Da oft eine Vielzahl an Varianten nahezu parallel produziert werden, steigt auch die Gesamtzahl der benötigten Produktionsschritte und -stationen.

In Kombination mit dem konstanten technischen Fortschritt, welcher die Komplexität der Produkte erhöht, erschweren diese Trends die Planung und Inbetriebnahme automatisierter Produktionsumgebungen stark. Ein wesentlicher Grund für die hohen Aufwände bei der Erstinbetriebnahme sowie bei Anpassungen an neue Produktvarianten ist die Heterogenität in der Produktionsautomatisierung. So finden sich hier deutliche Unterschiede in den Programmieroberflächen, Steuerbefehlen sowie in der Konfiguration der Komponenten. Zudem ist eine Vielzahl zueinander inkompatibler Steuerschnittstellen und Protokolle vorzufinden. Die Softwareentwicklung und Projektierung ist gerade im Maschinen- und Anlagenbau mittlerweile ein wesentlicher Kostenfaktor. Dabei erfolgt die Ansteuerung der einzelnen Komponenten immer noch bit- und byteweise, statt serviceorientiert, wie es in der I4.0-Vision vorgeschlagen wird (KAGERMANN ET AL. 2013).

Abbildung 1:

Herausforderungen in der Produktionsautomatisierung



Diese hohen Aufwände führen gerade im Bereich der Montagetechnik dazu, dass sich automatisierte Montagelinien nur bei hohen Losgrößen als wirtschaftlich erweisen. Sollen nur kleine Stückzahlen produziert werden, so ist eine manuelle Montage gerade in Ländern mit niedrigeren Löhnen deutlich wirtschaftlicher und schneller an neue Produktvarianten anzupassen.

Um Deutschland somit weiterhin als Produktionsstandort zu sichern, muss das Engineering von Montagesystemen derart vereinfacht werden, dass eine automatisierte Produktion auch bei häufig wechselnden Produkten wirtschaftlicher bleibt als die manuelle Produktion im Ausland.

1.2 Vereinfachte Lösungsfindung bei Automatisierungssystemen

Ein Ansatz, um die genannten Engineering Aufwände zu verringern, stellt das Konzept der „Fähigkeiten“ dar (engl. Skills bzw. Capabilities). Dabei handelt es sich um abstrahierte Funktionalität beliebiger automatisierter Ressourcen, bei welchen es sich zum Beispiel um einzelne Geräte, Stationen, Zellen, Maschinen oder ganze Anlagen handeln kann. Ein Robotersystem mit Greifer und auch eine Pick-&-Place-Einheit können beispielsweise eine „Bewegen-Fähigkeit“ bezogen auf ein Werkstück bereitstellen. Ziel ist es, in der Planung von Automatisierungssystemen einen stärkeren Fokus darauf legen zu können, was getan werden soll, und nicht wie es erledigt werden kann. Somit können Automatisierungsprozesse mit ihren Anforderungen zunächst „ressourcenneutral“ beschrieben werden, ohne bereits auf einen oder mehrere spezifische Hersteller festgelegt zu sein. Damit letztlich die passenden Ressourcen gefunden werden können, ist es notwendig, dass die Gerätehersteller ihren Produkten die passenden Fähigkeiten zuordnen. Aktuell entstehen hohe Aufwände, da Kunden auf Basis der Datenblätter von Komponenten verschiedener Hersteller zunächst herausfinden müssen, „was“ eine Komponente leisten kann, also welche Funktionalität diese anbietet. Die reine Zuordnung von Fähigkeiten kann bei vielen Komponenten recht offensichtlich sein, beispielsweise die bereits genannte Bewegen-Fähigkeit zu einem 6-Achs-Knickarm-Roboter. Jedoch können Fähigkeiten zum einen weiter spezialisiert oder auch mehrere Fähigkeiten zu einer höherwertigen kombiniert werden. Betrachtet man beispielsweise eine lineare Antriebseinheit, so kann diese nur die Fähigkeit „lineare Bewegung“ anbieten, während ein Verbund mehrerer Lineareinheiten als Pick-&-Place-Einheit wiederum auch eine mehrdimensionale „Bewegen-Fähigkeit“ bereitstellen kann. Neben der eigentlichen Fähigkeit ist aber auch entscheidend unter welchen Randbedingungen (*Zusicherungen*) diese ausgeführt werden können. So bestehen bei Bewegungsfähigkeiten

beispielsweise Limitierungen bezüglich Traglast, Genauigkeiten oder Arbeitsraum. Wird ein beliebiges Automatisierungsproblem nun auf Basis von Fähigkeiten beschrieben und werden diese Randbedingungen entsprechend als Anforderungen hinterlegt, so kann auf dieser Basis eine herstellerneutrale Zuordnung (*Matching*) zwischen den fähigkeitenbasierten *Anforderungen* und den *Zusicherungen* der Ressourcen stattfinden. Der Hersteller sowie der eigentliche Ressourcentyp haben hierbei keinen unmittelbaren Einfluss auf das *Matching*, stattdessen sind rein funktionale Aspekte von Relevanz. Neben einer deutlich schnelleren Auswahl möglicher passender Ressourcen könnte somit auch auf die am besten geeignete Ressource hin optimiert werden. In der Praxis werden hingegen oft die Komponenten ausgewählt, die bei den Entwicklern bereits „bekannt“ sind, da es zu zeitaufwändig ist für jede neue Problemstellung sämtliche am Markt verfügbaren Ressourcen zu analysieren.

1.3 Vereinfachtes Engineering durch Fähigkeiten

Wurden auf Basis ihrer Fähigkeiten die passenden Ressourcen zur Umsetzung eines Automatisierungsproblems gefunden, so entstehen immer noch hohe Aufwände bei der Inbetriebnahme. Gerade hier können Fähigkeiten ihr volles Potential ausschöpfen, sofern sie mit passenden Technologien realisiert werden. Insbesondere im Fertigungsumfeld steigt die Bedeutung von kleinen und mittleren Losgrößen (HERMANN 2021, HERMANN ET AL. 2019).

Klassischerweise werden Automatisierungssysteme händisch programmiert, um die ressourcenspezifischen Unterschiede ausgleichen zu können. So sind beispielsweise herstellerspezifische Softwareoberflächen vorhanden, um verschiedene Speicherprogrammierbare Steuerungen (SPS) oder Robotersteuerungen programmieren zu können. Zudem unterscheiden sich die eigentlichen Befehle, welche zum

Steuern der einzelnen Ressourcen notwendig sind. Hier müssen oft „Treiber“ geladen werden (z.B. in Form von Gerätebeschreibungs-Dateien), damit Automatisierungsressourcen korrekt angesprochen werden können. Ein wesentlicher Grund dafür ist die immer noch bit- und byteweise Ansteuerung von Komponenten. So repräsentieren einzelne Bits bestimmte Ressourcenbefehle, welche entweder durch den Treiber identifiziert oder vom Nutzer in der Dokumentation nachgeschlagen werden müssen. Für bestimmte Komponenten kann es weiterhin erforderlich sein, eine Konfiguration direkt am Gerät oder mithilfe von proprietärer Software vorzunehmen (z.B. zum Einstellen der Betriebsart oder zum Durchführen einer Referenzfahrt). Nicht zuletzt sind auch die heterogenen Kommunikationsschnittstellen im industriellen Umfeld ein entscheidender Faktor, welche die Inbetriebnahme erschweren. Wenn gleich automatisierte Ressourcen eigentlich nur auf Basis dessen was sie leisten ausgewählt werden sollten, so entscheiden in der Praxis oft andere, die eigentliche Funktion nicht betreffende Faktoren. Ein Beispiel für ein solches Auswahlkriterium ist die passende Kommunikationsschnittstelle, welche keinen direkten Einfluss auf die Funktionalität hat, jedoch oft großen Einfluss auf die Ressourcenauswahl hat. Diese Abhängigkeiten können von Automatisierungsherstellern auch gezielt genutzt werden, um Kunden an ihr eigenes Ökosystem zu binden.

Doch wie kann die Beschreibung von Fähigkeiten nun bei diesen Problemstellungen helfen? Beschreibt man die Funktionalitäten von Automatisierungsressourcen fähigkeitenbasiert, so können viele der herstellerspezifischen Eigenheiten harmonisiert werden. Würden beispielsweise verschiedene Hersteller von Pick-&-Place Einheiten auf die gleiche Art und Weise eine „Bewegen“-Fähigkeit anbieten, innerhalb welcher die verschiedenen Bewegungsparameter konfiguriert werden können, so würden sowohl die individuelle Programmieroberfläche als auch die herstellerspezifischen Steuerbefehle und Konfigurationen entfallen. Jedoch bleibt insbesondere

bezüglich möglicher Kommunikationsschnittstellen die Frage offen, „wie“ diese Fähigkeit denn nach außen angeboten wird.

Wie bereits eingangs erwähnt, müssen Fähigkeiten hier mit einer geeigneten Technologie verbunden werden, welche eine herstellerunabhängige Ansteuerung ermöglicht. Als vielversprechend erweist sich zum Beispiel die Open Platform Communications Unified Architecture (OPC UA), da sie nicht nur eine reine Kommunikationsschnittstelle, sondern eine Gesamtarchitektur anbietet, in welcher beispielsweise auch beschreibende Informationen von Fähigkeiten modelliert werden können. So ist es möglich in einem OPC UA Server ein Informationsmodell aufzubauen, welches ein entsprechendes „Browsen“, Konfigurieren und Ausführen der bereitgestellten Fähigkeiten mittels eines beliebigen OPC UA Clients realisiert. So können beliebige Automatisierungskomponenten stets gleichermaßen in Betrieb genommen und die Aufwände der Inbetriebnahme somit drastisch gesenkt werden. Die Beschreibung solcher Fähigkeiten in OPC UA erfordert ein hohes Maß an Standardisierung, das durch die Verwendung sogenannter OPC UA Companion Specifications erzielt werden kann. Das genaue Vorgehen zur Implementierung von Fähigkeiten in OPC UA wird in Kapitel 3.5 und die Beispielhafte Anwendung in Kapitel 4 aufgezeigt.

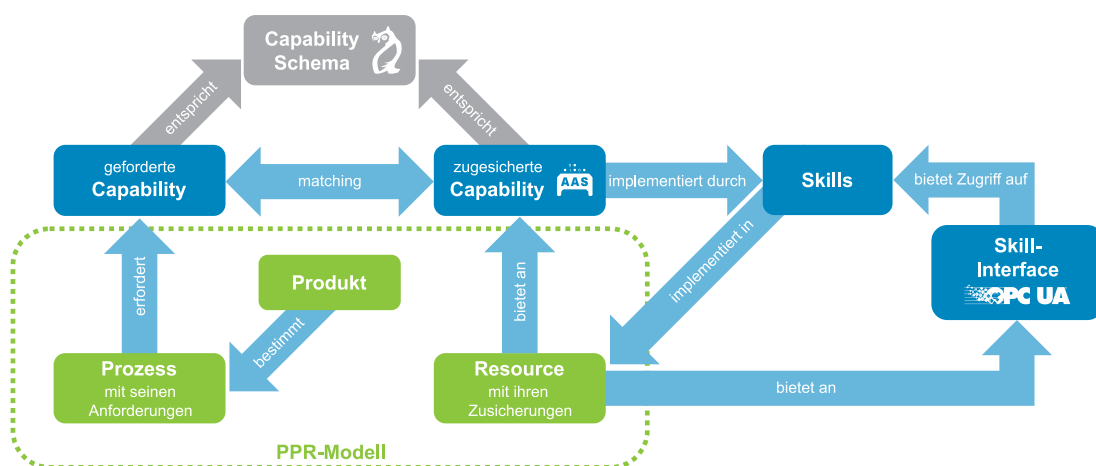
1.4 Fähigkeiten, Skills oder Capabilities?

Während im Deutschen für das Konzept das Wort „Fähigkeiten“ verwendet wird (vgl. HAMMERTINGL 2020), so gibt es im Englischen die Unterscheidung zwischen Skills und Capabilities (MOTSCH ET AL. 2021). Wenn auch beide Begriffe im Deutschen mit dem Wort „Fähigkeit“ übersetzt werden können, so handelt es sich hierbei um zwei verschiedene Konzepte, ähnlich den Begriffen Safety und Security, die beide als „Sicherheit“ übersetzt werden können. Im Folgenden sollen diese Unterschiede kurz erläutert werden.

Capabilities sind die abstrakten und ressourcenunabhängigen Fähigkeiten, welche genutzt werden können, um Anforderungen seitens des Produktionsprozesses und Zusicherungen der

Ressourcen zu beschreiben (vgl. Abbildung 2). Als grundlegendes Konzept zur Beschreibung von Produktionsanlagen und den in ihnen ablaufenden Vorgängen kann das sogenannte Produkt-Prozess-Ressource Modell (PPR-Modell) verwendet werden, mit dessen Hilfe auch die Verwendung der *Capabilities* erläutert werden kann (Backhaus 2016; SPUR & KRAUSE 1997; DRATH 2010; HOLLMANN 2013). Der benötigte Prozess wird dabei in der Regel vom Produkt bzw. der Produktspezifikation vorgegeben. Das Produkt bestimmt beispielsweise wie der Montageprozess ablaufen muss. So sind in der Regel mehrere Montageschritte erforderlich, welche auch Prozessschritte aus dem Bereich *Fügen* gemäß der DIN 8593 enthalten. Solche Normen können als Grundlage für die Definition von Fähigkeiten genutzt werden.

Abbildung 2:
Zusammenhang zwischen Capabilities und Skills



Die Anforderungen des Fügevorgangs *Einlegen* gemäß DIN 8593-1 können beispielsweise durch eine *Bewegen* Capability beschrieben werden, in dem hier der benötigte Bewegungsumfang, geforderte Genauigkeiten oder auch die zu bewegend Masse angegeben werden. Ressourcen können ebenfalls die *Bewegen* Capability anbieten und über diese entsprechende Bewegungsräume, Genauigkeiten oder bewegbare Masse (Traglast) zusichern. Eine solche Ressource könnte beispielsweise eine Pick-&-Place Einheit oder auch eine Fräsmaschine sein.

Auf Basis dieser geforderten und zugesicherten *Capabilities* kann dann das sogenannte *Matching* stattfinden, mit dessen Hilfe die für die Ausführung des Prozesses geeigneten Ressourcen gefunden werden können. Damit dieses *Matching* überhaupt durchgeführt werden kann, ist es wichtig, dass vergleichbare *Capabilities* vorliegen. Dafür bedienen sich sowohl die Seite der *Anforderungen* als auch der *Zusicherungen* dem gleichen Capability-Schema, welches beispielsweise durch eine Ontologie beschrieben werden kann. Der Aufbau und Einsatz einer solchen Ontologie mithilfe der sogenannten Web Ontology Language wird in Kapitel 3.3 genauer erläutert. Das Capability-Schema bietet die Möglichkeit mit weiteren Eigenschaften die Capability zu spezifizieren. Dabei ist zu beachten, dass es unterschiedliche Abstraktionsebenen geben kann, z.B. eine *Handling Capability*, welche konkreter durch *Bewegen* bzw. lineares *Bewegen* realisiert werden kann. Weiterhin können *Capabilities* auch zusammengesetzt werden, um höherwertige Funktionalitäten zu erreichen. Eine *Pick-&-Place Capability* kann so beispielsweise durch mehrere einzelne *lineare Bewegungen* in Kombination mit einer *Greifen* Capability realisiert werden. In diesem Falle würden auch mehrere miteinander verbundene einzelne Ressourcen die Capability zur Verfügung stellen, während bei einem Robotersystem mit Greifer keine weitere Zerlegung in einzelne *lineare Bewegungen* notwendig wäre. Beim *Matching* muss es somit möglich sein diese verschiedenen Capability-Ebenen miteinander abzugleichen, um alle geeigneten Ressourcen bzw. Ressourcenkombinationen zu ermitteln. Insgesamt unterstützen die *Capabilities* vor allem

die frühen Planungsphasen im Engineering-Prozess bis zur Auswahl der geeigneten Ressourcen. Sie ermöglichen damit eine verkürzte Planungszeit und erlauben so auch eine schnellere und häufiger durchführbare Anpassung bei Änderungen.

Skills sind wiederum die konkrete Implementierung der *Capabilities* innerhalb einer bestimmten Ressource. Sie können gleiche Funktionalitäten verschiedener Ressourcentypen und Hersteller serviceorientiert in einheitlicher Art und Weise anbieten. Der Zugriff auf *Skills* erfolgt dabei über ein *Skill-Interface*, welches wie bereits erwähnt beispielsweise mit OPC UA realisiert werden kann. Die Realisierung des für den *Skill* notwendigen Programmcodes kann dabei immer noch ressourcenspezifisch sein. Eine *Pick-&-Place-Einheit* kann beispielsweise einen *Bewegen* Skill einheitlich über OPC UA anbieten, während die zur Ausführung des Skills notwendige Bewegungskoordination und Ansteuerung der Achsen durch proprietäre Befehle innerhalb der jeweiligen Gerätesteuern erfolgen, welche den OPC UA-Server bereitstellt. Die Skills werden vor allem innerhalb der Inbetriebnahme- sowie Betriebsphase verwendet. Durch die über alle Ressourcen hinweg einheitliche Bereitstellung eines Skill-Interfaces zur Gerätekonfiguration sowie für die Ausführung der geforderten Prozesse zur Laufzeit kann eine deutliche Zeitersparnis bei der Inbetriebnahme erreicht werden. Analog zu den *Capabilities* können auch die *Skills* orchestriert werden. So kann im bereits erwähnten Beispiel der *Pick-&-Place-Einheit* die übergeordnete Steuerung direkt den Skill *Pick-&-Place* anbieten, während bei dessen Aufruf unterlagert eine Sequenz einzelner „*lineare Bewegungen*“ und „*Greifen*“ Skills aufgerufen wird.

In Summe kann durch die Verwendung von *Capabilities* und *Skills* somit das gesamte Engineering von Automatisierungssystemen unterstützt werden, von der Anforderungsmodellierung bis hin zum Betrieb der automatisierten Ressourcen. Das Konzept ist sowohl auf die Erstellung neuer Systeme (*Greenfield*) als auch die Rekonfiguration bestehender Systeme (*Brownfield*) anwendbar und dient somit auch der Umsetzung einer wandlungsfähigen Produktion.

2 Mehrwert durch den Einsatz von Fähigkeiten

Fähigkeiten können sehr unterschiedlich eingesetzt werden und unterschiedliche Potenziale für verschiedene Anwendergruppen aufzeigen. Im Folgenden werden die Ziele beim Einsatz von Fähigkeiten für den Maschinen- und Anlagenbau sowie für die Endanwender der Automatisierungssysteme vorgestellt.

2.1 Ziele aus Sicht des Maschinen- und Anlagenbaus:

Innerhalb des Maschinen- und Anlagenbaus können Fähigkeiten an verschiedenen Stellen eingesetzt werden. So können Komponentenhersteller beispielsweise Ressourcen anbieten, welche direkt Capabilities sowie die zugehörigen implementierten Skills bereitstellen. Systemintegratoren können diese wiederum nutzen, um Maschinen und Anlagen in deutlich kürzerer Zeit aufzubauen. Somit kann ein kontinuierlicher Übergang zwischen Planung, Inbetriebnahme und Betrieb mit vereinheitlichter Modellierung geschaffen werden („capability-based Continuous Engineering and Operation“).

Im Folgenden soll unterschieden werden zwischen der Planungsphase, in welcher die Anforderungen aufgestellt und eine passende Automatisierungslösung bestimmt werden, sowie der Realisierungsphase, in welcher Entwicklung, Aufbau, Inbetriebnahme, Betrieb oder die Umpassung des Systems stattfinden. Auch wenn mit der Vision von Industrie 4.0 diese klassischen Planungsphasen verschwimmen und das Konzept der Fähigkeiten hierzu einen großen Beitrag leisten kann, so wird zum besseren Verständnis diese Aufteilung beibehalten.

2.1.1 Planungsphase

Vielzahl möglicher am Markt verfügbarer Ressourcen aus dem Bereich der Automatisierungstechnik die passenden ausgewählt werden.

Für gleiche Funktionalitäten können dabei unterschiedliche Lösungen verschiedener Anbieter existieren. Die Hersteller beschreiben ihre Komponenten grundsätzlich sehr detailliert in Datenblättern und bieten oft unzählige Abstufungen bezüglich Größen- und Leistungsklassen an. Dabei ist die eigentliche Funktionalität der Ressource und ihre möglichen Anwendungsgebiete jedoch oft nicht sofort ersichtlich. Stattdessen muss ein Kunde jede einzelne Komponente analysieren, um den möglichen Anwendungsbereich zu ermitteln und auch die Grenzen in der Anwendbarkeit der Ressourcen aus den Datenblättern extrahieren.

Von den Systemintegratoren und Endkunden wird dabei erwartet, dass sie sich in der Welt der jeweiligen Ressourcenhersteller bestens auskennen und direkt nach der exakt für ihre Anwendung geeigneten Ressource suchen. Dies führt dazu, dass auf der Kundenseite sowie auf der Seite des Vertriebs der Ressourcenhersteller eine Vielzahl an Ressourcenexperten existieren müssen, welche in Beratungsgesprächen gemeinsam nach den für die jeweilige Kundenanwendung passenden Ressourcen suchen. In den Gesprächen werden häufig viele Iterationen absolviert, da die Anforderungen oft nicht eindeutig formuliert werden und somit nicht klar ist, was der Kunde wirklich benötigt. Dabei haben Kunden oft schon sehr konkrete Vorstellungen, was denn eine mögliche Lösung für ihre jeweilige Problemstellung sein könnte, was die Suche nach einer optimalen Lösung unter Berücksichtigung sämtlicher geeigneter Ressourcen weiter erschwert. Dies ist auch darauf zurückzuführen, dass Kunden für neue Problemstellungen stets auf die gleichen, ihnen bereits bekannten, Automatisierungslösungen zurückgreifen wollen, obwohl für das spezifische Automatisierungsproblem möglicherweise besser geeignete Ressourcen am Markt verfügbar wären.

Ziele durch den Einsatz von Fähigkeiten

Grundsätzlich kann durch die fähigkeitenbasierte Beschreibung in der Planungsphase das bei den Kunden benötigte Expertenwissen bezüglich der Produkte verschiedener Ressourcenhersteller deutlich reduziert werden. Capabilities können prozess- oder produktbezogen beschrieben werden. Eine prozessbezogene Beschreibung ist beispielsweise die Capability *Bewegen* oder *Bohren*, während produktbezogene Capabilities die Aufgabe direkt über das Produkt ausdrücken, beispielsweise die *Motorhaube montieren* oder ein *Loch* fertigen. Je nach Anwendungsfall kann eine der beiden Beschreibungsarten vorteilhafter sein. Die produktorientierte Beschreibung gibt in der Regel noch nicht vor mit welchen konkreten Subprozessen die Capability ausgeführt werden soll. Die Motorhaube kann mit verschiedensten Fügeverfahren montiert und das Loch ebenso mit verschiedensten Fertigungsverfahren erstellt werden. So können beispielsweise von einem Zulieferer, Systemintegrator oder Bediener die konkreten, passenden Prozesse sowie die dafür benötigten Ressourcen zu einem späteren Zeitpunkt festgelegt werden. Die prozessbezogene Beschreibung gibt wiederum bereits die konkreten Prozesse vor, für welche auf Basis der Capabilities beispielsweise die passenden Ressourcen gefunden werden können. Im Bereich von Handhabungsaufgaben ist häufig eine prozessbezogene Beschreibung sinnvoll, während im Bereich der Werkstückbearbeitung produktbezogene Beschreibungen Vorteile mit sich bringen. Bei Werkstücken sind durch die Geometrie und Werkstoffvorgaben die benötigten Fertigungsverfahren (z.B. Fräsen, Drehen, Biegen, ...) bereits indirekt vorgegeben. Zu fertigende geometrische Merkmale (Feature) des Produktes wie Taschen oder Löcher können daher genutzt werden, um vorhandene Capabilities und Skills zur Fertigung eines geometrischen Merkmales zu identifizieren. Für den Anwendungsfall einer fähigkeitenbasierter Ressourcenauswahl ist es wichtig, dass die Beschreibung der geforderten Capabilities mit den *zugesicherten Capabilities* (vgl. Abbildung 2) übereinstimmt.

Wird hingegen im Bereich der Fertigungstechnik produktbezogen beispielsweise die Herstellung eines Lochs als Capability gefordert und dieses auf Basis der Capability-Eigenschaften weiter hinsichtlich Durchmesser und Tiefe spezifiziert, so müssen auch die Ressourcen die gleiche Beschreibung besitzen. Die Zuordnung ist hier oft sehr klar, da ein rundes Loch in einer bestimmten Tiefe in der Regel von einer Bohrvorrichtung gefertigt wird. Eine weitere Besonderheit in der Fertigung ist die Zusammensetzung von sogenannten *aktiven* und *passiven* Capabilities, die nur gemeinsam eine Zusicherung ergeben. Während *aktive Capabilities* ihre Funktionalität direkt ausführen können, so können *passive Capabilities* nur in Kombination mit *aktiven Capabilities* ausgeführt werden. Ein Beispiel hierfür wäre eine Werkzeugmaschine, bei welcher die Werkzeuge jeweils *passive Schneiden* Capabilities mit sich bringen und diese erst in Kombination mit der *Drehen* Capability der Spindel sowie den *Bewegen* Capabilities der weiteren Achsen zu den zusammengesetzten Capabilities wie *Bohren* oder *Fräsen* führen. Erst die geeignete Kombination eines spezifischen Werkzeugs mit einer spezifischen Maschine und die Auswahl geeigneter Fertigungsstrategien und -parameter führen zu hochqualitativen Produkten.

Während die produktbezogene Beschreibung und die Machbarkeitsprüfung der bereits in Betrieb genommenen Ressourcen aufgrund der Abnutzung von Werkzeugen das empfohlene Vorgehen für den Bereich der Fertigung darstellen, so sind derartige Capabilities im Bereich der Montage für eine Ressourcenauswahl nur eingeschränkt verwendbar. Wird die Capability *Motorhaube montieren* betrachtet, so stellt diese eine starke Spezialisierung dar, welche jedoch aus sehr generischen *Bewegen* und *Greifen* Capabilities aufgebaut werden kann. Für Hersteller von Ressourcen wäre es schwierig sämtliche möglichen Produkte, welche mit ihrer Komponente montierbar sind aufzulisten, damit ein Abgleich erfolgen kann. Stattdessen ist es hier sinnvoll den Montageprozess in ressourcenbezogene Capabilities zu zerlegen, in diesem Falle also *Bewegen* und *Greifen*, welche so generisch sind, dass sie von einer Vielzahl von Ressourcen angeboten werden können. Werden hingegen Speziallösungen gesucht, welche tatsächlich genau für die Montage eines

bestimmten Produkts in hohen Stückzahlen konzipiert werden soll, dann kann durchaus auch eine produktbasierte Beschreibung sinnvoll sein.

Durch die klare Trennung von Anforderungsmodellierung und Lösungssuche wird vermieden, dass Kunden bereits Anforderungen mit halbfertigen Lösungsvorstellungen vermischen. Somit ist sichergestellt, dass ein Kunde beispielsweise eine Lösung für „die Bewegung von Bauteil X mit Gewicht Y von Position A nach B“ fordert und nicht bereits ein „2D-Handlingsystem in Größe X, welches die Traglast Y“ bereitstellt. Mithilfe dieser lösungsneutralen Anforderungsbeschreibung können Ressourcenexperten der Komponentenhersteller deutlich effizienter die passenden Automatisierungslösungen auswählen. Doch auch auf Herstellerseite kann das benötigte Expertenwissen und somit die Zeit zur Erarbeitung möglicher Lösungsansätze durch den Einsatz von Fähigkeiten deutlich reduziert werden. Indem die Hersteller die Funktionalitäten und Grenzen des Einsatzbereichs ihrer Ressourcen mithilfe von *zugesicherten Capabilities* beschreiben, kann durch das *Matching* schnell eine Vorauswahl an Lösungen gefunden werden (vgl. Abbildung 2). Eine Machbarkeitsprüfung auf Ebene der ausgewählten Ressourcen ähnlich dem Bereich der Fertigung kann für bestimmte Ressourcenkombinationen durchgeführt werden (beispielsweise ein Roboter mit bestimmten Anbauten, welche seine Traglast reduzieren könnten), ist jedoch für die beschriebene Vorauswahl nicht zwingend erforderlich.

Werden die *zugesicherten Capabilities* von sämtlichen Herstellern dem Kunden durch *Verwaltungsschalen* zur Verfügung gestellt, können auch die Kunden selbst das *Matching* durchführen und so bereits sehen welche Hersteller überhaupt passende Lösungen anbieten. Somit wird der Markt geöffnet und es ist sichergestellt, dass der Kunde wirklich die „optimale Lösung“ erhält und nicht „die Lösung, die bereits bekannt ist“. Dies kann für einen Kunden bei der Suche nach neuen Lösungen auch drastisch die Anzahl der benötigten Herstelleranfragen reduzieren, da nur noch geeignete Ressourcen aufgeführt werden.

Die entsprechend notwendigen Modelle könnten über die Portale der Hersteller von Automatisierungskomponenten angeboten werden. Hierbei

existieren heute schon Content-Management-Systeme (CMS) oder Product-Information-Management-Systeme (PIM), auf welche über Webportale zugegriffen werden kann. Die Capabilities mit ihren Zusicherungen könnten zukünftig automatisiert über diese Portale angefragt werden, um das Matching durchzuführen.

Ein solches generisches herstellerübergreifendes Matching kann aus Sicht von Komponentenherstellern zum einen als Risiko gesehen werden, da Kunden nun auf neue, besser geeignete Lösungen eines Marktbegleiters stoßen. Andererseits besteht auch die Chance neue Kunden zu gewinnen, welche bisher stets die gleichen Lösungen eines anderen möglicherweise marktdominierenden Herstellers genutzt haben.

Aus Sicht eines Komponentenherstellers ist somit (bei entsprechender Produktqualität) die Chance neue Kunden zu gewinnen deutlich größer als das Risiko Bestandskunden zu verlieren, zumal die Unwissenheit über ein vielleicht besser geeignetes Produkt der Konkurrenz keinesfalls eine Marktstrategie sein sollte. Aus Sicht der Anwender von Komponenten, beispielsweise Systemintegratoren und Maschinenhersteller, bietet das Konzept eine hohe Reduzierung der Planungsaufwände. Die aufgebauten Systeme und Maschinen können zudem stärker hinsichtlich der Ressourcen optimiert werden, da durch das automatisierte Matching ein größerer Auswahlbereich an Ressourcen betrachtet wird. Aus diesem Grunde sollten aus Anwendersicht Komponentenhersteller bevorzugt werden, die Capabilities zu ihren Ressourcen anbieten.

2.1.2 Inbetriebsetzung

Wurden durch das Capability Matching die passenden Ressourcen identifiziert, so können diese beschafft werden und der Aufbau sowie die Inbetriebnahme des Automatisierungssystems beginnen. Während in der Planungsphase die Capabilities im Vordergrund standen, so sind es nun die in den Ressourcen implementierten und über Interfaces aufrufbaren *Skills*.

Im Maschinen- und Anlagenbau werden typischerweise Speicherprogrammierbare Steuerungen (SPS) oder andere Ressourcenspezifische Steuerungen (z.B. eine Robotersteuerung) eingesetzt, um Automatisierungssysteme zu steuern. Dabei verursachen die Einrichtung und Programmierung dieser Steuerungen hohe Aufwände, deren Ursache im Wesentlichen auf Inhomogenitäten in vier Teilbereiche basiert:

1. Programmieroberflächen:

Die Programmierung der jeweiligen Steuerungen erfolgt über verschiedenste Programmieroberflächen. Bekannte Vertreter hierfür sind beispielsweise TIA (Siemens), TwinCAT (Beckhoff) oder auch CODESYS. Wenn auch grundsätzlich die gleiche Programmiersprache genutzt werden kann (meist Structured Text basierend auf der IEC 61131-3), so ist die Einrichtung der verschiedenen Schnittstellen und Gesamtkonfiguration sehr unterschiedlich. Zwar gibt es die Möglichkeit große parametrierbare SPS Projekte zu nutzen, wenn Maschinen und Anlagen aus immer wiederkehrenden Ressourcen mit ähnlichem Funktionsumfang bestehen, jedoch ist gerade die Programmierung neuer Anlagen oder die Integration neuer Ressourcen sehr aufwändig. Dies verstärkt den bereits angesprochenen Trend stets identische Ressourcen zu nutzen.

Weiter verschärft wird die Situation durch Geräte innerhalb von Maschinen und Anlagen, die eine eigene Steuerung mit proprietären Programmiersprachen besitzen. Typische Beispiele hierfür sind Bildverarbeitungssysteme oder auch Robotersteuerungen. Für jede dieser Oberflächen und Sprachen müssen entsprechende Experten für die Inbetriebnahme vorhanden sein oder gewisse Systeme werden

bereits in der Ressourcenauswahl ausgeschlossen, weil solche Experten nicht zur Verfügung stehen.

2. Ressourcenspezifischer

Applikationscode:

Auch Geräte welche keine eigene Steuerung besitzen, erfordern oft einen sehr individuellen Applikationscode für deren Konfiguration und Betrieb. Grundsätzlich erfolgt die Kommunikation, völlig unabhängig von der verwendeten Kommunikationsschnittstelle, über Bits und Bytes, deren individuelle Belegung meist der entsprechenden Dokumentation der Ressource entnommen werden kann. Dabei können selbst bei der gleichen Schnittstelle (z.B. IO-Link) und der gleichen Ressourcenart (z.B. Greifern) völlig unterschiedliche Belegungen der Befehle auf Bit-Ebene vorhanden sein. Durch die bitweise Übertragung kann zudem nicht aus der Schnittstelle heraus die Information über deren Belegung bezogen werden. Grundsätzlich gibt es in diesem Bereich Weiterentwicklungen, beispielsweise der Einsatz von Funktionsbausteinen (FB) oder die Ansätze der PLCopen, jedoch sind diese Steuerbausteine nicht so austauschbar wie es suggeriert wird. Die einzelnen Hersteller bieten immer wieder die ein oder andere Zusatzfunktion an, um sich vom Marktbegleiter zu differenzieren, weshalb immer wieder Anpassungen notwendig sind und eine direkte Wiederverwendbarkeit praktisch nicht gewährleistet werden kann. Zudem sind FBs oft zu feingranular aufgebaut und müssen kombiniert werden, um überhaupt die Gesamtfunktion einer Ressource abbilden zu können.

3. Kommunikationsschnittstellen

zur Steuerung:

Unterschiedliche Kommunikationsstandards in der Feldebene führen ebenfalls zu erhöhten Integrationsaufwänden, da die Steuerungen entsprechende Module für sämtliche Schnittstellen aufweisen müssen. Alternativ kann natürlich eine bestimmte im System dominierende Schnittstelle festgelegt werden und die Ressourcenauswahl entsprechend auf diese eingeschränkt werden. Dies kann in Summe

dazu führen, dass zwar die Integrationsaufwände gesenkt werden, gleichzeitig aber möglicherweise bezüglich Funktionalität und Wirtschaftlichkeit nicht die optimale Wahl getroffen wird.

4. Komponentenkonfiguration:

Während sich einige Komponenten mit recht komplexen Operationen direkt über ihre Steuerschnittstelle durch eine übergeordnete Steuerung konfigurieren lassen, so muss häufig eine Konfiguration über eine herstellerspezifische Software erfolgen. Oft muss dabei eine eigene Konfigurationsschnittstelle verwendet werden. Hierrüber werden sämtliche Einstellungen für den späteren Betrieb vorgenommen, beispielsweise die Betriebsart, oder die Konfiguration von Referenzfahrten.

Neben diesen speziellen Problemstellungen, führt auch die grundsätzliche Entkopplung zwischen Planungs- und Inbetriebnahme-phase zu erhöhten Aufwänden. Selbst modernere Standards wie die IEC 61499 schaffen keine Abhilfe, denn der Inbetriebnehmer muss stets den in der Planung erstellten Ablauf in Steuerungscode überführen, da er erst sehr spät in den Gesamtplanungsprozess eingebunden wird.

Ziele durch den Einsatz von Fähigkeiten

Durch den Einsatz von *Skills* kann einer Vielzahl dieser Herausforderungen begegnet werden. Beim Einsatz von Ressourcen ist zu beachten, dass mit einer einheitlichen Beschreibung der Fähigkeiten auch immer eine hierfür geeignete physische Schnittstelle einhergehen muss. Es ist zwar auch bei den *Capabilities* hilfreich gleiche Austauschformate zu nutzen, um das *Matching* möglichst weit zu automatisieren, jedoch ist es in der Planungsphase einfacher diese Formate ineinander zu überführen, solange Struktur und Inhalt (= Semantik) identisch sind. Wird nur die Inbetriebnahme bzw. der Betrieb im Bereich des Maschinen- und Anlagenbaus betrachtet, so ist mit dem einheitlichen semantischen Zugriff auf die *Skills* einer Ressource auch eine einheitliche Kommunikationsschnittstelle verbunden.

Wie in Abbildung 2 dargestellt, muss klar zwischen den eigentlich ausführbaren *Skills* und dem *Skill-Interface* unterschieden werden. Ziel ist es in sämtlichen Ressourcen die *Capabilities* so zu implementieren, dass sie direkt ausgeführt werden können. Diese Implementierung kann ressourcenspezifisch sein, vor allem dann, wenn die Ressource den *Skill* direkt anbietet. Wird *Bewegen* als *Skill* betrachtet, so kann der eigentliche Steuerungscode zur Ausführung der Bewegung im Falle einer aus linearen Antriebseinheiten und Greifer bestehenden Pick-&-Place Einheit durch eine übergeordnete Steuerung realisiert werden. Hierfür können beispielsweise die FB aus der PLCopen Motion Control Bibliothek verwendet werden oder aber auch eine völlig proprietäre Programmierung, welche direkt die für die Achsen und Greifer notwendigen Bits und Bytes anspricht. Bei einer Kombination von Roboter und Greifer könnte es hingegen notwendig sein, den *Bewegen Skill* in der entsprechenden roboterspezifischen Programmiersprache zu implementieren, wenn auch die meisten Roboterhersteller mittlerweile ebenfalls PLCopen Motion Control anbieten.

Der Zugriff auf diese implementierten *Skills* erfolgt nun über das sogenannte *Skill-Interface*. Um das Ziel einer signifikanten Vereinfachung für die Inbetriebnahme zu erreichen, muss dieses Interface bestimmte Eigenschaften erfüllen. Hierzu zählt beispielsweise die Möglichkeit die *Skills* auffinden („browsen“) zu können, eine Beschreibung des *Skill* zu erhalten, Zugriff auf Konfigurations- sowie Laufzeitparameter zu erhalten sowie die Ausführung des *Skill* zu ermöglichen. Um diese Ziele zu erreichen, muss ein geeigneter Kommunikationsstandard gewählt werden. Der Kommunikationsstandard OPC UA erfüllt die genannten Anforderungen und ist zudem im Maschinen- und Anlagenbau bereits weit verbreitet. Dabei können OPC UA Server direkt in die Steuerungen der Ressourcen integriert werden und Informationsmodelle bereitstellen. In diesen kann die Beschreibung der *Skills*, ihre Konfigurations- und Laufzeitparameter sowie Methodenaufrufe zum Ausführen bereitgestellt werden. Ein generischer OPC UA Client erlaubt Zugriff auf diese Server und ermöglicht ein Browsing ihrer Inhalte.

Dies ermöglicht es die Skills verschiedenster Ressourcen gleichermaßen zu konfigurieren und aufzurufen, ohne herstellerspezifische Programmieroberflächen, Steuerbefehle, Kommunikationsschnittstellen oder Konfigurationstools benutzen zu müssen. Es kann also ein einzelnes generisches Tool für die Programmierung verwendet werden, auch wenn völlig verschiedene Steuerungen eingesetzt werden. In Summe wären somit sämtliche genannten Herausforderungen adressiert und eine deutliche Verkürzung der Aufwände sowie eine Erhöhung der Qualität der Inbetriebnahme durch weniger Fehler auf Bit- und Byteebene gewährleistet.

2.2 Ziele aus Sicht der Endanwender

Auch Endanwender profitieren von Maschinen und Anlagen, welche auf Basis von Fähigkeiten aufgebaut wurden. Derzeit bestimmen Fertigungsleitsteuerungen den Produktionsablauf. Dabei müssen die Signale über die einzelnen Ebenen der klassischen Automatisierungsebene hinweg weitergeleitet und oftmals konvertiert werden. Die Variantenbildung erfolgt im Automobilbau beispielsweise durch RFID-Tags im Produkt, die bestimmen welches Programm ausgeführt wird. Ziel beim Einsatz von Fähigkeiten wäre es vor allem die Komplexität der Systeme aus Endanwendersicht zu reduzieren, sowohl in der Planungs- als auch Betriebsphase. Weiterhin soll eine direktere Kommunikation zwischen den Leitsystemen und den prozessausführenden Maschinen- und Anlagen erfolgen, was sowohl die Transparenz der Prozesse als auch die Produktionsflexibilität erhöht. Die Fähigkeiten können dabei auch direkt durch ERP und MES angesprochen werden, sofern diese über entsprechend einheitliche Schnittstellen bereitgestellt werden.

Endanwender können weiterhin für die Anforderungsspezifikation ihrer Produktionslinien die Fähigkeiten dazu nutzen, den Prozess mit seinen einzelnen Prozessschritten direkt zu modellieren und somit die Anforderungen einheitlich zu dokumentieren. Wie bereits erwähnt, können Fähigkeiten zunächst produktbezogen modelliert und dann durch Zerlegung in immer weitere Teilschritte, beispielsweise auf einzelne Fügeoperationen, heruntergebrochen werden. Diese können dann als „Anforderungs-Capabilities“ modelliert und von einem Systemintegrator oder Hersteller von Ressourcen wie in Kapitel 2.1 beschrieben für die Ressourcenplanung oder die Inbetriebsetzung genutzt werden.

Weichenstellung in der Produktion

Ein Szenario für den Einsatz von Fähigkeiten, welches speziell auf Endkunden zutrifft, ist die vereinfachte Anpassung des Produktionsablaufs. Würde die Ablaufsteuerung des Produktionsprozesses auf Skills basieren, so könnte ein hohes Maß an Transparenz der jeweils ausgeführten Prozessschritte erreicht werden. Diese sind bei einer herkömmlichen vollständig manuell durchgeführten Programmierung oft hinter kryptischen Steuerungsvariablen „versteckt“. Ein Eingriff auf den Prozessablauf muss hierbei explizit vorgesehen und durch entsprechende Variablen vorgesehen werden.

Skills bieten einen vereinfachten Zugang zu betriebswirtschaftlichen Aspekten und ermöglichen es, diese nahtlos in den Automatisierungsprozess einzubinden. Konkret kann so beispielsweise überprüft werden, ob bestimmte „Weichen“ in der Produktion in Abhängigkeit von Auftrag oder Materialstammkonfiguration gestellt werden können. Diese Weichenstellung kann anschließend auch direkt fähigkeitenbasiert ausgelöst werden. Als „Weichen“ sind grundsätzlich sämtliche Entscheidungspunkte zu verstehen. Dies können sowohl tatsächliche „reale“

Weichen sein oder beispielsweise auch anpassbare Prozesse, Rezepturen oder Zustände (z.B. durch Sensorik).

Ziel ist es bereits aus der Bestellung heraus eine Priorisierung abzuleiten, welche in eine fähigkeitsbasierte Weichenstellung überführt werden kann. Die Kundenpriorität sollte dabei nicht nur auf ERP-Ebene, sondern über die gesamte Automatisierungspyramide hinweg bekannt sein. Jeder Auftrag im MES könnte mit einer Priorisierung versehen sein, welche flexibel in die Automatisierungstechnik eingebracht werden kann. Selbstverständlich könnte dies auch ein Leitrechner übernehmen. Im Zuge der „Auflösung“ der klassischen Automatisierungspyramide im I4.0 Kontext könnte der Informationsfluss hier aber abgekürzt werden. Ziel ist es, die Information über die Priorität eines Auftrages bis zur Weiche hin bekannt zu machen. Wenn diese Information beispielsweise bei den einzelnen Maschinen zu Beginn der Auftragsbearbeitung vorliegt, könnten noch kurzfristige Entscheidungen getroffen werden. Übergeordnet kann somit die Komplexität zwischen ERP und Leitrechner mithilfe von Skills reduziert werden und es erfolgt eine ganzheitliche Betrachtung der Produktion.

Rekonfiguration

Neben der Weichenstellung im Produktivbetrieb helfen Skills bei der Umstellung einer Produktionslinie auf eine andere Produktvariante oder ein völlig neues Produkt. Durch die Möglichkeit einer Parametrisierung von Skills können schnell Anpassungen an neue Produkte erfolgen, sofern sämtliche benötigten Skills bereits innerhalb der Produktionslinie vorhanden sind. Liegt ein Produkt mit völlig neuen Anforderungen vor, welche möglicherweise über die von den vorhandenen Ressourcen angebotenen Skills hinausgehen, so lassen sich schnell weitere benötigte Ressourcen identifizieren. Durch die Skill-basierte

Inbetriebsetzung können die manuellen Rekonfigurationsaufwände zur Anpassung der Linie an ein neues Produkt minimiert werden.

Für die Umsetzung des Konzepts der Fähigkeiten ist es notwendig einen gewissen Standardisierungsgrad anzustreben, da ein Endanwender nur so von den Fähigkeiten profitieren kann. Hierzu zählt sowohl die Modellierung der Fähigkeiten als auch die Technologie zur Umsetzung auf Skill-Ebene. Grundsätzlich werden viele Softwareanwendungen bereits heute anhand ihrer Fähigkeiten gegliedert und OPC UA bei Endanwendern bereits flächendeckend eingesetzt. Dabei ist OPC UA aufgrund seiner Herstellerunabhängigkeit und der Möglichkeit Informationsmodelle direkt im Server darzustellen, die bevorzugte Kommunikationstechnologie für das Skill-Interface, während die Skills für eine herstellerunabhängige Implementierung der Fähigkeiten sorgen. Derzeit liegt die Hauptanwendung jedoch in der Datenbereitstellung, während beim Einsatz von OPC UA als Skill-Interface vor allem Steuer und Konfigurationsdaten zu den Ressourcen gesendet werden. Für eine nahtlose Anwendung der Konzepte fehlt es aus Endanwendersicht noch an entsprechenden Werkzeugen.

Neben den genannten und aus Sicht des Maschinen- und Anlagenbaus besonders relevanten Anwendungsfällen für Capabilities und Skills, sind gerade in der Betriebsphase viele weitere Einsatzszenarien denkbar, wie beispielsweise ein schnelles Reagieren auf Störungen, die Produktionsüberwachung oder eine Optimierung des Produktionsprozesses (DIETRICH ET AL. 2022).

3 Modelle zur Beschreibung von Fähigkeiten

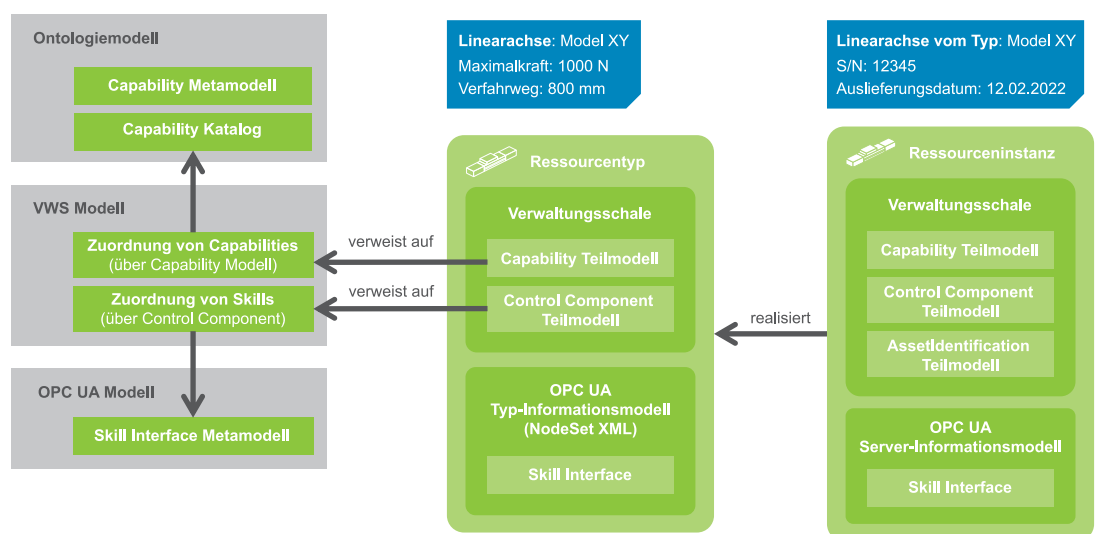
Während in Kapitel 2 verschiedene Mehrwerte aus Sicht der Stakeholder durch den Einsatz der Fähigkeiten vorgestellt wurden, werden im Folgenden die Modelle und das Vorgehen aufgezeigt, wie Fähigkeiten modelliert und angewandt werden könnten.

3.1 Einordnung der Capability- und Skill-Modelle

Wie im Abschnitt 1.4 beschrieben, ist eine Unterscheidung zwischen ressourcenbezogenen Skills und implementierungsunabhängigen Capabilities erforderlich, damit eine Planung auf Grundlagen von Fähigkeiten bereits vor dem Vorliegen konkreter Produktionsressourcen stattfinden kann.

Entscheidend für einen Abgleich von Anforderung(en) und Zusicherung(en) ist die Beschreibung von Capabilities bzw. Skills in einer „neutralen“ Sprache. Sowohl Sprache als auch Inhalt müssen entsprechend unabhängig von konkreten Prozessen und Ressourcen definiert sein. Im Gegensatz zur natürlichen Sprache muss es sich um eine in hohem Maße formale Sprache handeln, um maschineninterpretierbar zu sein und entsprechende automatisierte Matching-Algorithmen darauf anwenden zu können. Die Beschreibungen des Aufbaus dieser Modelle, d.h. wie Capabilities und Skills zu modellieren sind, werden Metamodelle genannt.

Abbildung 3:
Einordnung der verschiedenen Capability- und Skill-Modelle



Grundsätzlich sind drei verschiedene Metamodelle erforderlich, welche (1) die ressourcen- und prozessübergreifende Beschreibung von Capabilities, (2) die allgemeine Beschreibung des Skill-Interfaces, sowie (3) die Zuordnung von Capabilities bzw. Skills zu einem konkreten Prozess bzw. einer konkreten Ressource beschreiben. Diese Metamodelle und darauf aufbauende Modelle lassen sich grundsätzlich durch verschiedene formale Sprachen darstellen. Entscheidend für die Komplexität und Effizienz der Repräsentation in einer Sprache ist dabei die Ausdrucksmächtigkeit der Sprache und die sich dadurch ergebende Passung auf den darzustellenden Inhalt (hier: Capabilities, Skills, und deren Zuordnung zu Ressourcen und Prozessen).

Als geeignete Sprache zur Beschreibung des Capability-Metamodells (1) und entsprechenden konkreten Capabilities (Capability-Kataloge) bieten sich Ontologien, insb. die Web Ontology Language (OWL) an. Dafür sprechen unter anderem eine Standardisierung durch das W3C mit entsprechendem Tooling, sowie der Bezug zu einer formalen Logik, welche als Grundlage für formales, beweisbares und erklärbares Capability-Matching dienen kann. Das Capability-Metamodell ist entsprechend als OWL-Ontologie realisiert, welche das grundlegende Vokabular zur Beschreibung von Capabilities, deren Zusammenhänge und Attribute definiert.

Die Beschreibung des Skill-Interface Metamodells (2) sei durch ein standardisiertes OPC UA Informationsmodell (z.B. als Companion Specification) gegeben. Dieses beschreibt, wie Skill-Interfaces zu modellieren sind.

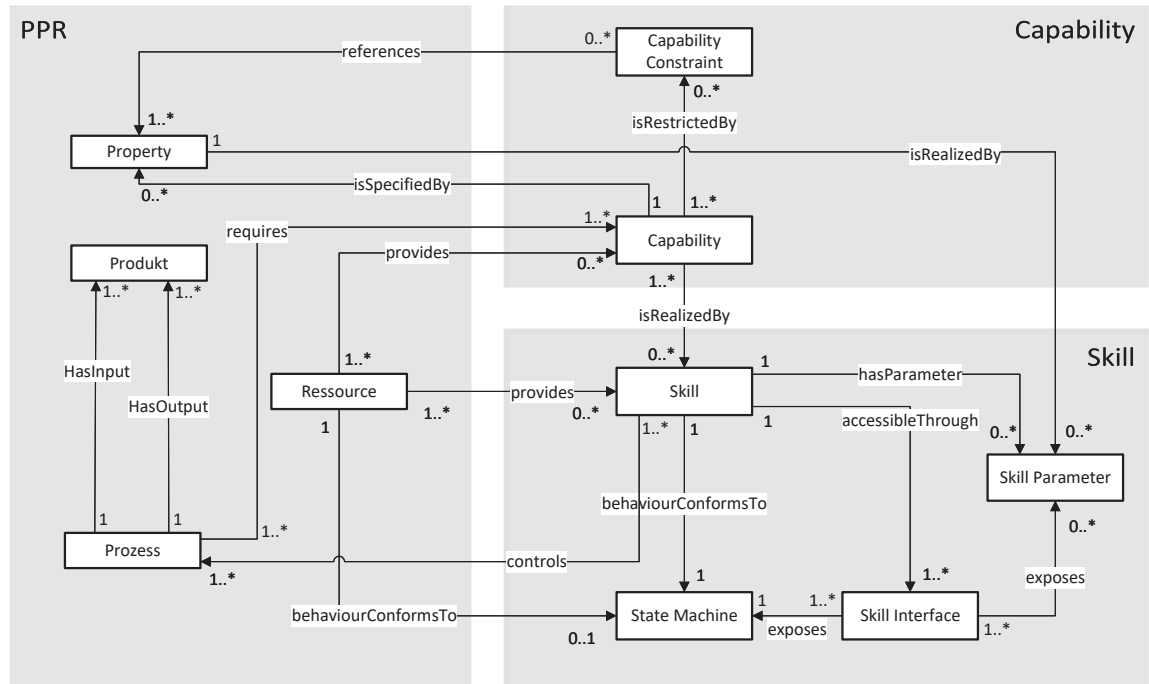
Die Zuordnung von Capabilities und Skills zu konkreten Ressourcen- und Prozesstypen (3) erfolgt über die Verwaltungsschalen bzw. deren Teilmodelle und ist über das Verwaltungsschalenmetamodell vorgegeben.

Hierfür befindet sich zur Zeit ein entsprechendes Capability-Teilmodell in Entwicklung zur Beschreibung der durch den Ressourcentyp angebotenen Capabilities (IDTA 2022A). Das Capability-Teilmodell stellt die Verknüpfung der Ressource bzw. des Prozesses zu den verwaltungsschalenexternen Ontologien her. Neben dem Verweis auf die jeweils angeforderten bzw. zugesicherten Capabilities erfolgt an dieser Stelle auch die Ausprägung der konkreten Datenwerte bzw. Datenwertebereiche. Konkret bedeutet dies, dass das Capability-Teilmodell alle Informationen enthalten muss um in eine OWL-Klassenbeschreibung, wie in Abschnitt 3.3 beschrieben, übersetzt werden zu können. Entsprechend dem PPR-Modell in Abbildung 2 benötigt ein Prozess eine Zuordnung zur formalen Beschreibung der von ihm angeforderten Capabilities. Ressourcen hingegen benötigen eine Zuordnung zur formalen Beschreibung der von ihr angebotenen Capabilities, sowie zu den von ihr angebotenen Skills. Neben dem Capability-Teilmodell befindet sich auch ein sogenanntes „Control-Component-Teilmodell“ in Entwicklung, welches den Bezug zu den angebotenen Skill-Interfaces herstellt (IDTA 2022B). Dieses Teilmodell ist in der Lage verschiedene Möglichkeiten der Ressourcenansteuerung einheitlich zu abstrahieren, unter anderem auch das Konzept der Skills.

Es sei herausgestellt, dass sich die Capability- und Skill-Beschreibungen, sowie die Zuordnung zu Ressource bzw. Prozess auf den Ressource- bzw. Prozesstyp beziehen. Die in Bezug gesetzten Capabilities bzw. Skills gelten für alle spezifischen Ausprägungen (=Instanzen) der jeweiligen Typen. Somit ist neben der Anwendung im konkreten Betrieb auch eine Nutzung in der Planungsphase möglich, sprich zu einem Zeitpunkt an dem konkrete Prozess- bzw. Ressourceinstanzen noch nicht vorhanden sind.

Abbildung 3 stellt eine Einordnung der Capability- und Skillmodelle für die Zusicherungsseite (Ressourcen) dar. Die jeweilige Ressourceninstanz beinhaltet für die Capability-Zuordnung dieselben Informationen wie der Ressourcentyp. Das Skill-Interface hingegen wird über ein konkretes OPC UA Servermodell anstelle einer reinen Node-Set Beschreibung angeboten.

Abbildung 4:

Metamodell für Capabilities und Skills auf Basis von DIETRICH ET AL. (2022)

3.2 Allgemeines Metamodell für Capabilities und Skills

Abbildung 4 illustriert ein übergeordnetes Metamodell für Capabilities und Skills auf Grundlage von DIETRICH ET AL. (2022). Das Metamodell beschreibt die Art und Weise, wie Capabilities und Skills zu beschreiben sind. Ein einheitliches Metamodell ist essentiell für die Interoperabilität, also die einheitliche Interpretierbarkeit der darauf aufbauenden konkreten Capability- und Skill-Beschreibungen. Es ist zu beachten, dass das einheitliche Metamodell für Capabilities und Skills durch unterschiedliche Sprachmodelle (jeweils für Capabilities und Skills) ausgedrückt werden kann und konform zu dem in DIETRICH ET AL. (2022) definierten Metamodell ist.

Das Metamodell gibt vor, dass eine Capability durch Angabe von Properties näher detailliert werden kann. Zudem kann die Anwendbarkeit einer Capability durch Constraints eingeschränkt sein. Constraints beziehen sich in ihrer Beschreibung ihrerseits wiederum auf Properties.

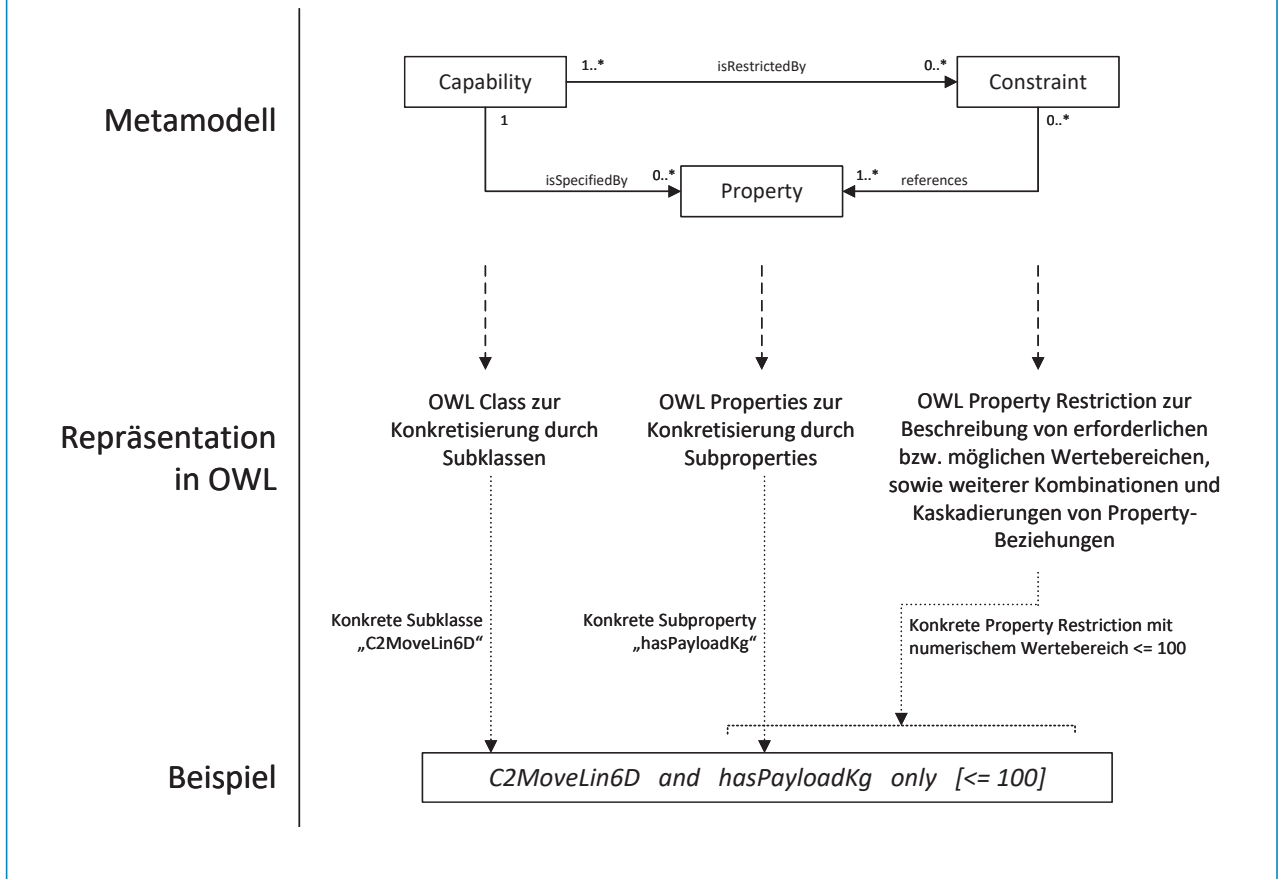
Neben der implementierungsunabhängigen Capability-Repräsentation beschreibt das Metamodell eine optionale Verknüpfung einer Capability zu ihrem Skill. Skills können Parameter besitzen (konkret Eingangs- und Ausgangsparameter), welche implementierungsseitig den Capability-beschreibenden Properties entsprechen können. Neben den Parametern ist jeder Skill durch mindestens ein Skill-Interface gekapselt und ist intern durch einen Zustandsautomaten beschrieben.

3.3 Umsetzung von Capabilities als Ontologie in OWL

Das Metamodell für Capabilities kann entsprechend obiger Argumentation durch ein Ontologiemodell in der Web Ontology Language (OWL) realisiert werden. Der Zusammenhang zwischen dem in Abschnitt 3.2 beschriebenen allgemeinen Metamodell und der Repräsentation in OWL mit Beispielen ist exemplarisch in Abbildung 5 dargestellt.

Abbildung 5:

Zusammenhang zwischen allgemeinem Metamodell und OWL Repräsentation



Neben den im Metamodell dargestellten Elementen sieht die OWL-Repräsentation die folgenden Möglichkeiten zur Beschreibung von konkreteren Capabilities vor.

OWL Sprachelement

OWL-Subklassen-Axiome zur Modellierung einer Klassenhierarchie zur Spezialisierung von Capabilities.

OWL Object Properties (hasCapability, associatedWithCapability) zur Beschreibung von Beziehungen zwischen Prozessen bzw. Ressourcen und Capabilities.

Beschreibung

Aufbau einer Capability-Hierarchie im Sinne einer Spezialisierung (nicht Komposition). Z.B. mit Bezug auf Normen für Fertigungsmethoden, wie etwa DIN 8580 oder VDA2860.

Die Verlagerung auf die Ebene der Assets (Prozess bzw. Ressource) erlaubt die Beschreibung von nicht-hierarchischen Querbeziehungen zwischen Capabilities, z.B. im Rahmen einer Komposition.

Das allgemeingültige Metamodell kann von verschiedenen Organisationen verwendet werden um konkrete Capabilities zu beschreiben. Diese Beschreibung besteht insbesondere in der Ausprägung von Subklassen des Root-Elements und der Beschreibung von Kompositionen. Dabei ist eine beliebige Spezialisierungshierarchie möglich. So können beispielsweise auf Grundlage standardisierter Richtlinien (bspw. Handhabungstechnik) oder branchenspezifisch (bspw. Robotik) Ontologien für generische Capabilities in den jeweiligen Bereichen veröffentlicht werden. Diese können bei Bedarf bis auf die Ebene von konkreten Firmen und deren Produkten herab spezifiziert werden. Die Formalisierung und der modulare Ansatz ermöglichen somit ein automatisiertes Capability-Matching. Beispielsweise kann eine Anforderung etwas zu bewegen (Capability Bewegen bzw. Move, allgemein spezifiziert für die Handhabungstechnik) durch einen Roboter einer konkreten Firma (Capability *Move linear 6D*) angeboten werden. Durch die Verwendung der gemeinsamen Sprache und den Spezialisierungspatterns kann der Roboter als Anbieter der angeforderten Capability klassifiziert werden. Analog besteht die Möglichkeit zum Abgleich von Wertebereichen für Properties. Dazu kann ein Capability-Constraint einer Zusicherung durch eine Property Restriction derart formuliert werden, dass die Capability nur Property-Ausprägungen eines bestimmten Wertebereichs erfüllt. Umgekehrt kann ein Capability-Constraint einer Anforderung durch eine Property Restriction derart formuliert werden, dass die Capability Property-Ausprägungen eines bestimmten Wertebereichs erfüllen muss. Beispielsweise kann die Zusicherung eines Roboters beschrieben werden als *hasCapability some (C2MoveLin6D and hasPayloadKg only [≤ 100])*

d.h. die Klasse aller Dinge (hier Ressourcen), welche die Capability „besitzen“ (hier in der Interpretation von „zusichern“), etwas in 6D linear zu bewegen und zwar ausschließlich Traglasten von weniger als 100 kg.

In dieser Beispielbeschreibung steht *C2MoveLin6D* für die Capability, *hasPayloadKg* für eine Property und der Ausdruck *hasPayloadKg only [≤ 100]* für ein Constraint entsprechend dem Metamodell.

Eine Anforderung kann entsprechen exemplarisch dargestellt werden als *hasCapability some (C2Move and hasPayloadKg some [≥ 90])*

d.h. die Klasse aller Dinge (hier Prozesse), welche die Capability „besitzen“ (hier in der Interpretation von „anfordern“) etwas zu bewegen und zwar Traglasten von mehr als 90 kg.

Ausgehend von dem in diesen Beispielen beschriebenen Modellierungsmuster, lassen sich durch bool'sche Klassenverknüpfungen (UND, ODER, NICHT) beliebig komplexe Capability-Beschreibungen konstruieren, welche neben numerischen auch symbolische Property-Ausprägungen beinhalten. So könnte eine Capability zu „Fügen“ mittels eines Constraint nur solche Füge-teile zulassen, die sich einer bestimmten, für einen entsprechenden Roboter-/Greifertyp greifbaren Materialklasse wie etwa „Gehäuseteil“ zuordnen lassen und selbst ein für formschlüssiges Greifen notwendiges geometrisches Feature wie eine „Einkerbung“ mit einer Mindestbreite aufweisen. Dieses Beispiel zeigt, dass je nach gewünschter Beschreibungsgenauigkeit der Matching-Szenarien durchaus auch komplexere Objektmodelle nötig werden können, die über eine einfache Liste numerischer Parameter hinausgehen. Hier würden z.B. verschiedene Domänenmodellelemente in Form von weiteren Klassen und Properties eingeführt zur Unterscheidung von Fügeteilen von stationären Teilen, geometrischen Features mit eigenen Eigenschaften, sowie Beziehungen dazwischen. Generell bieten semantische Technologien wie OWL Ontologien hierfür ein sehr geeignetes Mittel um Domänenmodelle sehr genau zu repräsentieren.

ⁱ Die hier verwendete Darstellung von OWL Klassenbeschreibung ist eine vereinfachte Darstellung der OWL Manchester Syntax.

Diese Modellierung entsprechend des Meta-modells und der Zugrundelegung eines gegebenen Capability-Modells (C2MoveIn6D als Subklasse von C2Move, ihrerseits als Subklasse von Capability, sowie *hasPayloadKg* als Data Subproperty von *hasProperty*) erlaubt die Durchführung eines Capability-Matchings. Hierzu kann auf bestehende Ansätze zum ontologiebasierten Matchmaking zurückgegriffen werden, z.B. über Erfüllbarkeit von Klassenkonjunktion oder Subsumption zwischen Klassen, mit den Techniken wie in LI & HORROCKS (2003) dargestellt. Die Open World Semantik von OWL erlaubt hierbei zunächst eine Negativaussage über die Inkompatibilität aus Anforderung und Zusicherung, und damit die Möglichkeit zu berücksichtigende Ressourcen in einem Auswahlverfahren signifikant einzuschränken – dort wo logische Widersprüche im Abgleich der Anforderungen auftreten, können Ressourcen klar ausgeschlossen werden. Eine umgekehrte (positive) Aussage ist per se zunächst nicht ohne weiteres möglich, wenn nicht sichergestellt ist, dass die Modellierung der Capabilities eine gewisse Vollständigkeit aufweist. Dies entspricht der Intuition, dass ohne vollumfängliches Wissen über alle relevanten Rahmenbedingungen, keine Garantie für die Durchführbarkeit einer Capability-Anforderung gegeben werden kann. (Besonderheiten der Open-World-Semantik von OWL in Bezug auf ein Matchmaking wurden in GRIMM (2009) ausführlich diskutiert.)

Unter der Annahme der Kenntnis über alle relevanten Faktoren, welche für ein Matching erforderlich sind, lässt sich die vorgeschlagene Modellierung allerdings um Modellierungsmuster, wie sog. Closure Axiome, etc. erweitern, um auch Positivaussagen zu Matchings zu erzielen. Die Kenntnis über erforderliche Property-Ausprägungen ließe sich zudem bspw. über die OWL-kompatible Semantic Web Technologie der Shapes Constraint Language (SHACL) sicherstellen. Zudem

bieten sich Erweiterungspunkte des Ansatzes über die Anwendung von SPARQL Queries zur Durchführung numerischer Vorberechnungen, bspw. im Rahmen eines Preprocessings, als weitere Technologien aus dem Semantic Web Stack an. Wenn ein Fokus auf komplexere numerische Berechnungen gelegt wird, bietet sich außerdem die Einbindung von Constraint Solvern an. Generell gilt es die genaue Einsatzart der hier vorgeschlagenen Technologien im Detail für den Zweck des Capability-Matching noch zu erforschen und gerade die Kombination der Technologien auszuarbeiten. Dennoch bietet das Framework von RDF und OWL einen guten Startpunkt um den Capability-Teil des abstrakten Konzeptmodells mit Mitteln der modernen Wissensrepräsentation und dem automatisierten Schlussfolgern direkt umzusetzen und in der Praxis zu erproben bzw. zu erweitern.

3.4 Implementierung von Skills

Im Laufe der letzten Jahre wurden im Rahmen verschiedener Forschungsprojekte, wie beispielsweise DEVEKOS, BaSys4.0/4.2 oder AKOMI, die Implementierungsmöglichkeiten von Skills untersucht (MALAKUTI ET AL. 2018, DOROFEEV & ZOITL 2018, ZIMMERMANN ET AL. 2019, HAMMERTINGL 2020, VOLKMANN ET AL. 2021). Während die tatsächliche Implementierung von Skills ressourcenspezifisch durch proprietären Steuerungscode realisiert werden muss, so hat sich für die Implementierung des *Skill-Interfaces* der Kommunikationsstandard OPC UA als vielversprechender Ansatz herauskristallisiert. Kernidee der Skills ist die herstellerübergreifende Ressourcensteuerung, womit die Kommunikationsschnittstelle unmittelbar an eine erfolgreiche Umsetzung des Konzepts der Skills gebunden ist. Gerade im Bereich der Steuerungstechnik hat OPC UA mittlerweile einen hohen Verbreitungsgrad und wird als hersteller- und ressourcenneutrale Informations- und Kommunikationstechnologie eingesetzt. Die Beschreibungen von Skills mit sämtlichen Eigenschaften und Parametern kann zudem direkt innerhalb des Informationsmodells im OPC UA Server abgebildet werden. Ebenso bietet OPC UA verschiedenste Interaktionsmechanismen (z.B. read, write, method call, eventing oder auch Pub/Sub) um die Skills nicht nur zu überwachen, sondern auch steuernden Zugriff auf diese zu ermöglichen.

Im VDMA wurden in Zusammenarbeit mit der OPC Foundation sogenannte „Companion Specifications“ für verschiedene Domänen innerhalb des Maschinen- und Anlagenbaus erarbeitet. Dabei fokussieren sich die meisten dieser Spezifikationen auf die reine Beschreibung von Hersteller- und Zustandsinformationen für Anwendungsfälle wie das Asset Management oder Condition Monitoring, also vor allem den lesen-den Zugriff auf diese Informationen („read only“). Diese Art von Zugriff erfüllt die

Anforderungen einer vollständigen Interoperabilität von Maschinen im Sinne der Industrie 4.0 Vision jedoch noch nicht vollständig, weshalb es wichtig ist, einen schreibenden bzw. steuernden Zugriff auf die Maschinen und Anlagen zu ermöglichen. Erste Companion Specifications wie z.B. die PackML Zustandsmaschine (OPC 30050) oder OPC UA Programs (OPC 10000-10) zeigen Ansätze für solch einen Zugriff. Dennoch fehlt es an einem einheitlichen und domänenübergreifenden Konzept.

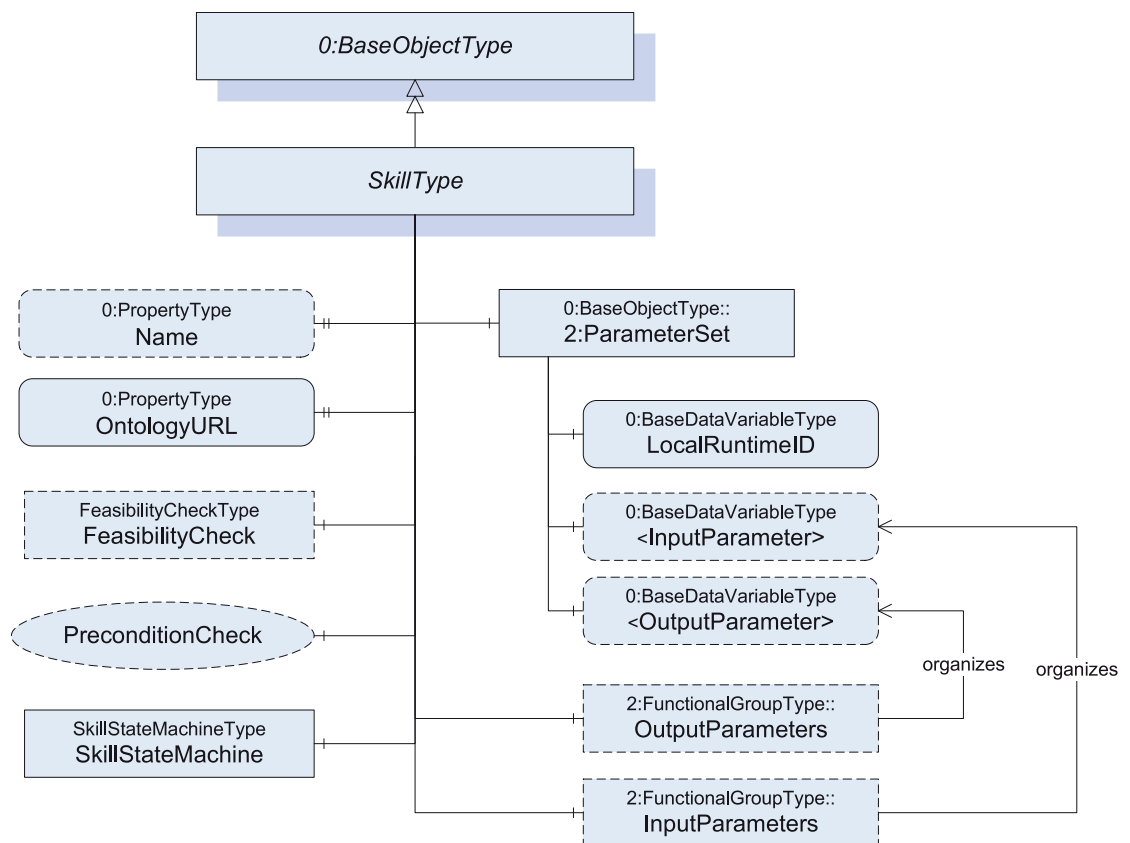
Bisher ist es in der Regel notwendig, eine Kommunikation für die Datengewinnung parallel zu den klassischen netzwerkbasierenden sowie echtzeitfähigen Kommunikationsschnittstellen auf der Feldebene aufzubauen. Dies verursacht entweder Mehraufwände, da ein zweites Netz für OPC UA aufgesetzt werden muss oder es wird der TCP/IP-fähige Teil der Echtzeitkommunikation für OPC UA genutzt, was in der Praxis häufiger der Fall ist. Letzteres kann wiederum zu starken Einschränkungen im Datendurchsatz führen, da sich OPC UA und Echtzeitkommunikation oft Netze mit maximalen Datenübertragungsraten von lediglich 100 Mbit/s teilen müssen. Das volle Potenzial von OPC UA kann somit nur ausgeschöpft werden, wenn neben der Informationsbereitstellung auch eine Ausführung von Aufgaben und Services sowie eine direkte Einbindung in die übergeordnete Produktionsplanung- und -steuerung (MES) ermöglicht wird. Daher ist es wichtig, das Konzept der Fähigkeiten in Form von ansteuerbaren Skills auf OPC UA zu übertragen.

Implementierung von Skills auf der Steuerungsebene

Wie bereits in Abbildung 2 dargestellt, muss bei der Implementierung von Skills zwischen dem *Skill-Interface* und dem eigentlichen Skill unter-

schieden werden. Das *Skill-Interface* wird als Informationsmodell auf einem OPC UA Server bereitgestellt. Abbildung 6 zeigt einen Vorschlag für dieses Modell.

Abbildung 6:
Skill Metamodell in OPC UA



Hierzu wird ein eigener OPC UA ObjectType mit dem Namen „*SkillType*“ erstellt, welcher die

wesentlichen Elemente für das *Skill-Interface* bereithält:

- **Name:**
Der Name des Skills ist optional und dient dem Nutzer als Klartext, um den Skill schneller identifizieren zu können.
- **OntologyURL:**
Die OntologyURL muss verpflichtend angegeben werden und bildet die Referenz zur Capability im Ontologiemodell (vgl. Absatz 3.3). Über diese Referenz kann eindeutig identifiziert werden um welchen Skill es sich handelt und was dieser tatsächlich leisten kann.
- **SkillStateMachine:**
Die *SkillStateMachine* stellt einen endlichen Zustandsautomaten (Finite State Machine) dar, nach welchem sich gemäß des allgemeinen Metamodells (vgl. Abbildung 4) der Skill verhalten muss. Daher dient die *SkillStateMachine* im Wesentlichen der Verhaltensbeschreibung des Skills sowie auch zu dessen Steuerung. Die einzelnen Zustände (States) sowie das entsprechende OPC UA Modell sind im nächsten Absatz beschrieben.
- **FeasibilityCheck:**
Der optionale *FeasibilityCheck* wird benötigt, um die Ausführung komplexer Skills im Voraus durch eine Machbarkeitsprüfung zu bestätigen (VOLKMANN ET AL. 2021). Als Beispiel zur Durchführung eines komplexen Skills ist in Kap. 4.2 eine mögliche Anwendung aus der Fertigungstechnik beschrieben. Der *FeasibilityCheck* kann im OPC UA Informationsmodell ebenfalls als State Machine ähnlich zu Abbildung 7 umgesetzt werden. Im Gegensatz zur Skill-State-Machine reichen hier die Zustände *Idle*, *Executing* und *Locked* sowie die Methoden *start()*, *stop()*, *reset()*, *lock()*. Der *Suspended* Zustand ist nicht notwendig, da davon ausgegangen wird, dass ein *FeasibilityCheck* nicht pausiert und wieder gestartet werden kann. Der *Feasibility-Check* enthält ebenfalls ein *ParameterSet* mit **In-** und **Output Parametern**.

Zur Überprüfung der Ausführbarkeit eines Skills werden die benötigten **Inputparameter** in das *ParameterSet* des *FeasibilityChecks* geschrieben und dieser über einen Methodenaufruf (*start()*) gestartet. Mit Hilfe von z.B. Simulationen, Entscheidungsbäumen oder Knowledge Graphen wird geprüft ob und in welcher Art die Durchführung eines Skills möglich ist. Als Ergebnis des *FeasibilityChecks* werden die Ergebnisse als **Outputparameter** zurückgegeben. Die **Outputparameter** können z.B. die Zeit zur Skillausführung, den voraussichtlichen Energieverbrauch oder auch die voraussichtlichen Fertigungskosten enthalten.
- **PreconditionCheck:**
Der optionale *PreconditionCheck* prüft kurz vor Ausführung eines Skills, ob die benötigte Resource alle Bedingungen erfüllt. Dies ist vor allem der Fall, wenn die Ausführung von vielen weiteren Faktoren abhängt. Im Bereich der Montage könnte dies beispielsweise die Überprüfung des Füllstands der Bauteillager oder im Bereich der Werkzeugmaschinen die Verfügbarkeit von Werkzeugen und ihr möglicher Abnutzungsgrad sein.
- **ParameterSet:**
Im *ParameterSet* sind die notwendigen Parameter zur Ausführung eines Skills hinterlegt:
 - **Die LocalRuntimeID** dient einem Client als numerischer Wert zur Identifikation des Skill für die Ausführung. Dieser kann vom Client bei der Konfiguration des Steuerungsablaufes festgelegt werden.
 - Der Placeholder **<InputParameter>** dient zur Festlegung beliebiger Eingangsparameter, die für die Ausführung bzw. Konfiguration des Skills notwendig sind. Dies können beispielsweise Positions- oder

Geschwindigkeitsangaben sein, die durch eine Vielzahl einzelner Parameter festgelegt werden. Sämtliche **<InputParameter>** werden dabei über die *FunctionalGroup InputParameters* organisiert.

- Der Placeholder **<OutputParameter>** dient zur Festlegung beliebiger Ausgangsparameter, die als Rückgabewert des Skills erhalten werden. Dies können beispielsweise aktuelle Ist-Werte (Drehzahl, Geschwindigkeit) während der Skillausführung sein, welche beispielsweise für die Synchronisation mit anderen Skills notwendig sind. Weiterhin ist es auch denkbar sensorische Skills zu erstellen, beispielsweise für die Qualitätssicherung, deren Ergebnis ebenfalls durch die **<OutputParameter>** dargestellt wird (HAMMERSTINGL 2020). Sämtliche **<OutputParameter>** werden dabei über die *FunctionalGroup OutputParameters* organisiert.

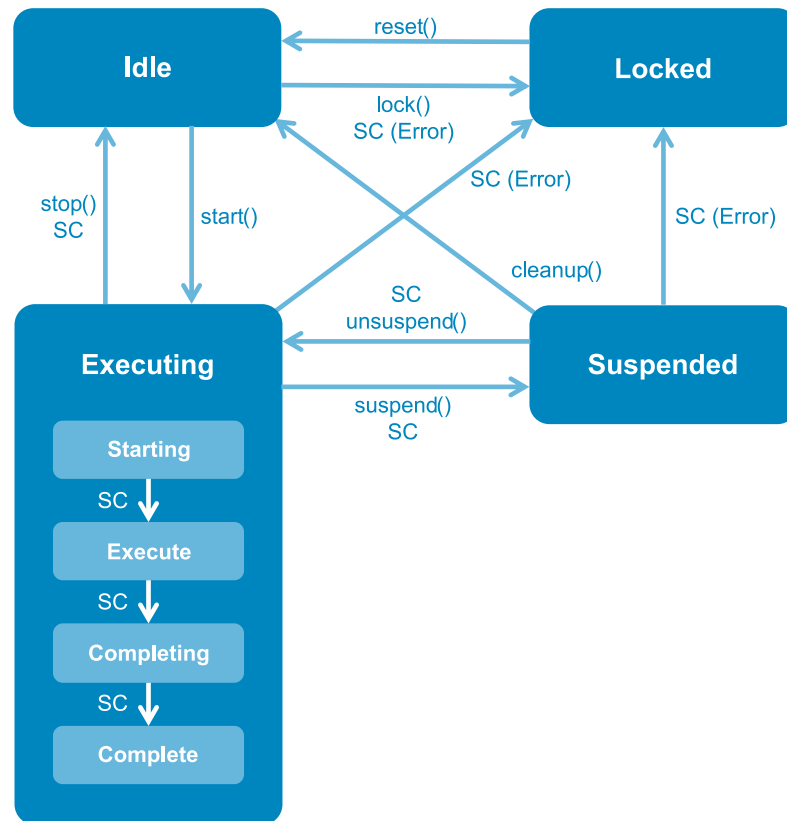
Zum Anlegen des *Skill-Interfaces* gibt es verschiedene Möglichkeiten, die sich auch je nach Steuerungshersteller unterscheiden. Manche Steuerungen bieten bereits die Möglichkeit, sogenannte „Nodeset2.xml“-Dateien einzulesen und somit den OPC UA Server zu konfigurieren. Über diesen Weg können instanzspezifische OPC UA Server Informationsmodelle für die jeweiligen Maschinen und Anlagen erstellt werden. Die im OPC UA Server angelegten Variablen und Methoden wiederum können mittels eines Bindings bzw. eines Callbacks mit entsprechenden SPS-Variablen und damit der eigentlichen Skill-Implementierung verknüpft werden.

Nicht alle Hersteller bieten den Import von XML-Dateien an. Hier wird üblicherweise die Programmstruktur der SPS direkt im OPC UA-Server widerspiegelt. Anhand von Attributen o.Ä. können die Variablen der SPS explizit für den OPC UA-Zugriff freigegeben werden. Hier muss die Struktur des Skill-Interfaces durch die Struktur der SPS-Programme, der Funktionsbausteine und durch strukturierte Datentypen abgebildet werden.

Die Implementierung eines Skills erfolgt auf der SPS bzw. dem Controller einer beliebigen Resource und kann sich, je nach verwendeter SPS oder verwendetem Controller, stark unterscheiden. Im Falle einer SPS wird empfohlen, die einzelnen Skills als Funktionsbausteine zu kapseln. Solche Funktionsbausteine können in Zukunft auch von den Herstellern einzelner Automatisierungskomponenten zur Verfügung gestellt werden. Durch die Kapselung von Skills als SPS-Funktionsbausteine ist einerseits eine einfache Ansteuerung von teilweise sehr komplexen Skills innerhalb der SPS möglich, andererseits können die Skills so mit wenig Aufwand zu komplexeren, kombinierten Skills aggregiert werden.

Damit Skills im Betrieb gesteuert und überwacht werden können, ist es notwendig ihren Zustand stets zu kennen. Innerhalb des OPC UA Informationsmodells wird daher eine State Machine definiert, deren Zustände und Zustandsübergänge in vereinfachter Form in Abbildung 7 dargestellt sind. Dabei kann der Skill in einem von vier Grundzuständen sein, deren Übergänge entweder durch das Triggern eines entsprechenden Methodenaufrufs oder mittels eines automatischen Zustandsübergangs (Automatic State Change (SC)) geschehen:

Abbildung 7:
Skill State Machine



- **Locked:**

Der Skill kann nicht ausgeführt werden. Dieser Zustand ist der Initialzustand, bevor ein Gerät beispielsweise eingerichtet wurde. Des Weiteren dient der Zustand als Fehlerzustand einer ausführenden Ressource, um eine weitere Ausführung zu unterbinden. Der *Locked* Zustand kann aus jedem anderen Zustand erreicht werden. Der Zustandsübergang wird entweder durch einen Maschinenfehler (*Error*) automatisch ausgelöst, kann aber auch vom Anwender über den *lock()* Befehl initiiert werden. Um *Locked* zu verlassen ist ein *reset()* notwendig, welcher den Skill wieder in *Idle* zurückversetzt.

- **Idle:**

Der Skill ist zur Ausführung bereit. Er wurde somit initialisiert und es liegen keine Fehler in der Maschine vor, die eine Ausführung verhindern würden. *Idle* stellt den Grundzustand dar, in welchem sich jeder Skill befinden sollte, sobald die zugehörige Ressource

hochgefahren und initialisiert wurde. Von *Idle* aus kann der Skill entweder über einen *start()* Befehl ausgeführt oder über *lock()* wieder gesperrt werden.

- **Suspended:**

Die Ausführung des Skills wurde vorübergehend unterbrochen. Dies kann beispielsweise automatisch durch prozessbedingtes Warten ausgelöst werden oder manuell durch den Nutzer während der Ausführung (*Executing*) mittels *suspend()* ausgelöst werden. Eine entsprechende Wiederaufnahme ist durch Ausführung von *unsuspend()* möglich bzw. erfolgt automatisch, sofern der Zustand durch prozessbedingtes Warten erreicht wurde. Auch ein Materialmangel oder Stau am Ausgang führt automatisch in den Zustand *Suspended*. Soll die Skillausführung nicht wiederaufgenommen werden, so kann über *cleanup()* die Ausführung beendet und direkt in den *Idle* Zustand zurückgekehrt werden.

- **Executing:**

Der Skill befindet sich in der Ausführung. Der Zustandsübergang von *Idle* zu *Executing* erfolgt durch den Aufruf von `start()`. Über `stop()` wird die Ausführung des Skills vollständig beendet und die Ressource kehrt in ihren Ausgangszustand zurück. Eine direkte Wiederaufnahme des Skills ist in diesem Falle nicht möglich.

Es kann dazu kommen, dass die einen Skill bereitstellende Ressource zunächst über den *Idle*-Zustand herausgehende Vorbereitungen treffen muss, um mit der Ausführung zu beginnen. Ebenso kann ein „Herunterfahren“ der Ressource nach der Skill-Ausführung notwendig sein. Um diese Vorgänge umzusetzen können Subzustände des Zustands *Executing* verwendet werden:

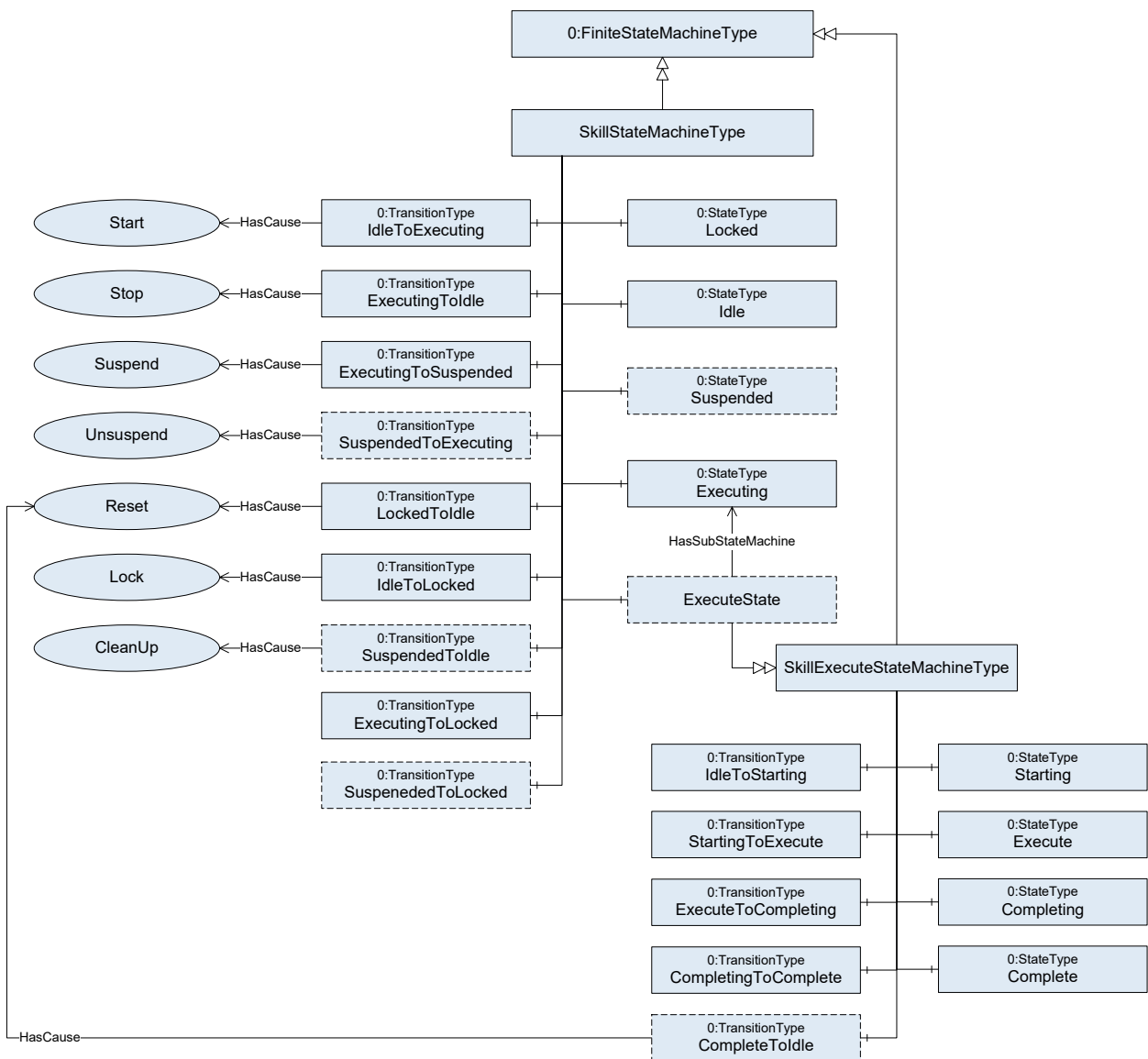
- **Starting:** Beschreibt die unmittelbare Vorbereitung zur Ausführung des Skills. Bei Maschinen kann dies beispielsweise das Hochfahren von Antrieben beinhalten.
- **Execute:** Beschreibt die eigentliche produktive Ausführung der Skills innerhalb der Sub-StateMachine.
- **Completing:** Beschreibt den unmittelbar nach der produktiven Ausführung vorliegenden Zustand, in welchem sich die Maschine auf den darauffolgenden *Complete* Subzustand bzw. letztlich die Rückkehr zum Hauptzustand *Idle* vorbereitet. Bei Maschinen kann dies beispielsweise das Herunterfahren der Antriebe beinhalten.
- **Complete:** Die Skillausführung wurde korrekt abgeschlossen. Der Zustandsübergang von *Complete* zu *Idle* kann je nach Anwendungsfall entweder durch einen automatischen Zustandsübergang (SC) oder

über einen manuellen `reset()` erfolgen. Das Verbleiben im *Complete* Zustand kann dazu genutzt werden, beispielsweise die korrekte Ausführung des Skills zustandsbasiert zu überwachen. Dies kann insbesondere sinnvoll sein, wenn es sich um Prozesse mit hohen Taktzeiten handelt (z.B. Batchprozesse) und nach erfolgter Skill-Ausführung eine weitere Nachbereitung benötigt wird, bevor der Skill wieder bereit (*Idle*) ist. Dies können beispielsweise auch manuelle Entnahmeprozesse des Produkts sein.

Da die einzelnen Zustände verschiedene Bedeutungen haben können, je nachdem, ob sie automatisch oder durch den Client ausgelöst werden, ist es erforderlich die Ursache des Übergangs zu überwachen. Hat ein Client beispielsweise kein Signal gesendet und der Skill befindet sich in *Suspended*, kann beispielsweise ein Materialmangel oder Stau am Ausgang vorliegen. Greifen mehrere Clients auf die State Machine zu, so können die Zustandsübergänge auch über Events überwacht werden oder es erfolgt eine Erweiterung um zusätzliche Substates, welche automatische und vom Nutzer bzw. Client ausgelöste Zustandsübergänge unterscheiden.

Ein Vorschlag für die Implementierung der State Machine ist als *SkillStateMachineType* in Abbildung 8 dargestellt. Innerhalb des *SkillType* aus Abbildung 6 wird dann eine *SkillStateMachine* von diesem Typ erstellt.

Abbildung 8:
OPC UA Informationsmodell des SkillStateMachineType



Mapping der Zustände

Da sich Skills nur auf die eigentliche Ausführung von Produktionsprozessen beziehen, ist die entsprechende State Machine ebenso nur Prozessbezogen. Ressourcen können sich jedoch abgesehen von der eigentlichen Prozessausführung in einer Vielzahl von Zuständen und Applikationsmodi befinden. Diese werden in der Companion Specification *OPC UA for Machinery Part 1- Basic*

Building Blocks (OPC 40001-1) beschrieben. Abbildung 9 zeigt die in der *OPC UA for Machinery* definierten Maschinenzustände.

Neben diesen *MachineryItemState* wurde auch der *MachineryOperationMode* (siehe Abbildung 10) definiert, welcher primär angibt, in welchem Operationsmodus die Maschine durch den Nutzer versetzt wurde.

Abbildung 9:

MachineryItemState aus OPC UA for Machinery Part 1- Basic Building Blocks

Not Available	The unit is not available and does not perform any activity*. (e.g. Switched Off, in Energy Saving Mode)
Out of Service	The unit is not functional and does not perform any activity*. (e.g. Error, Switched Off)
Not Executing	The unit is available and functional and does not perform any activity*. It waits for an action from outside to start or restart an activity*.
Executing	The unit is available and functional and is actively performing an activity* (pursues a purpose).

Abbildung 10:

MachineryOperationMode aus der OPC UA for Machinery Part 1

✕ None	There is currently no <i>MachineryOperationMode</i> available for the <i>MachineryItem</i> .
✕ Maintenance	<i>MachineryItem</i> is set into maintenance mode with the intention to carry out maintenance or servicing activities of the <i>MachineryItem</i> .
▢ Setup	<i>MachineryItem</i> is set into setup mode with the intention to carry out setup, preparation or postprocessing activities of a production process.
⚙ Processing	<i>MachineryItem</i> is set into processing mode with the intention to carry out the value adding activities .

Abbildung 11:

Mapping zwischen Machine State, Application Mode und Skill State

Example	Machinery Item State	Machinery Operation Mode	Skill A State	Skill X State
Non Scheduled Time	any	✕ None	Locked	Locked
Unscheduled Downtime	Out of Service	⚙ Processing	Locked	Locked
Scheduled Downtime	Out of Service	✕ Maintenance	Locked	Locked
Engineering Time	any	▢ Setup	Locked	Locked
Standby Time	Not Executing	⚙ Processing	Idle / Suspended	Idle / Suspended / Locked
Productive Time	Executing	⚙ Processing	Executing	Locked / Idle / Suspended / Executing

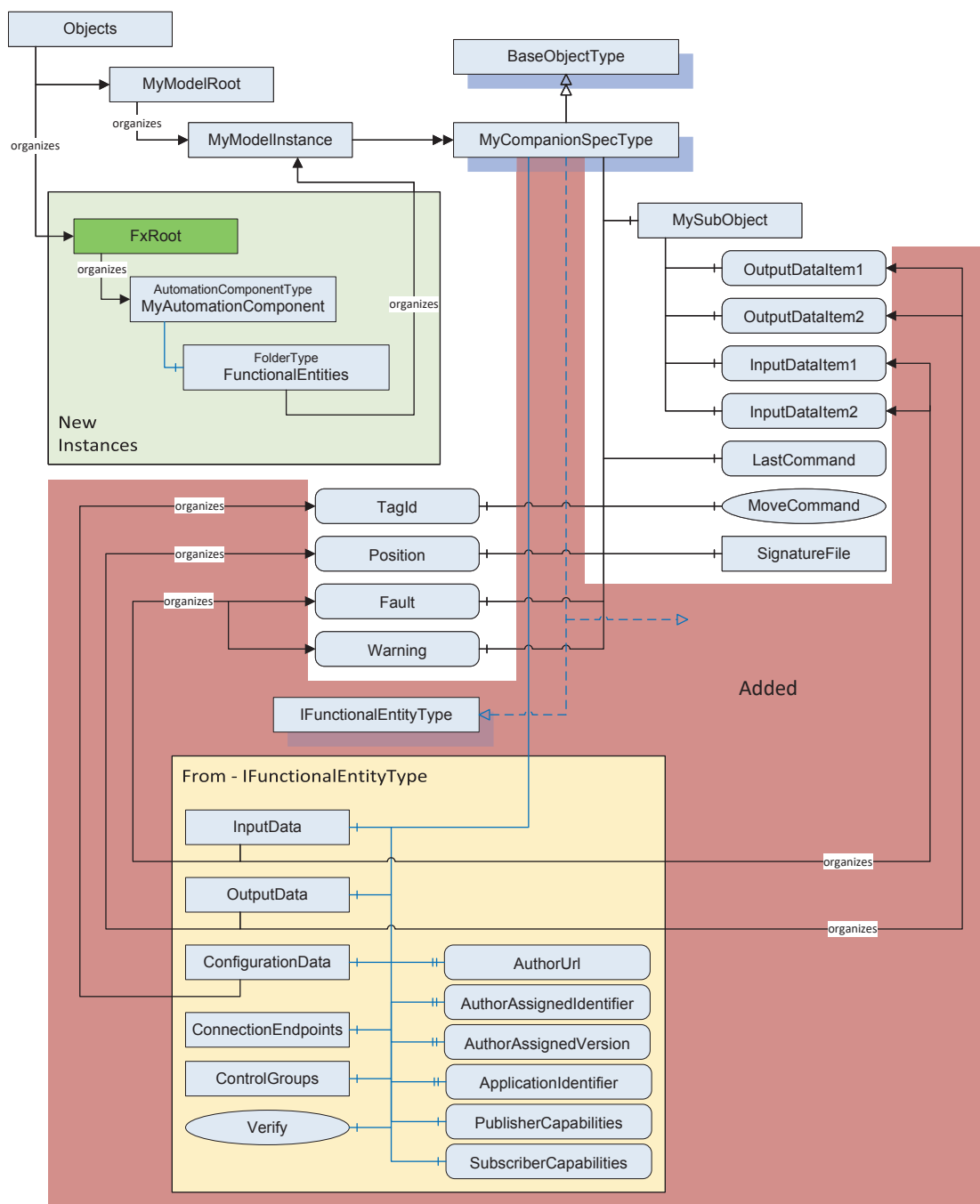
Der *MachineryItemState*, *MachineryOperationMode* und der *SkillState* stehen in direkter Abhängigkeit zueinander, welche exemplarisch in Abbildung 11 dargestellt ist. Skill A State steht in Abbildung 11 für den Zustand eines bestimmten Skills einer Ressource und Skill X für mögliche weitere Skills, die eine Ressource bereitstellt. Wenn sich die Ressource beispielsweise im *MachineryItemState* „Out of Service“ befindet, so können unabhängig vom *MachineryOperationMode* keine Skills ausgeführt werden, weshalb sich sämtliche Skills im *Locked* Zustand befinden müssen. Als Voraussetzung zur Ausführung der Skills muss sich die Ressource im *MachineryOperationMode* „Processing“ befinden und keinen Fehlerzustand aufweisen („Not Executing“). In diesem Falle sind sämtliche Skills entweder im *Idle* oder *Suspended*, da auch hierbei keine Prozessausführung stattfindet. Besitzt die Ressource mehrere Skills, welche nicht gleichzeitig ausgeführt werden können, so müssen sich, sobald ein Skill in *Executing* (oder einem der Subzustände) oder in *Suspended* befindet, alle anderen nicht gleichzeitig ausführbaren Skills im *Locked* Zustand befinden.

Die Abhängigkeiten der drei State Machines müssen entsprechend in der Applikation vorgesehen und in OPC UA wiedergegeben werden.

Ausführung von Skills über das OPC UA Skill-Interface

Die Ausführung der Skills über das OPC UA Skill-Interface kann je nach Anforderungen hinsichtlich zeitlichem Verhalten und Determinismus unterschiedlich erfolgen. Grundsätzlich kann der Skill über einen Methodenaufruf (*start()*) in der *SkillStateMachine* gestartet werden. Der Methodenaufruf muss dabei entsprechend mit dem in der Steuerung implementierten Skill verknüpft werden, beispielsweise als FB oder in einer proprietären Programmiersprache. Ebenso ist es wichtig, dass die *SkillStateMachine* im OPC UA Skill-Interface stets den Zustand des tatsächlich implementierten Skills widerspiegelt (vgl. Abbildung 4). Da die TCP/IP-basierte Client-/Server Verbindung von OPC UA jedoch kein deterministisches Verhalten aufweist, ist dieser Aufruf für synchronisierte Prozesse ungeeignet. Aus diesem Grund könnten hierfür auf Konzepte der OPC UA FX (Field eXchange)-Arbeitsgruppe zurückgegriffen werden, im Speziellen die *Functional Entities*, welche auf Basis von OPC UA-Pub/Sub für die Controller-to-Controller-Kommunikation auf Feldebene geeignet sind. Diese sind durch die grundsätzlichen Datentypen *InputData*, *OutputData* und *ConfigurationData* definiert und können eine entsprechende *organizes*-Referenz auf andere Objekte im Informationsmodell besitzen (vgl. Abbildung 12).

Abbildung 12:
Beispielhafte Verwendung von FunctionalEntities

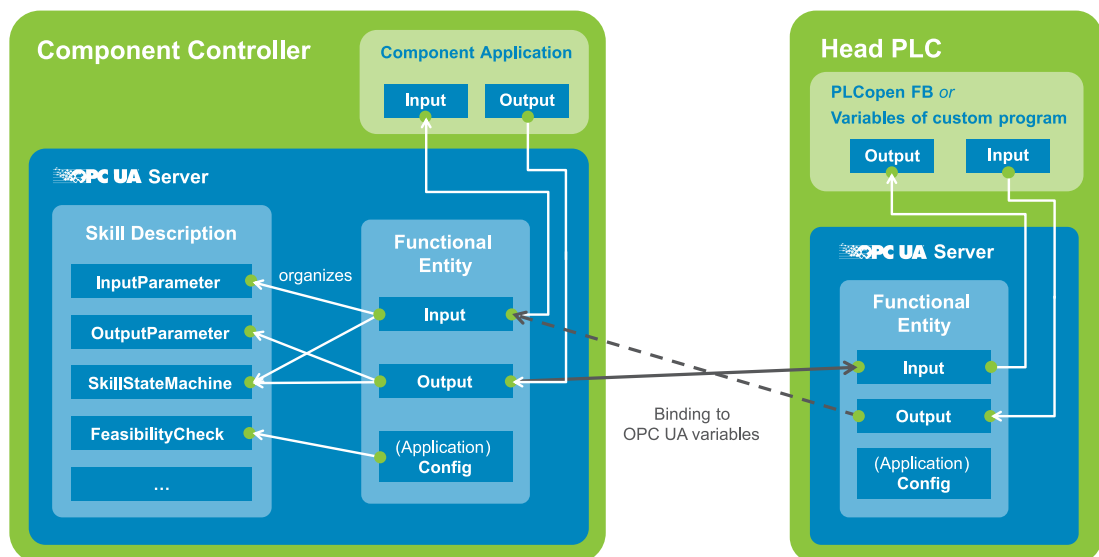


Analog zu dem in Abbildung 12 gezeigten „MySubObject“, können auch die Input- und Outputparameter von Skills entsprechend mit einer *Functional Entity* verknüpft werden, um eine Controller-to-Controller-Kommunikation mit OPC UA-Pub/Sub auf Basis der Skills aufzubauen. Ein Vorschlag hierfür ist in Abbildung 13 dargestellt. Die Inputs und Outputs der Functional Entity werden wiederum mit den Ins und Outs der Component Application verknüpft und sind somit unmittelbar mit dem beispielsweise als FB implementierten Skill verbunden. Auf Seiten des Controllers, welcher die Ablaufsteuerung übernimmt, sind ebenfalls Functional Entities implementiert, welche mit dem entsprechenden Ablaufprogramm verknüpft sind. Über das *Skill-Informationsmodell*

können die entsprechenden Skills aufgefunden und verknüpft, sowie mithilfe der *FunctionalEntity* eine unmittelbare Kommunikation zwischen den Controllern mit entsprechenden Verknüpfungen in die Programme aufgebaut werden. Die Pub/Sub-Verbindung zwischen den *FunctionalEntity* ermöglicht dabei eine deutlich schlankere und direktere Kommunikation als der Methodenaufruf über die Client/Server-Verbindung und könnte in Verbindung mit Time-Sensitive Networking (TSN) eine echtzeitfähige Kommunikation ermöglichen. Der Skillaufruf mittels *FunctionalEntity* stellt jedoch lediglich einen Konzeptvorschlag dar, welcher im Rahmen eines Expertengremiums erarbeitet, jedoch noch nicht in der Praxis validiert wurde.

Abbildung 13:

Möglicher Aufruf von Skills über das Konzept der FunctionalEntity



3.5 Vorgehen beim Einsatz von Fähigkeiten

Mithilfe der in Absatz 3.1 - 3.4 dargestellten Modelle und Technologien können Fähigkeiten nun aus verschiedenen Rollen heraus angewandt werden. Im Folgenden wird nun das empfohlene Vorgehen aufgezeigt wie Fähigkeiten aus Sicht der Hersteller von Ressourcen, aus Sicht von Systemintegratoren bzw. Maschinenbauern sowie von Endanwendern konkret genutzt werden könnten.

Vorgehen aus Sicht von Ressourcenherstellern:

Als Hersteller von Ressourcen ist es von zentraler Wichtigkeit, potenziellen Kunden die Beschreibung von Capabilities und ihre Zusicherungen zur Verfügung zu stellen. Ebenso wichtig ist die Zurverfügungstellung der in den Ressourcen implementierten Skills über die entsprechenden Interfaces. Im Folgenden sollen vor allem die Hersteller von Einzelkomponenten betrachtet werden, wie beispielsweise Hersteller von Achsen, Greifern und Komponenten ähnlicher Größe. Hersteller ganzer Maschinen und Anlagen können beim Einsatz von Fähigkeiten eher mit Systemintegratoren gleichgesetzt werden. Das Vorgehen hierfür wird im darauffolgenden Abschnitt beschrieben.



1. Im ersten Schritt bestimmt ein Ressourcenhersteller, welche Capabilities seine Ressource erfüllen muss. Hierfür könnte die entsprechende, in Absatz 3.3 beschriebene Ontologie verwendet werden. Sofern auch kombinierte Ressourcen angeboten werden (beispielsweise eine aus mehreren Achsen bestehende Pick-and-Place-Einheit), so sollten sowohl für die einzelnen als auch für die kombinierten Ressourcen die entsprechenden Capabilities mit der Ontologie verknüpft werden. Weiterhin ist es möglich, dass Kunden spezielle Capabilities fordern, welche noch nicht in der Ontologie vorhanden sind. Auch solche können an dieser Stelle den Ressourcen entsprechend zugeordnet werden.
2. Damit eine herstellerunabhängige Bestimmung geeigneter Ressourcen möglich ist, müssen im nächsten Schritt die spezifischen Zusicherungen der Ressource beispielsweise aus den zugehörigen Datenblättern extrahiert werden. Typischerweise haben bestimmte Ressourcenklassen identische Capabilities (ein linearer Antriebsstrang z.B. „linear Bewegen“) unterscheiden sich je nach konkretem Typ in den Zusicherungen (z.B. bezüglich ihrer Traglast, Arbeitsraum/Hub, etc.). Diese Zusicherungen werden auf Typebene festgelegt und als zugesicherte Capability über das Produktportal der Hersteller angeboten.
3. Um nun einen nahtlosen Übergang in die Inbetriebnahmephase zu gewährleisten, müssen die Capabilities entsprechend als Skill implementiert werden. Die Skill-Implementierung erfolgt auf Basis der gerätespezifischen Steuerung, sofern die Komponente über einen eigenen Controller verfügt. Anderenfalls müssen entsprechende Programmbausteine wie z.B. entsprechender SPS code in Form von FBs für die Implementierung der Skills bereitgestellt werden.

4. Im letzten Schritt erfolgt die Bereitstellung des Skill-Interface innerhalb von OPC UA. Hier gilt analog zur Skill Implementierung, dass das entsprechende Skill Informationsmodell direkt auf der Komponente als OPC UA Server bereitgestellt und mit dem implementierten Skill verknüpft wird. Alternativ ist das Modell als Nodeset vorhanden und kann in die übergeordnete Steuerung importiert sowie anschließend mit dem standardisierten Steuerungscode verknüpft werden (vgl. Kapitel 3.4)

Vorgehen aus Sicht von Systemintegratoren:

Als Systemintegrator bzw. Maschinenhersteller ist es wichtig sowohl hinsichtlich der eigenen Kunden als auch gegenüber den Zulieferern von Ressourcen zu agieren. Dabei gilt es vor allem die Kundenanforderungen in eine mögliche Automatisierungslösung zu übersetzen und diese auf Basis von Capabilities zu beschreiben, welche dann für die Ressourcenauswahl und die skillbasierte Inbetriebnahme genutzt werden können. Da hier nur auf den Unterschied zu einer klassischen Planung eingegangen wird, soll das Vorgehen gleich mit den Capabilities beginnen.



1. Im ersten Schritt müssen die möglichen skizzierten Lösungen bzw. der vom Kunden geforderte Produktionsprozess in Form von ressourcenneutralen geforderten Capabilities formuliert werden (vgl. Abbildung 2). Dabei ist zu unterscheiden, ob der Kunde produkt- oder prozessbezogene Capabilities verwendet hat. Wie bereits in Kapitel 2.1.1 beschrieben, kann es je nach Domäne in diesem Schritt notwendig sein die produktbezogenen geforderten Capabilities in prozessbezogene zu übersetzen. Eine wesentliche Aufgabe des Systemintegrators ist dabei auch die Zerlegung und Komposition der Capabilities auf Basis der Ontologie. Gerade bei stark produktbezogenen Capabilities kann eine Vielzahl aufeinanderfolgender oder parallel ausgeführter Capabilities erforderlich sein, um die übergeordnete Aufgabe zu erreichen. Aus der Ontologie kann entnommen werden, welche Capabilities miteinander kombiniert werden können (vgl. Kapitel 3.3), wie die bereits erwähnte Pick-&-Place-Capability, welche sich aus mehreren Bewegungen sowie einer Greifen Capability zusammensetzt.
2. Damit diese Capabilities auch einen tatsächlichen Mehrwert leisten, ist es im nächsten Schritt erforderlich die Anforderungen mithilfe der entsprechenden Properties möglichst genau zu beschreiben. Nur so kann die Ressourcenauswahl gut eingegrenzt werden. Wird beispielsweise eine „Bewegen“-Capability gefordert, so gibt es eine Vielzahl möglicher Ressourcen, welche durch eine Beschreibung konkreter Anforderungen in Form von Properties, wie beispielsweise dem geforderten Bewegungsraum, den benötigten Freiheitsgraden oder auch der notwendigen Verfahrensgeschwindigkeit und Genauigkeit weiter spezifiziert werden können.

3. Im dritten Schritt können mithilfe der ressourcenneutralen geforderten Capabilities nun passende zugesicherte Capabilities mit ihren dahinterliegenden Ressourcen (vgl. Abbildung 2) durch das Matching gefunden werden. Eine Vorauswahl könnte hierbei auch toolgestützt erfolgen, sofern die Hersteller der Ressourcen entsprechende Tools anbieten oder auch über eine herkömmliche Anfrage. Vorteil ist dabei, dass die Hersteller der Ressourcen sowohl einen einheitlichen Detaillierungsgrad der Anforderungen erhalten und zum anderen eine lösungsneutrale Beschreibung erhalten, welche sie nicht bezüglich der anzubietenden Ressourcen einschränkt. Anstelle einer bereits lösungsbezogenen Anforderungsspezifikation, welche beispielsweise bereits einen konkreten Antriebsstrang vom Typ X mit den Kräften Y fordert, wird durch die geforderte Capability lediglich die Bewegung mit ihren Properties vorgegeben. Diese kann wiederum von einer beliebigen Kinematik mit einem beliebigen Wirkprinzip erreicht werden. Damit dieses Vorgehen jedoch in Summe funktioniert, ist grundsätzlich ein einheitlicher Zugriff auf die modellierten zugesicherten Capabilities sämtlicher am Markt angebotener Ressourcen erforderlich.
4. Sind die passenden Ressourcen nun identifiziert, erfolgt entsprechend die Planung des konkreten Systemaufbaus sowie die Beschaffung der Ressourcen und anschließende Inbetriebsetzung. An dieser Stelle erfolgt der Übergang der Capabilities zu den in den Ressourcen implementierten Skills, sofern diese auch über ein entsprechendes Skill-Interface zur Verfügung gestellt werden. Nun kann eine ressourcenübergreifende Steuerapplikation erstellt werden. Aufgrund der einheitlichen Interfaces der Ressourcen ist das ressourcenspezifische Expertenwissen stark reduziert, was eine Zeitersparnis für die Inbetriebsetzung mit sich bringt. Konkrete Beispiele für eine solche Inbetriebsetzung auf Basis von Skills befinden sich in Kapitel 4.
5. Im letzten Schritt sollten die Skills wiederum zusammengefasst und somit zu kombinierten Skills aggregiert werden, welche dann den Endanwendern ebenfalls über das generische Skill-Interface bereitgestellt werden. Wie tief der Endanwender dabei in die unteren Ebenen des Systems eindringt, hängt dabei vom Anwendungsfall ab. Dabei ist es möglich, dass er sich nur auf die oberste Ebene der kombinierten Skills beschränkt. Ebenso können dem Endanwender gegenüber beispielsweise einzelne Bewegungen Skills auf Ebene der Einzelressourcen angeboten werden. Über ein entsprechendes OPC UA Informationsmodell ließen sich solche Ebenen entsprechend browsen sowie die zugehörigen Skills ansprechen.

Vorgehen aus Sicht von Endanwendern:

Als Endanwender im produzierenden Gewerbe können Fähigkeiten vor allem in der Phase der Anforderungsmodellierung gegenüber Zulieferern genutzt werden (z.B. gegenüber Systemintegratoren, Maschinenherstellern oder auch direkt die Hersteller der automatisierten Ressourcen) sowie im Betrieb zur Produktionssteuerung. Um die in Kapitel 2.2 dargestellten Ziele erreichen zu können, empfiehlt sich das folgende Vorgehen:



1. Im ersten Schritt müssen die für die Herstellung des Produkts notwendigen Prozessschritte identifiziert werden und idealerweise direkt in geforderte Capabilities übertragen werden. Diese können zunächst auf einem hohen Level und damit in der Regel sehr produktbezogen sein (z.B. das bereits erwähnte *Motorhaube montieren*). Sofern aus dem Produktherstellungsprozess jedoch bereits konkretere Einzelprozesse vorhanden sind, ist es sinnvoll die geforderten Capabilities auf Basis der Ontologie auf einem möglichst niedrigen Level zu beschreiben. Anderenfalls ist es wie bereits zuvor beschrieben die Aufgabe des Systemintegrators diese Beschreibung in Zusammenarbeit mit dem Endanwender zu erarbeiten.
Ebenso kann es auch sein, dass Systemintegratoren oder Maschinenhersteller direkt Maschinen und Anlagen mit sehr produktbezogenen zugesicherten Capabilities anbieten (z.B. Sondermaschinen, um ein bestimmtes Produkt herzustellen) in welchem Falle eine weitere Zerlegung der Capabilities nicht erforderlich ist. Ebenso können seitens der Systemintegratoren oder auch beim Endanwender selbst bereits Templates für die Zerlegung von komplexen Capabilities in ihre einzelnen Bestandteile vorhanden sein oder die notwendigen Informationen hierfür können aus der Ontologie entnommen werden.
2. Im zweiten Schritt werden die geforderten Capabilities dem Systemintegrator übergeben, welcher diese dann auf Basis der Kundenanforderungen gerade bei größeren Anlagen, wie im vorherigen Abschnitt beschrieben, in ihre einzelnen untergeordneten Capabilities zerlegen kann. Ziel ist ein direktes Matching mit bereits bekannten und genutzten Ressourcen oder mit Ressourcen die neu bei Herstellern gesucht werden
3. Im dritten Schritt folgt die Beschaffung der entsprechenden Ressourcen, welche entweder als Sonderlösung bei einem Systemintegrator in Auftrag gegeben oder direkt bei Herstellern aufgrund erfolgreicher Matches bestellt werden.
4. Im letzten Schritt müssen die beschafften Ressourcen in das System beim Endkunden eingebunden werden. Hierzu werden die implementierten Skills über ihr Skill-Interface angesprochen und darüber in die übergeordnete Produktionssteuerung eingebunden. Somit wird auch beim Endanwender eine starke Reduktion der Integrationsaufwände erreicht und gleichzeitig wie in Kapitel 2.2 beschrieben eine Stellung von Produktionsweichen aus sämtlichen übergeordneten Systemen während der Produktion möglich.

4 Beispiele für die Umsetzung von Skills mit hilf von OPC UA

4.1 Umsetzung von Skills am Beispiel der Montagetechnik

Das Konzept der Fähigkeiten wurde im Rahmen des VDMA R+A OPC UA Demonstrators bereits für die Automatica 2018 umgesetzt (Zimmermann et al. 2019). Ziel eine komplette Steuerungsarchitektur – von der einzelnen Komponente bis zur Gesamtmaschine – mit Skills auf Basis von OPC

UA abzubilden. Dabei haben verschiedene Hersteller von Automatisierungskomponenten zusammengearbeitet um das herstellerübergreifende Konzept von Fähigkeiten und Skills zu demonstrieren. Der Demonstrator stellt eine Montagezelle für Fidget Spinner dar, welcher diese aus dem Grundkörper, Kugellagern sowie Deckeln montiert.

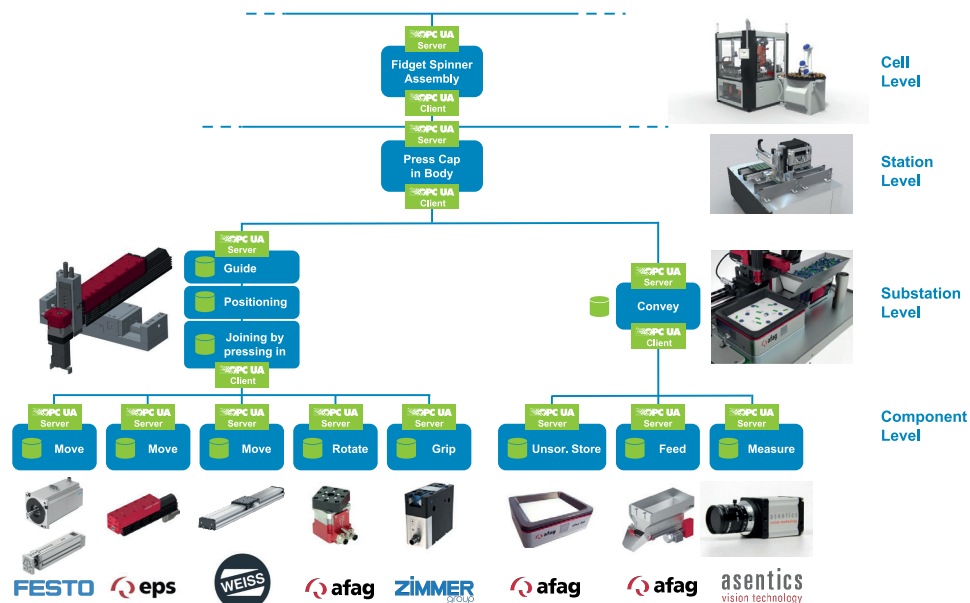
Abbildung 14:
VDMA R+A OPC UA Demonstrator



Die Steuerungsarchitektur mit den verschiedenen Aggregationslevel von Skills ist in Abbildung 15 für die Station zum Einpressen der Deckel für die Kugellager der Fidgetspinner dargestellt. Hierbei lässt sich gut der Übergang von produktspezifischen zu produktneutralen Skills sowie das Zusammenfassen von einzelnen Skills zu kombinierten Skills beobachten. Der übergeordnete Skill der gesamten Zelle ist „Fidget Spinner montieren“ (Fidget Spinner Assembly). Würde die Zelle einem Endanwender zur Herstellung von Fidget Spinners verkauft werden, so wäre dies der Skill der vom Endanwender angesprochen wird. Auf der untersten Ebene (Component Level) befinden sich wiederum einzelne Komponenten wie Achsen und Greifer, welche entsprechend *Bewegen*,

Rotieren oder *Greifen* Fähigkeiten anbieten. Hierbei wurden zwei Arten von Geräten verwendet: Zum einen Geräte mit einem direkt integrierten OPC UA Server und zum anderen Geräte mit klassischen digitalen Feldbussen oder digitalen I/O Schnittstellen. Dies bedeutete, dass für letztere sowohl Skills als auch das Skill-Interface in Form eines OPC UA Servers entsprechend in der nächsthöheren Steuerung mithilfe von CODESYS generiert werden mussten. Die notwendigen Bausteine sowie eine Beschreibung der angebotenen Skills mit ihren im Skill-Interface zu repräsentierenden Parametern kommt dabei direkt von den Herstellern der einzelnen Ressourcen, welche an diesem Demonstratorprojekt beteiligt waren.

Abbildung 15:
Skill-Architektur des Demonstrators



Aufgabe des Systemintegrators war es nun die von den Einzelkomponenten angebotenen Skills so zu kombinieren, dass letztlich der *Fidget Spinner montieren* Skill bereitgestellt werden kann. Im ersten Schritt wurden daher „Substationen“ gebildet, welche die elementaren prozessbezogenen Skills zu höherwertigen kombinieren. Ein Beispiel ist hier der *Positionieren* Skill, welcher sich aus mehreren *Linear Bewegen* Skills sowie *Greifen* zusammensetzt. Dabei kann diese Kombination an Grundskills verschiedene höherwertige Skills bereitstellen, welche aus der Ontologie entnommen werden können. Hierzu zählen noch der *Führen* sowie der *Fügen durch Einpressen* Skill.

Zusammen mit dem *Transportieren* Skill einer weiteren Substation können sie zum nun produktbezogenen Skill *Deckel in Grundkörper einpressen* zusammengesetzt werden, welche einen Teil des obersten *Fidget Spinner Montieren* Skills darstellt. Die Aggregationen der Skills finden in Steuerungen statt, welche die unterlagerten Komponenten per OPC UA Client ansprechen und nach oben hin einen Server mit dem aggregierten Skill bereitstellen. Die Ablauflogik für die korrekte Reihenfolge in der Skillausführung kann in jeder aggregierenden Steuerung hinterlegt werden. Ebenso könnte aber auch eine einzelne übergeordnete Steuerung die komplette Aggregation aller Ebenen vornehmen. Hierbei würde zwar die Modularität des Systems verringert werden, jedoch würden auch deutlich weniger Steuerungen benötigt werden. Dem Client-Server Interaktionsprinzip geschuldet, führt der Weg über die Verkettung von Clients und Servern zu Verzögerungen in der Ausführung.

Insgesamt konnten am Demonstratorprojekt verschiedene Anwendungsszenarien gezeigt werden:

- **Vereinfachte Planung und Inbetriebnahme:**

Durch die Verwendung von Capabilities in der Planungsphase, konnte die übergeordnete Capability „Fidget Spinner Montieren“ weiter zerlegt werden, bis ein entsprechendes Matching auf die Einzelressourcen möglich war. Nach Beschaffung der Ressourcen, konnte mithilfe der passenden implementierten Skills sowie des OPC UA Skill-Interface eine deutlich verkürzte Inbetriebnahme erreicht werden, da weder proprietärer Steuerungscode noch herstellereigenspezifische Steuerungsschnittstellen genutzt werden mussten.

- **Flexibilität und Wandlungsfähigkeit:**

Durch den Einsatz von Skills war es möglich, schnell Ressourcen zu ändern und an neue Produkte anzupassen. Dies kann zum einen auf Basis der Skill-parameter erreicht werden, sofern keine konstruktiven Änderungen für die neue Produktvariante notwendig sind oder sofern diese Änderungen nur Anbauteile betreffen (z.B. das Ändern der Greifbacken und die anschließende neue Parametrierung des Greifen Skills). Dies geht jedoch nur im Rahmen des Flexibilitätskorridors der jeweiligen Ressourcen, da beispielsweise der Greifer eine endliche Spannweite oder Kraft hat. Daher wurde zum anderen auch der unkomplizierte Austausch ganzer Ressourcen am Beispiel der Greifer gezeigt. Hierbei konnten sowohl Greifer verschiedener Hersteller für die gleiche Montageaufgabe verwendet als auch Greifer gewechselt werden, wenn beispielsweise ein neues Produkt höhere Spannweiten oder höhere Kräfte beim Greifvorgang erforderlich machten. Die grundsätzliche Abfolge, in welcher die Skills aufgerufen werden, bleibt hierbei identisch, lediglich die Skillparameter müssen über das generische Skill-Interface angepasst werden. Voraussetzung hierfür ist die mechanische Kompatibilität sowie die passende Medienversorgung.

4.2 Umsetzung von Skills am Beispiel der Fertigungstechnik

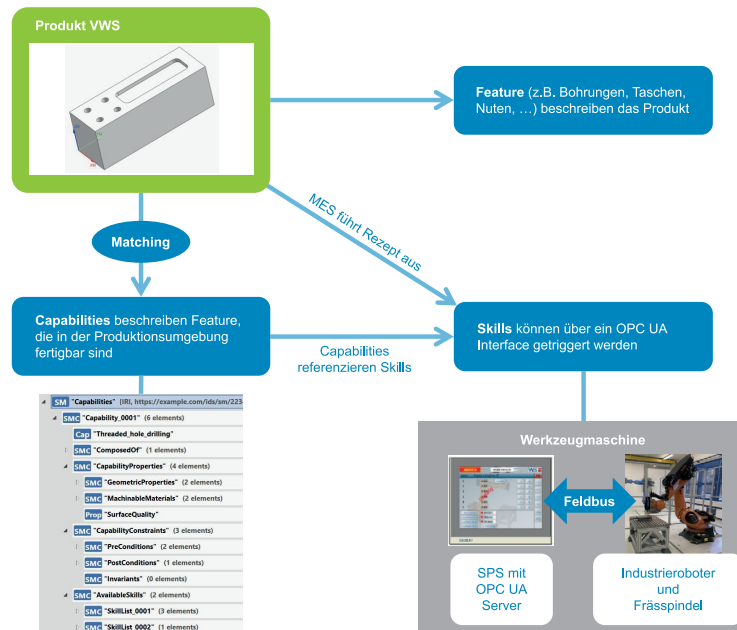
Für das Beispiel der Fertigungstechnik wird ein Szenario beschrieben, in dem ein Kunde ein individuelles Bauteil konstruiert und dieses mit Hilfe von Skills gefertigt wird. Das CAD-Modell, sowie zugehörige Metadaten, werden als Angebotsanfrage an ein Unternehmen gesendet. Zur spezifischen Beschreibung des Produktes sind Informationen über die Stückzahl, das Material, die Außenmaße und Form sowie die konstruierten geometrischen Features (z.B. Nuten, Taschen, Bohrungen, Fasen, ...) spezifiziert. Jedes der Features wird durch mehrere Parameter spezifiziert und geometrisch eindeutig beschrieben. Mit Hilfe einer automatisierten Featureextraktion können die Produkthanforderungen aus der CAD-Datei erstellt und auf die in der Produktionsumgebung vorhandenen Capabilities gemappt werden. So wird z.B. für das geometrische Feature „Tasche“ eine Capability zur Fertigung der Tasche benötigt. Um das Matching zu vereinfachen ist es sinnvoll, die Definition von Capabilities produktbezogen an geometrischen Features zu orientieren. Die benötigten Capabilities zur Fertigung eines Features verweisen wiederum auf einen oder mehrere Skills, die zur Fertigung sequenziell ausgeführt werden müssen. Im Gegensatz zur Capability Beschreibung im Engineering ist die Capability im Lebenszyklus der Nutzung einer Ressource nicht zwangsläufig statisch. Über Vergangenheitsdaten aus der Produktion lässt sich die Capability weiter spezifizieren und updaten. So wird ermöglicht, dass sich die Komplexität der Fertigung, die oftmals von der Kombinatorik aus Maschine, Werkzeug und Einspannung abhängt, abbilden lässt.

Die Ausführbarkeit eines Skills mit den durch das Feature vorgegebenen Parametern wird vor der Fertigung geprüft. Dies erfolgt mithilfe des FeasibilityChecks, welcher für Skills in der Fertigung beispielsweise durch Kalkulationen und Simulationen die Umsetzbarkeit des Fertigungsprozesses

hinsichtlich Kollisionen, benötigter Werkzeuge und Qualitätsansprüchen überprüft. Das Ergebnis des FeasibilityChecks ist die Bestätigung einer realisierbaren Trajektorie, die Werkzeug sowie Spannmittelwahl. Weitere Ergebnisse des FeasibilityChecks, wie die benötigte Zeit, Kostenaufwand oder der Ressourcenverbrauch, dienen zum einen als Entscheidungsgrundlage zur Auswahl einer geeigneten Ressource für den späteren Fertigungsprozess und zum anderen als Grundlage zur automatisierten Angebotserstellung. Aufgrund dessen wird der FeasibilityCheck üblicherweise bereits in der Planungsphase ausgeführt, die zeitlich bereits weit vor der Ausführung des Skills liegt. Der FeasibilityCheck ist somit als langfristige und generelle Machbarkeitsprüfung zu verstehen. Die Ergebnisse sollten für die spätere Wiederverwendung zwischengespeichert werden. Nachdem ein möglicher Fertigungsprozess aus einem oder mehreren Skills identifiziert wurde, erhält der Kunde ein entsprechendes Angebot.

Nach Auftragseingang wird ein spezifischer, zeitlich abgestimmter Fertigungsplan generiert. Vor Ausführung eines Skills wird mit Hilfe des PreconditionChecks die kurzfristige Ausführbarkeit der zur Fertigung notwendigen Skills angefragt. Die Ressourcen prüfen hierfür die bereits im Vorfeld berechneten Ergebnisse des FeasibilityChecks auf Gültigkeit. Zusätzlich wird nun auch geprüft, ob sich z.B. das korrekte Spannmittel und Werkzeug in der Maschine befinden. Sollte der PreconditionCheck positiv ausfallen, kann das Werkstück in die Zelle geladen werden. Zur Ausführung des Skills prüft die Ressource selbständig die im Feasibility-Check und PreconditionCheck kalkulierten Daten noch einmal auf ihre Gültigkeit und nutzt diese für die Abarbeitung.

Abbildung 16:
Fertigung von Features in einer Fräspannung.



Als Werkzeugmaschine wird in der Umsetzung ein Roboterarm mit aufgesetzter Frässpindel genutzt. Zusätzlich ist der Robotersteuerung eine SPS als Adapter vorgeschaltet, die das Skill-Interface über einen OPC UA Server bereitstellt und verschiedene Berechnungen sowie verschiedene Steuerungsaufgaben der Peripherie übernimmt. Über einen Feldbus werden Daten zwischen SPS und Robotersteuerung ausgetauscht. Die Skills können von einem Steuerungssystem über einen OPC UA Client aufgerufen werden.

Mehrere Skills sind auf der SPS als Funktionsbaustein implementiert (Rechtecktasche fräsen, Bohren, Kreistasche fräsen, Nut fräsen). Die relevanten, internen Variablen (Parameter, Aktueller Zustand, Ergebnisdaten etc.) sind explizit für den OPC UA Zugriff freigegeben. Ebenso sind dem Funktionsbaustein zugehörige Methoden über den OPC UA Server aufrufbar. Der FB wiederum beinhaltet einen Zustandsautomaten für den FeasibilityCheck sowie einen weiteren Zustandsautomaten für die Ausführung des Skills, sodass der FeasibilityCheck und die Skill-Ausführung par-

allel und unabhängig ablaufen können. Der Pre-conditionCheck ist als Methode implementiert. Die Zustandsautomaten können über die OPC UA Methoden des Skill-Interface gesteuert werden, wobei die Zustandsmaschinen zusätzlich durch den Zustand der (Gesamt) Maschine beeinflusst werden (z.B. ist bei einem Fehlerzustand der Maschine der Skill automatisch im Locked-Zustand).

Befindet sich der Skill im Wartezustand (Idle) und ruft ein Client die Start-Methode auf, so werden die nötigen und zuvor kalkulierten Parameter von der SPS an den Roboterarm übertragen. Sobald die Bearbeitung beginnt, wechselt auch der Skill in den Zustand Executing. Auf der Robotersteuerung wird ein parametrisiertes Programm für eine Kreistasche abgearbeitet, das dem eines CNC-Zyklus auf standardmäßigen CNC-Werkzeugmaschinen ähnelt. Sollte zukünftig die Synchronisation von Skills in Echtzeit möglich werden (mit der OPC UA FX Specification), so könnten auch hier zur Bewegung der Achsen entsprechende Skills zum Einsatz kommen.

5 Ausblick

Insgesamt bietet das Konzept der Fähigkeiten ein großes Potential viele Vorgänge von der Planungsphase über die Inbetriebnahme bis hin zur flexiblen Ausführung von Produktionsvorgängen zu vereinfachen. Wenn auch die Konzepte bereits ausgereift sind und in einigen Demonstratoren und Forschungsprojekten bereits validiert werden konnten, so sind noch einige Fragestellungen offen. Diese betreffen insbesondere die Standardisierung der entsprechenden Datenmodelle für Capabilities und Skills, sowie Interaktionsmechanismen zur Steuerung der Skills über das OPC UA Skill-Interface. Derzeit befinden sich bereits einige dieser Datenmodelle in der Entwicklung, wie beispielsweise das Capability Teilmodell für die AAS, welches im Rahmen der IDTA entwickelt wird. Gleichzeitig arbeitet OPC UA FX an finalen Konzepten, wie OPC UA zur Controller-to-Controller

Kommunikation eingesetzt werden kann. Hierbei sind jedoch auch noch Fragen bezüglich der erreichbaren Latenzen offen, gerade in Verbindung mit TSN. Dies wird die Anwendbarkeit zur Steuerung auf Feldebene stark beeinflussen. Gleichzeitig müssen Ontologien aufgebaut werden, welche Herstellern von Ressourcen, Systemintegratoren aber auch Endanwendern die richtige Verknüpfung der Fähigkeiten ermöglicht. Ebenso gilt es auch die OPC UA Informationsmodelle hinter den Skills zu standardisieren, wenn auch dieser Leitfaden bereits einige Vorschläge hierfür aufzeigt. Können die genannten Bausteine auf industriellem Niveau standardisiert und validiert werden, steht einer erfolgreichen und breiten Anwendung von Fähigkeiten nichts im Wege.

Projektpartner / Impressum

VDMA e.V.

Lyoner Str. 18
60528 Frankfurt am Main

Fraunhofer-Institut für Gießerei-, Composite- und Verarbeitungstechnik IGCV

Am Technologiezentrum 10
86159 Augsburg

Forschungskuratorium Maschinenbau e.V. (FKM)

Lyoner Str. 18
60528

Projektleitung

Johannes Olbort, VDMA
Patrick Zimmermann, Fraunhofer IGCV

Beteiligte Unternehmen, Hochschulen und Vereine

Jürgen Bock, Technische Hochschule Ingolstadt
Christian Diedrich, Otto von Guericke Universität Magdeburg
Rüdiger Fritz, SAP SE
Stephan Grimm, Siemens AG
Jesko Hermann, Technologie-Initiative SmartFactory KL e.V.
Johannes Hoos, Festo SE & Co. KG
Anna Kernspecht, Volkswagen AG
Jonathan Nussbaum, TU Kaiserslautern, WSKL
Magnus Volkmann, TU Kaiserslautern, WSKL
Andreas Wagner, TU Kaiserslautern, WSKL
Etienne Axmann, VDMA
Jakob Albert, VDMA

Layout

Gabriela Neugebauer, VDMA DesignStudio

Bildnachweis

Titelbild: Shutterstock

Grafiken

VDMA
Fraunhofer IGCV

Erscheinungsjahr

2022

Copyright

Fraunhofer IGCV
VDMA

Quellenverzeichnis

DIETRICH ET AL. 2022

Dietrich, C.; Belyaev, A.; Bock, J.; Grimm, S.; Hermann, J.; Klausmann, T.; Köcher, A.; Meixner, K.; Peschke, Jörn; Schleipen, Miriam; Schmitt, Siwara; Volkmann, Magnus; Watson, Kym; Winter, Michael; Zimmermann, Patrick: Discussion Paper - Information Model for Capabilities, Skills & Services.

DOROFEEV & ZOITL 2018

Dorofeev, K.; Zoitl, A.: Skill-based Engineering Approach using OPC UA Programs. (Hrsg.): INDIN, 16th International Conference on Industrial Informatics (INDIN). 2018/07: IEEE 2018.

GRIMM 2009

Grimm, S.: Semantic Matchmaking with Nonmonotonic Description Logics. Burke: IOS Press 2009. ISBN: 9781614993353. (Studies on the Semantic Web v.1).

HAMMERSTINGL 2020

Hammerstingl, V. G.: Steigerung der Rekonfigurationsfähigkeit von Montageanlagen durch Cyber-physische Feldgeräte. (Dissertation). Institut für Werkzeugmaschinen und Betriebswissenschaften, Technische Universität München. München (2020).

HERMANN 2021

Hermann, J.: Dynamische Generierung alternativer Fertigungsfolgen im Kontext von Production as a Service. (Dissertation) (2021).

HERMANN ET AL. 2019

Hermann, J.; Rübel, P.; Birtel, M.; Mohr, F.; Wagner, A.; Ruskowski, M.: Self-description of Cyber-Physical Production Modules for a product-driven manufacturing system. Procedia Manufacturing 38 (2019), S. 291-298.

IDTA 2022A

IDTA: „Capability“ - Registered AAS Submodel Templates. <<https://industrialdigitaltwin.org/en/content-hub/submodels>>.

IDTA 2022B

IDTA: „Control Component Type“ und „Control Component Instance“ - Registered AAS Submodel Templates. <<https://industrialdigitaltwin.org/en/content-hub/submodels>>.

KAGERMANN ET AL. 2013

Kagermann, Henning; Wahlster, Wolfgang; Helbig, Johannes (Hrsg.): Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Abschlussbericht des Arbeitskreises Industrie 4.0. acatech – Deutsche Akademie der Technikwissenschaften e.V. Frankfurt: Büro der Forschungsunion 2013.

KOREN & SHPITALNI 2010

Koren, Y.; Shpitalni, M.: Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems* 29 (2010) 4, S. 130-141.

LI & HORROCKS

Li, L.; Horrocks, I.: A software framework for match-making based on semantic web technology. In: Hencsey, G. (Hrsg.): *Proceedings of the 12th international conference on World Wide Web, the twelfth international conference*. Budapest, Hungary, 5/20/ 2003 - 5/24/2003. New York, NY: ACM 2003, S. 331. ISBN: 1581136803. (ACM Conferences).

LINDEMANN ET AL. 2006

Lindemann, Udo; Reichwald, Ralf; Zäh, Michael (Hrsg.): *Individualisierte Produkte. Komplexität beherrschen, in Entwicklung und Produktion*. Berlin: Springer 2006. ISBN: 3-540-25506-0. (VDI-Buch).

MALAKUTI ET AL.

Malakuti, S.; Bock, J.; Weser, M.; Venet, P.; Zimmermann, P.; Wiegand, M.; Grothoff, J.; Wagner, C.; Bayha, Andreas: *Challenges in Skill-based Engineering of Industrial Automation Systems*. (Hrsg.): ETFA, 23rd International Conference on Emerging Technologies and Factory Automation (ETFA). 2018/09: IEEE 2018.

MOTSCH ET AL. 2021

Motsch, W.; Dorofeev, K.; Gerber, K.; Knoch, S.; David, A.; Ruskowski, M.: *Concept for Modeling and Usage of Functionally Described Capabilities and Skills*. (Hrsg.): 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021 IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA). Vasteras, Sweden, 07.09.2021 - 10.09.2021: IEEE 2021, S. 1-8. ISBN: 978-1-7281-2989-1.

NYHUIS 2008

Nyhuis, Peter (Hrsg.): *Wandlungsfähige Produktionssysteme. Heute die Industrie von morgen gestalten*. Garbsen: PZH Produktionstechnisches Zentrum 2008. ISBN: 9783939026969.

VOLKMANN ET AL. 2021

Volkman, M.; Sidorenko, A.; Wagner, A.; Hermann, J.; Legler, T.; Ruskowski, M.: Integration of a feasibility and context check into an OPC UA skill. *IFAC-PapersOnLine* 54 (2021) 1, S. 276-281.

WIENDAHL ET AL. 2004

Wiendahl, H.-P.; Gerst, D.; Keunecke, L.: *Variantenbeherrschung in der Montage*. Berlin, Heidelberg: Springer Berlin Heidelberg 2004. ISBN: 978-3-642-62372-1.

ZIMMERMANN ET AL. 2019

Zimmermann, P.; Axmann, E.; Brandenbourger, B.; Dorofeev, K.; Mankowski, A.; Zanini, P.: *Skill-based Engineering and Control on Field-Device-Level with OPC UA*. In: *Institute of Electrical and Electronics Engineers et al. (Hrsg.): IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain, 9/10/2019 - 9/13/2019. Piscataway, NJ: IEEE 2019, S. 1101-1108. ISBN: 978-1-7281-0303-7.

Hinweis

Die Verbreitung, Vervielfältigung und öffentliche Wiedergabe dieser Publikation bedarf der Zustimmung des VDMA und seiner Partner. Auszüge der Publikation können im Rahmen des Zitatsrechts (§51 Urheberrechtsgesetz) unter Beachtung des Quellenhinweises verwendet werden.

VDMA

Information Interoperability

Lyoner Straße 18
60528 Frankfurt am Main
Internet www.vdma.org

Kontakt

Johannes Olbort
Telefon +49 69 6693-1368
E-Mail johannes.olbort@vdma.org

vdma.org

Förderhinweis

Das Projekt „Interoperable Interfaces for Intelligent Production – II4IP“ wird mit Mitteln des Bundesministeriums für Wirtschaft und Klimaschutz (BMWK) gefördert.
Durchgeführt wird das Projekt vom Forschungskuratorium Maschinenbau e.V. (FKM) in Kooperation mit dem Verband Deutscher Maschinen- und Anlagenbau e.V. (VDMA) in der Zeit von Februar 2020 bis Januar 2023.