BaSys

Basissystem Industrie 4.2

D-2.11 Capability- und Feasiblity-Interaktion

Dokument-Version	1.0
Datum	18.10.2021
Verbreitungsgrad	Öffentlich
Projekt	BaSys 4.2
Förderkennzeichen	01 IS 190 022
Laufzeit	1.7.2019 – 30.6.2022

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Änderungsverzeichnis

Änderung			Geänderte Abschnitte	Autoren	Beschreibung
Nr.	Datum	Version			
1	20.04.2020	0.1	alle	Ch. Diedrich	Inhaltsverzeichnis, Inhalte
2	06.08.2020	0.2	3.1	T. Miny	API-basierte Umsetzung
3	04.09.2020	0.3	2	S. Schmitt	Glossar, Einleitung
4	07.10.2020	0.4	1, 2	S. Schmitt	Use Cases, Rollenspezifikation
5	21.10.2020	0.5	3.1.1	Ch. Diedrich	Interaktionssequenz mit Rollenbeschreibung harmonisiert für Use Case 1
6	15.03.2021	0.6	Kurzfassung, alle	S. Schmitt	Kurzfassung hinzugefügt, Glossar bearbeitet, Referenzen bearbeitet, Formatierung.
7	30.08.2021	0.7		Ch. Diedrich	Anpassung Abb. an 2.3 Akteursbezeichnungen Beantwortung offene Kommentare Begründung für I4.0-Sprachumsetzung
8	18.10.2018	1.0	alle	S. Schmitt	Finalisierung und Veröffentlichung des Deliverables

Editoren

Christian Diedrich, ifak eV Magdeburg

Matthias Riedl

Autoren

Christian Diedrich, ifak eV Magdeburg

Torben Miny, RWTH Aachen

Siwara Schmitt, Fraunhofer IESE

Reviewer

Michael Weser, KUKA AG

Philipp Blohm, KUKA AG

Mario Thron, ifak eV Magdeburg

Kurzfassung

Die Produktionsherstellung wandelt sich von einer Massen- zu einer individuellen Produktion, bei der in kleinen Losgrößen gefertigt wird. Um diesem Wandel Rechnung zu tragen, wurde in dem Forschungsprojekt BaSys 4.0 ein Konzept für eine Industrie 4.0 Middleware-Plattform entwickelt. Dieses Konzept wird im Nachfolgeprojekt BaSys 4.2 weiterentwickelt und als Referenzimplementierung in Form der Open Source Lösung Eclipse BaSyx bereitgestellt. Eclipse BaSyx ermöglicht eine rentable, flexible Produktion bis zur Losgröße eins.

Für eine flexible und (teil-) automatisierte Produktionsplanung wurde in BaSys 4.0 der Ansatz des fähigkeitsbasierten kontinuierlichen Engineerings konzipiert. Dabei werden die Funktionen und Dienste von Assets abstrahiert und in Form von semantischen, formalisierten Modellen beschrieben. Mit Hilfe dieser Modelle lässt sich ein Abgleich von geforderten Fähigkeiten zur Herstellung eines Produktes mit den vorhandenen Fähigkeiten von Ressourcen der Fabrikanlage durchführen.

Die Checks des fähigkeitsbasierten kontinuierlichen Engineerings sind Teil von Verhandlungen über die Herstellung von Produkten. In diesem Deliverable werden folgende zwei Verhandlungs-Use Cases betrachtet:

1. Verhandlung über die Herstellung eines Produktes innerhalb eines Produktionsstandortes
2. Production-as-a-Service-Plattform über Firmengrenzen und einzelne Produktionsstandorte hinaus

Gegenstand dieses Deliverables ist die Interaktion, die während der zwei Verhandlungs-Use Cases zwischen den beteiligten Akteuren stattfindet. Akteure sind in diesem Fall sowohl IT-Systeme als auch Geräte und Personen. Der Informationsaustausch wird als Interaktion verstanden. In Kapitel 2.3 sind Akteure und (Meta-)Informationsentitäten spezifiziert, die für die exemplarisch dargestellten Interaktionen der genannten Verhandlungs-Use Cases relevant sind.

Verhandlungen beinhalten die Produkt- und Produktionsschrittbeschreibung sowie die Konditionen, unter denen produziert werden soll. Mittels des fähigkeitsbasierten kontinuierlichen Engineerings lässt sich verifizieren und validieren, ob die Herstellung des geforderten Produktes unter den gegebenen Konditionen realisierbar und rentabel ist. Auch lassen sich konditionelle Abweichungen identifizieren, die zur Produktherstellung in Kauf genommen werden müssen. Beispielhaft wären dies Abweichungen im Zeitplan oder hinsichtlich des Preises, weil z. B. notwendige Ressourcen erst zu einem späteren Zeitpunkt verfügbar sind oder weil die Produktion zu dem gewünschten Preis nicht realisierbar ist.

Die Interaktion kann entweder API- oder Nachrichten-basiert umgesetzt werden. Beide Optionen bauen auf den Arbeiten des Deliverables 2.7 „Checker Framework“ auf und werden in Kapitel 3 näher erläutert.

Die API-basierte Interaktion verwendet das Client-Server-Pattern, wobei ein Server eine API zur Verfügung stellt, die eine Kommunikationsschnittstelle darstellt. Mittels Methodenaufrufe können Akteure miteinander interagieren. Das Kapitel 3.2 beschreibt die drei Methoden Info, Schema und Check, die für Verhandlung über die Produktherstellung spezifiziert wurden.

Die Nachrichten-basierte Interaktion nutzt das semantische Protokoll und die Nachrichten nach der Norm VDI 2193. An den Schnittstellen der einzelnen Akteure werden die VDI 2193-konformen Nachrichten angeboten. Eine nähere Erläuterung des Nachrichten-basierten Ansatzes zur Interaktion wird im Kapitel 3.3 gegeben.

Inhaltsverzeichnis

Glossar

Asset	Entität, die sich im Besitz einer Organisation befindet oder unter der Obhut einer Organisation steht, und die entweder einen vermeintlichen oder tatsächlichen Wert für die Organisation hat. [BMWi Hrsg. (2021)]
Aufgabe	Eine Aufgabe ist ein Prozess aus Anforderungssicht ohne Vorgabe einer Lösung. Bei einer Aufgabe sind ein Anfangszustand und ein Zielzustand festgelegt. Der Übergang zwischen diesen ist unbekannt. Durch Bedingungen an den Übergang kann der Lösungsraum der Aufgabe eingeschränkt werden. [BaSys (2021)]
Broker-Service	Service, bei dem sich Checker an- und abmelden und der eine anfragende Applikation an passende Checker vermittelt. [BaSys (2021)]
Capability (dt. Fähigkeit)	Die implementierungsunabhängige Beschreibung der Funktion einer Ressource zur Erzielung einer bestimmten Wirkung in der physischen oder virtuellen Welt. [übersetzt aus BMWi Hrsg. (2020a)]
Capability Check(-ing)	Kontextfreier Fähigkeitstest einer Ressource, ob sie eine gegebene Aufgabe ausführen kann. [BaSys (2021)] Ein formelles Verfahren zur Bewertung der Erfüllung einer geforderten Fähigkeit im Vergleich zu den bereitgestellten Fähigkeiten einer Ressource. [übersetzt aus BMWi Hrsg. (2020a)]
Capability- & Feasibility Checker-Framework	Das in Deliverable 2.7 „Spezifikation Generisches Architekturrahmenwerk für Machbarkeitstests“ beschriebene Framework bestehend aus Broker-Service, Checker-Services und einem Toolkit zur Erstellung von Checker-Services. [BaSys (2021)]
Capability- und Feasibility-Interaktion	Abfolge von Nachrichten zum fähigkeitsbasierten Engineering von Produktionsprozessen.
Checker-Framework	Siehe Capability- & Feasibility Checker-Framework. [BaSys (2021)]
Checker-Service	Service, der Szenarien checkt und Auskunft über Capabilities und Feasibilities gibt. [BaSys (2021)]
Feasibility (dt. Fertigkeit)	Umsetzbarkeit eines Szenarios mit einer bestimmten Ressource. [BaSys (2021)]
Feasibility Check(-ing)	Machbarkeitstest eines Szenarios, das über die grundsätzliche Fähigkeit einer Ressource eine Aufgabe auszuführen hinausgeht und auch die gegebenen Umstände mit in Betracht zieht. [BaSys (2021)] Ein formales Verfahren zur Beurteilung der Möglichkeit, die gewünschte Wirkung einer Fertigkeitsausführung in einem konkreten Kontext zu erzielen. [übersetzt aus BMWi Hrsg. (2020a)]

Kontext	Zusätzliche Informationen aus einer Beziehung oder einem Umfeld, die berücksichtigt werden können. [BMWi Hrsg. (2021)]
Ontologie	Eine formale, explizite Spezifizierung einer gemeinsamen Konzeptualisierung (d.h. ein abstraktes Modell eines Phänomens in der Welt durch die Identifizierung der relevanten Konzepte dieses Phänomens). [übersetzt aus BMWi Hrsg. (2020a)]
Prozess	Gesamtheit der Verfahren in einem System, mit deren Hilfe das Material, die Energie oder die Information umgewandelt, transportiert oder gespeichert wird. [BMWi Hrsg. (2021)]
Produkt	Die Zwischen- oder Endware, die als Ergebnis eines Prozessschrittes entsteht. [übersetzt aus BMWi Hrsg. (2020a)]
Ressource	Entsprechend der Definition im PPR-Modell [BWWi Hrsg. (2020)]. Eine Software- oder Hardware-Komponente, die in einem Prozess ein Produkt verarbeitet. Ein Asset, das in einem Prozess zur Durchführung der Verfahren verwendet wird. [übersetzt aus BMWi Hrsg. (2020a)]
Skill	Die anlagenabhängige Implementierung der Funktion einer Ressource, um eine bestimmte Wirkung in der physischen oder virtuellen Welt zu erzielen. [übersetzt aus BMWi Hrsg. (2020a)]
Szenario	Die Summe der übergebenen Informationen an einen Checker, die die PPR- sowie möglicherweise Kontextinformationen enthält. [BaSys (2021)]

1 Einleitung

Als zentrale Zielsetzung des Forschungsprojektes BaSys 4.2 wird die Weiterentwicklung der Open-Source Industrie 4.0 Middleware Eclipse BaSyx hinsichtlich des fähigkeitsbasierten kontinuierlichen Engineerings von Produktionsprozessen gesehen. [BaSys (2019)] In Abbildung 1 wird das kontinuierliche Engineering von Produktionsprozessen veranschaulicht. Es stellt die Fähigkeit einer Produktionsanlage dar, Produktionsprozesse jederzeit effizient auf unvorhergesehene Änderungen anzupassen. Änderungen können sich seitens der Produkte, der Prozesse oder Ressourcen ergeben. Beispiele für solche Änderungen sind der Auftrag für ein neues Produkt, der sich auch auf den Produktionsprozess zur Fertigung des Produktes bezieht, oder aber der Ausfall einer Ressource, wodurch die zuvor geplante Fertigung des Produktes nicht mehr möglich ist und einer Anpassung bedarf.

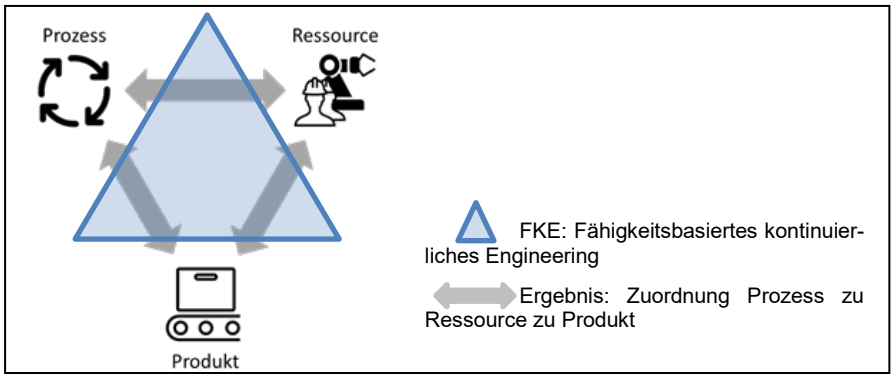

Abbildung 1: Fähigkeitsbasiertes kontinuierliches Engineering von Produktionsprozessen [BaSys (2019)]

Für eine wandelbare Produktion ist ein fähigkeitsbasiertes kontinuierliches Engineering von Produktionsprozessen unerlässlich, insbesondere wenn es um die effiziente Produktion der Losgröße 1 geht. Wie im Joint Whitepaper der Plattform Industrie 4.0, VDI GMA 7.20 und BaSys 4.2 in Kapitel 3.1 aufgezeigt, ermöglichen bestehende Engineering-Prozesse von Industrie 3.0 Systemen diese Flexibilität nicht. [BMWi Hrsg. (2020a)] Deswegen wurde in BaSys 4.0 das Konzept des fähigkeitsbasierten kontinuierlichen Engineerings entwickelt und im BaSys 4.2 Arbeitspaket 2 weiter fortgeführt.

Im Folgenden wird zunächst das Konzept des kontinuierlichen Engineerings von Produktionsprozessen mittels Fähigkeitsbeschreibungen vorgestellt. Im Anschluss werden die Interaktionen während des fähigkeitsbasierten kontinuierlichen Engineerings näher beleuchtet, die aus den Verhandlungen über die mögliche Herstellung von Produkten resultieren.

1.1 Fähigkeitsbasiertes kontinuierliches Engineering von Produktionsprozessen

Im BaSys 4.2-Antrag [BaSys (2019)] ist u.a. ausgeführt: „Grundlage für eine erfolgreiche Wandlung einer Produktionsanlage ist der intelligente Umgang mit durch Ressourcen zugesicherten und von Prozessen und Produkten angeforderten Fähigkeiten. Gegenstand des Arbeitspakets 2 ist die Modellierung von Fähigkeiten, sowie automatisierte Fähigkeitsabgleiche. Fähigkeiten werden durch komponentenübergreifende Fähigkeitsmodelle sowie ergänzend

durch fähigkeitsbezogene Teilmodelle der Verwaltungsschalen beschrieben. (…) Um den Abgleich von geforderten und angebotenen Fähigkeiten zu ermöglichen, wird ein Fähigkeitsabgleich und ein Machbarkeitstest realisiert. Der Fähigkeitsabgleich entscheidet anhand von logischen Modellen, ob eine geforderte Fähigkeit durch die angebotenen Fähigkeiten eines Geräts prinzipiell realisiert werden kann. Der Machbarkeitstest prüft zum Beispiel mit Simulationen unter Berücksichtigung der beteiligten Komponenten und Randbedingungen, ob eine Produktionsaufgabe durchgeführt werden kann.“

In Abbildung 2 ist der zeitliche Ablauf und der Zusammenhang zwischen dem oben geforderten Fähigkeitsabgleich (Capability Check) und dem Machbarkeitstest (Feasibility Check) dargestellt. Sind diese erfolgreich, kommt es zur Ausführung der Fähigkeit (Skill Execution).

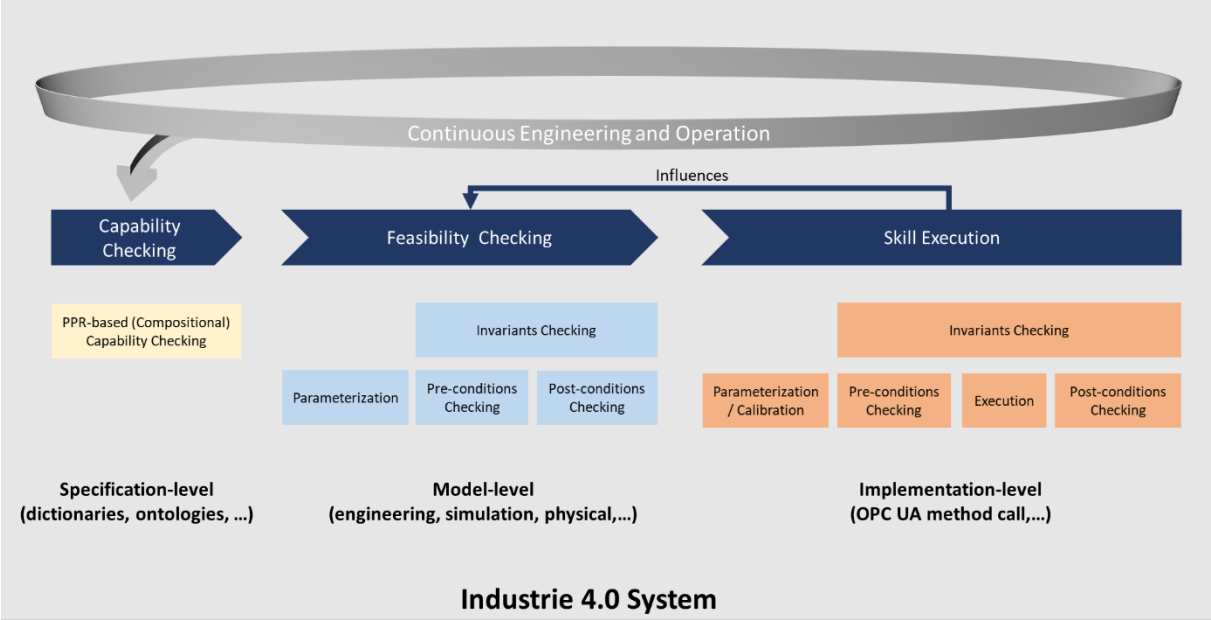

Abbildung 2: Einordnung des Fähigkeits- und Machbarkeitstest in ein I4.0-System

Wie in Abbildung 1 dargestellt, spielen folgenden drei Elemente eine wichtige Rolle für ein fähigkeitsbasiertes kontinuierliches Engineering und den Betrieb: Prozess, Produkt und Ressource, deren Abhängigkeiten im PPR-Modell modelliert werden. Hier sind sich die Ressourcen ihrer eigenen Fähigkeiten bewusst, bestimmte Effekte zu erzielen, ohne zu wissen, in welchen Prozessen und für welche Produkte sie verwendet werden. Jeder Prozess gibt die erforderlichen Fähigkeiten an. Die geforderten Eigenschaften von Prozessen und Produkten sowie die angebotenen Eigenschaften der Ressource bestimmen, ob eine Ressource die gewünschten Effekte in einem Prozess erzielen kann.

Abbildung 3 zeigt die Elemente des PPR-Modells jeweils als I4.0-Komponente bestehend aus der Verwaltungsschale (VWS) und ihrem Asset. Assets sind jeweils die Ressourcen, d.h. das Betriebsmittel (im Beispiel die Produktionszelle), das Produkt und der Prozess, der die Produktion des Produktes mit den dafür erforderlichen Fähigkeiten vorgibt. Assets können auch Akteure sein, wie z.B. Werker*in oder Monteur*in, die durch die Erfüllung ihrer Rollenbeschreibung Fähigkeiten besitzen.

Die geforderten Fähigkeiten des Prozesses (beschrieben in dessen VWS) müssen mit den angebotenen Fähigkeiten der Ressource abgeglichen werden. Dann kann der Plan im Prozess zum Produzieren des Produkts führen. Es soll auch möglich sein, Einzelfähigkeiten so zu kombinieren, dass die geforderten evtl. komplexeren Fähigkeiten angeboten werden können.

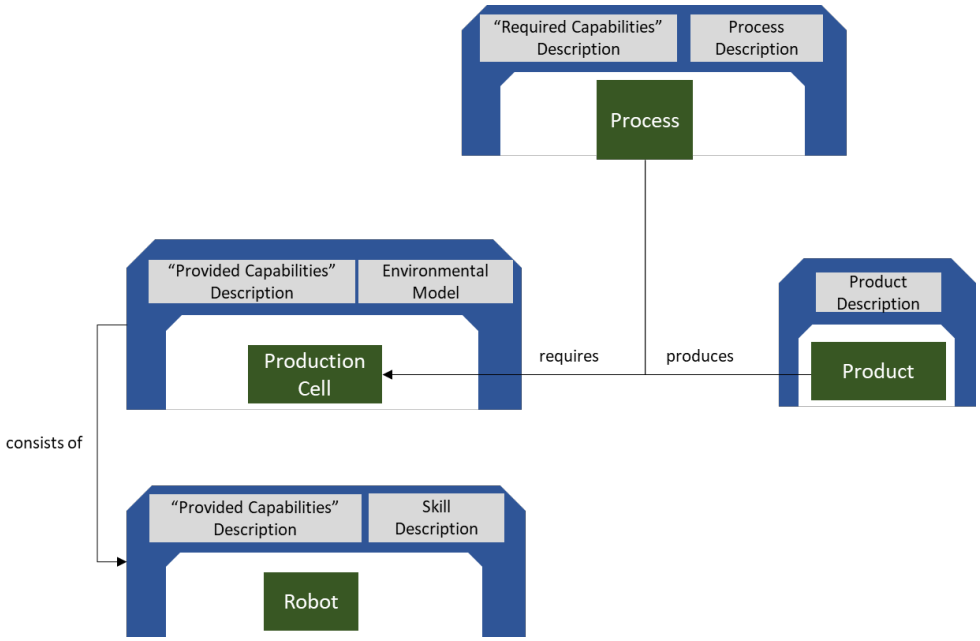

Abbildung 3: Betrachtung der Produkte, Prozesse und Ressourcen als I4.0Komponente (Asset + VWS) und deren Beziehungen untereinander

Für diesen Abgleich müssen die VWS interagieren. Dieses Interaktionsprinzip wird im Folgenden beschrieben.

1.2 Interaktionen während des kontinuierlichen Engineerings

Die Interaktionen während des fähigkeitsbasierten kontinuierlichen Engineerings resultieren aus Verhandlungen über die mögliche Herstellung von Produkten. Im Folgenden werden zwei Anwendungsfälle zur benötigten Verhandlungsfähigkeit der BaSys Middleware gezeigt, die im Rahmen von BaSys 4.2 betrachtet werden.

Tabelle 1: Use Case Herstellung eines Produktes innerhalb eines Unternehmens

D-2.11-1	**Verhandlung über die Herstellung eines Produktes innerhalb eines Unternehmens**
Beschreibung	Die Verhandlung, ob bzw. durch welche Ressourcen ein Produkt innerhalb eines Unternehmens hergestellt werden kann, kann aus folgenden Gründen notwendig werden: 1. Der/die Kund*in fragt die Herstellung eines neuen Produktes an.

	2. Eine Ressource fällt während eines Produktionsprozesses aus und muss ersetzt werden oder es soll eine Produktionsprozessoptimierung durchgeführt werden. 3. Der Produktionsprozess wird optimiert aufgrund neuer Produktionsanforderungen oder neuer Ressourcen. In beiden Fällen muss die Produktion des Produktes mittels des fähigkeitsbasierten, kontinuierlichen Engineerings neu geplant werden.
Vorbedingung	- Produkteigenschaften - Prozessbeschreibung zur Herstellung des Produktes - Fähigkeitsbeschreibung der vorhandenen Ressourcen - Optimierungs- und Auswahlkriterien
Eingabe	- Aus der Prozessbeschreibung abgeleitete geforderte Fähigkeiten - Produkt- und Prozesseigenschaften als Qualitätsparameter
Ausgabe	- Keine, eine oder mehrere Lösungen zur Umsetzung der Herstellung des Produktes, die auf ihre Machbarkeit geprüft wurden. - Zugesicherte Produkt- und Prozesseigenschaften als Qualitätsparameter - Zugesicherter Liefertermin

Tabelle 2: Use Case Production-as-a-Service-Plattform

D-2.11-2	**Verhandlung über die Herstellung eines Produktes mit externen Unternehmen oder innerhalb eines Unternehmens mit mehreren Produktionsstandorten auf einer Production-as-a-Service-Plattform**
Beschreibung	In einer Production-as-a-Service-Plattform (PaaS) werden Anfragen über die Herstellung von Produkten eingestellt und Angebote zu jenen Anfragen abgegeben. Eine PaaS-Plattform kann entweder innerhalb eines Unternehmens mit mehreren Produktionsstandorten eingesetzt werden, um eine effiziente Produktionsauslastung unternehmensübergreifend zu ermöglichen, oder kann durch einen PaaS-Plattform-Betreibenden als Marktplatz bzw. Bietplattform für voneinander unabhängige Unternehmen bereitgestellt werden. In beiden Fällen müssen Produktionsstandorte bzw. externe Unternehmen vor der Abgabe eines Angebots prüfen, ob sie die Ressourcen besitzen, das angefragte Produkt mit den geforderten Qualitätsmerkmalen zu den angegebenen Konditionen herstellen zu können. Diese Überprüfung kann mit Hilfe des fähigkeitsbasierten kontinuierlichen Engineerings durchgeführt werden. In diesem Anwendungsfall findet die Verhandlung über die Produktherstellung sowohl auf der PaaS-Plattform als auch innerhalb des fähigkeitsbasierten Engineerings statt.

Vorbedingung	- Produkteigenschaften - Prozessbeschreibung zur Herstellung des Produktes - Fähigkeitsbeschreibung der vorhandenen Ressourcen - Optimierungs- und Auswahlkriterien
Eingabe	- Aus der Prozessbeschreibung abgeleitete geforderte Fähigkeiten - Produkt- und Prozesseigenschaften als Qualitätsparameter - Preisgrenze - Geforderter Liefertermin
Ausgabe	- Keine, eine oder mehrere Angebote zur Herstellung des Produktes, die die Herstellung des Produktes garantieren. - Preisangebot - Zugesicherter Liefertermin

In beiden Verhandlungsszenarien werden geforderte Fähigkeiten aus der Prozessbeschreibung zur Herstellung eines Produktes und den dazugehörigen Qualitätsparametern mit zugesicherten Fähigkeiten, die durch vorhandene Ressourcen in einer bestimmten Qualität bereitgestellt werden, abgeglichen (Capability Check). Ergeben sich bei diesem Abgleich mögliche Lösungen zur Realisierung des Produktes, werden diese Lösungen auf ihre Machbarkeit in der realen Welt geprüft (Feasibility Check). Sowohl für den Capability- als auch für den Feasibility Check müssen Informationen ausgetauscht werden. Dieser Informationsaustausch wird als Capability- und Feasibility-Interaktion bezeichnet, zu der in Kapitel 2 eine Erläuterung folgt.

2 Konzept der Capability- und Feasibility-Interaktionen

Um die Interaktionen der Capability- und Feasibility Checks zu erläutern, bedarf es zunächst einer kurzen Vorstellung des Checker-Frameworks, das in dem Deliverable 2.7 eingeführt wird. Das Checker-Framework bildet die Grundlage für die Implementierung der Capability- und Feasibility Checks. Das Kapitel 2.1 befasst sich mit der Kurzvorstellung des Checker-Frameworks. In dem darauffolgenden Kapitel werden die Capability- und Feasibility-Interaktionen während des kontinuierlichen Engineerings an den oben aufgeführten Anwendungsfällen D-11-1 und D-11-2 aufgezeigt.

2.1 Checker-Framework aus dem Deliverable 2.7

Zu Beginn ist zu erwähnen, dass das Checker-Framework den Capability Check als einen Sonderfall des Feasibility Checks betrachtet. Der Capability Check überprüft dabei die Machbarkeit eines Produktionsszenarios rein symbolisch und kontextunabhängig. Ein Feasibility Check hingegen prüft die Machbarkeit eines Produktionsszenarios detaillierter bspw. mittels Simulationen innerhalb eines Umweltmodells. Aus diesem Grund wird in dem Checker-Framework nicht mehr weiter zwischen den zwei unterschiedlichen Checks unterschieden.

Basierend auf einer Micro-Service-Architektur (Abb. 4), bietet das Checker-Framework die zwei Services „Broker-Service“ und „Checker-Service“ an. Der Broker-Service ist für die Vermittlung passender Checker an anfragende Applikationen, wie z.B. eines Produktionsplanungssystems (D-11-1) oder einer Production-as-a-Service-Plattform (D-11-2), zuständig. Dazu müssen sich Checker an dem Broker-Service an- und abmelden. Der Checker-Service überprüft die Machbarkeit der Produktionsszenarien und erteilt Auskunft über Capabilities und Feasibilities der Ressourcen.

Die Abbildung 4 veranschaulicht die Micro-Service-Architektur des Checker-Frameworks. Komponenten stellen hierbei die Ressourcen einer Fabrik dar, die VWS besitzen. Wichtige Informationen der Komponenten werden als Teilmodelle in VWS gespeichert und verfügbar gemacht. Als Teilmodell werden bspw. die Fähigkeitsbeschreibungen, das Umweltmodell oder auch Services von Komponenten angeboten. Neben Produktionsplanungssystemen und Production-as-a-Service-Plattformen können Applikationen auch ERP-Systeme oder Softwareanwendungen der Komponenten sein, die Berechnung z.B. zur Ermittlung eines Greifpunktes durchführen. Es gibt die Möglichkeit, Checker-Services auf einer Komponente selbst über ein Teilmodell zur Verfügung zu stellen oder diese sowohl innerhalb als auch außerhalb der BaSyx-Middleware bereitzustellen. Checker-Services können miteinander und mit Applikationen entweder direkt oder aber auch über den Checker-Broker kommunizieren. Dies ist abhängig davon, ob die Checker (analog zu Applikationen) die ID des benötigten Checkers und dessen Endpunkt bereits kennen oder nicht. Ist die Checker-ID oder der Endpunkt nicht bekannt, erfolgt die Vermittlung des passenden Checkers über den Checker-Broker. [BaSys (2021)]

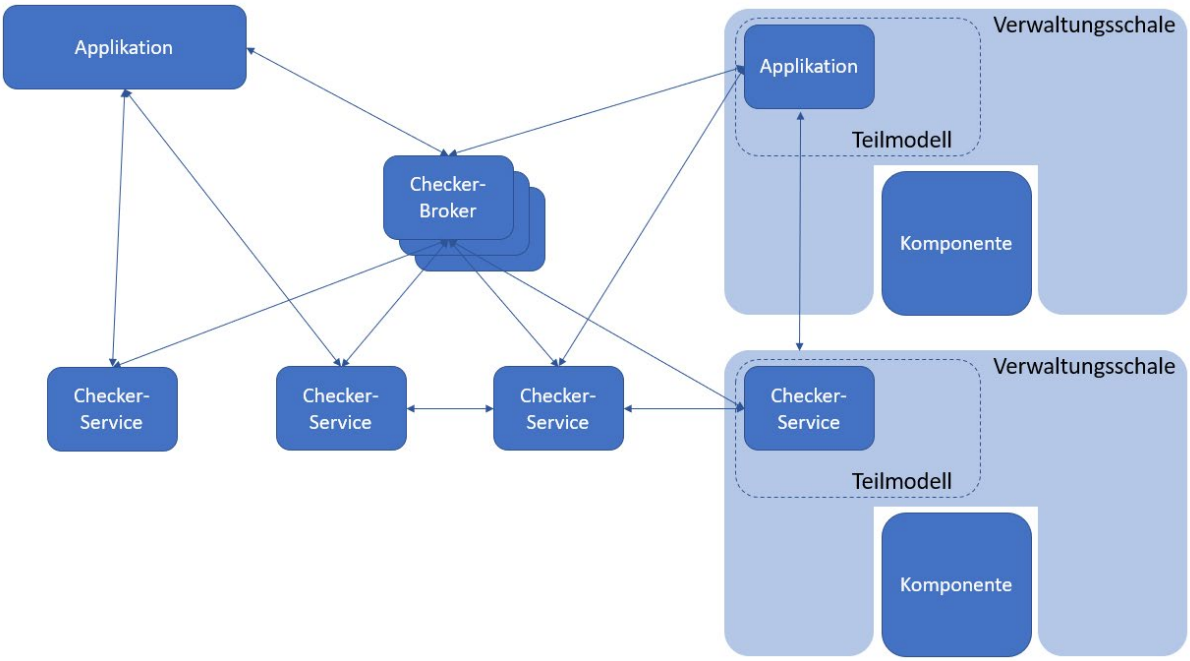

Abbildung 4: Architektur des Checker-Frameworks [BaSys (2021)]

2.2 Checker-Interaktionen

Die in Abbildung 4 dargestellten Pfeile zeigen die Interaktionen während der Checks. Diese können jeweils bidirektional zwischen einer Applikation und einem Checker oder zwischen zwei Checkern untereinander erfolgen.

Bei dem fähigkeitsbasierten kontinuierlichen Engineering der Anwendungsszenarien D-2.11-1 und D-2.11-2 beginnt eine Interaktion mit einer Anfrage bzgl. der Machbarkeit der Herstellung eines Produktes. Dabei werden die geforderten Fähigkeiten innerhalb eines Produktionsprozesses, der Technologie- und Ressourcen-agnostisch beschrieben ist, an einen Capability Checker übermittelt. Um die prinzipielle Machbarkeit zu prüfen, fragt der Capability Checker die vorhandenen Fähigkeiten der Fabrik-Ressourcen über deren VWS ab und gleicht diese mit den geforderten Fähigkeiten ab. Alternativ kann einem Checker fiktive Ressourcen inkl. ihrer vorhandenen Fähigkeiten direkt mit übermittelt werden. Ein positives Ergebnis des Capability Checks kann eine oder mehrere Möglichkeiten aufzeigen, wie die Produktionsschritte des Produktionsprozesses mit den vorhandenen Ressourcen ausgeführt werden können. Es werden eine oder mehrere Produktionsprozesse ausgegeben, die pro Produktionsschritt einen oder mehrere Ressourcentypen enthalten, durch die der Produktionsschritt ausgeführt werden kann.

Entsprechend des fähigkeitsbasierten kontinuierlichen Engineering-Prozesses folgt einem positiven Capability Check ein oder mehrere Feasibility Checks. Diese Feasibility Checks betrachten zusätzlich den Kontext der Ressourcen in der realen Welt und führen eine detaillliertere Machbarkeitsprüfung durch. Dazu werden entweder von dem Capability Checker oder durch eine andere Applikation die ressourcenabhängigen Produktionsprozesse an den Feasibility Checker übermittelt. Zusätzlich sind je nach Feasibility Checker Informationen, wie z.B.

die terminlichen und preislichen Konditionen, Betriebszustände, technische und geometrische Daten sowie Umwelt- und Simulationsmodelle der eingeplanten Ressourcen, relevant. Diese fragt der Feasibility Checker bei Applikationen und VWS der Ressourcen ab oder erhält sie direkt beim Aufruf. Das Ergebnis eines positiven Feasibility Checks sind ein oder mehrere ressourcenabhängige Produktionsprozesse, die je Produktionsschritt konkrete, parametrisierte Skills von Ressourceninstanzen enthalten. Applikationen und Checker können Meta-Information zu dem Erfolg eines Feasibility Checks abfragen. Diese Meta-Informationen werden bspw. als binäre Ausdrücke (z.B. Ja oder Nein) oder in Form einer Wahrscheinlichkeit bereitgestellt.

Im letzten Schritt des fähigkeitsbasierten kontinuierlichen Engineerings entscheiden Nutzer*innen über die Ausführung der identifizierten ressourcenabhängigen Produktionsprozesse. In beiden Anwendungsszenarien dieses Deliverables ist es möglich, dass ein Angebot bzgl. der Herstellung eines angefragten Produktes mit neuen Konditionen unterbreitet wird (z.B. neuer Liefertermin oder höherer Preis). Solche Informationen, die die Wirtschaftlichkeit oder auch die Vertragsbedingungen betreffen, sind für Nutzer*innen bei der Entscheidungsfindung zusätzlich relevant. Wird ein ressourcenabhängiger Produktionsprozess ausgeführt, ruft die Applikation zur Prozesssteuerung die jeweiligen parametrisierten Skills entweder über die VWS auf oder steuert sie direkt auf den Ressourcen an.

Zur besseren Verständlichkeit der technischen Umsetzung von Checker-Interaktionen werden in Kapitel 2.3 Akteure und ausgetauschte Informationsentitäten spezifiziert, die für beide Anwendungsfälle dieses Deliverables gelten.

2.3 Akteure und Informationsentitäten für Checker-Interaktionen

Für die zuvor beschriebenen Checker-Interaktionen werden Akteure und Informationsentitäten festgelegt, die unabhängig von technologischen Implementierungen sind. Die Zuordnung von Akteuren und Informationsentitäten zu konkreten Umsetzungsvarianten der Checker-Interaktionen erfolgt in Kapitel 3.

In diesem Unterkapitel werden als erstes die Akteure, die während einer Checker-Interaktion agieren, vorgestellt. Diese Akteure können sowohl Personen als auch (Software-)Systeme sein. Als nächstes werden die Meta-Informationsentitäten „Anfrage“ und „Angebot“ erläutert. Sie sind Gegenstand jeder Verhandlung über die Herstellung von Produkten und beinhalten Ein- und Ausgabewerte für die Checker-Interaktionen. Des Weiteren haben sie einen rechtlich bindenden Charakter. Zuletzt werden die Informationsentitäten präsentiert, die Teil der Meta-Informationsentitäten sein können und während einer Checker-Interaktion zwischen Akteuren ausgetauscht werden. Bei den Informationsentitäten wird unter Anderem zwischen Asset-Typen und Asset-Instanzen unterschieden. Ein Asset-Typ wäre bspw. der adaptive Formgreifer DHEF-20-A von Festo. Eine Asset-Instanz wäre hiervon ein physisch existierender adaptiver Formgreifer des Typs DHEF-20-A mit einem eindeutigen Identifikationsmerkmal (AssetID). Die Asset-Instanz verfügt über alle Eigenschaften des Asset-Typs, jedoch könnten diese je nach Kontext oder Ausführung beschränkt sein. So könnte bspw. die max. Geschwindigkeit von 120 mm/s des physischen Formgreifers gedrosselt sein, da die Sicherheitsbestimmungen der Fabrikanlage, in der der Formgreifer steht, eine max. Geschwindigkeit von nur 80 mm/s erlauben. Oder es wäre denkbar, dass Festo den Formgreifer in einer günstigeren, gedrosselten Variante verkauft, wobei Kunden die Möglichkeit haben, durch einen Mehrbetrag die schnellere Geschwindigkeit als zusätzliches Feature freischalten zu lassen. Ähnlich verhält es sich auch mit den Capabilities von Asset-Typen. Asset-Instanzen besitzen prinzipiell alle Capabilities,

die ihr Asset-Typ hat. Allerdings könnten einige je nach Kontext oder Ausführungsart nur teilweise oder überhaupt nicht verfügbar sein.

Akteure

- Prozessablaufplanung (Production Process Planning)
    - Die Prozessablaufplanung übernimmt die Planung des Produktionsprozesses. Dabei werden einzelne Arbeitsschritte, die zur Herstellung eines Produktes notwendig sind, in einer logischen und zeitlichen Reihenfolge zu einem Prozessablauf zusammengestellt.
    - Ergebnisse der Prozessablaufplanung sind die folgende Informationsentitäten:
        - Arbeitsplan/Rezept
        - Produktionsprozessablaufplan
- Capability Check (Capability Check)
    - Bei dem Capability Check findet eine Zuordnung von potentiellen Asset-Typen zu Arbeitsschritten auf der Basis der Capabilities statt.
    - Für den Capability Check werden geforderte Capabilities und Qualitätsparameter aus dem Arbeitsplan/Rezept benötigt.
- Feasibility Check (Feasibility Check)
    - Bei einem FeasibilityCheck werden Asset-Instanzen zu einem Arbeitsschritt mit einer konkreten Konfiguration und mit bestimmten Zeitangaben zugeordnet.
    - Zu den Ergebnissen zählen weiter die zugesicherten Konditionen für die Herstellung des Produktes.
    - Bei den Konfigurationen werden die Skills der Asset-Instanzen parametrisiert.
- Ressourcensteuerung (Resource Control)
    - Die Ressourcensteuerung übernimmt die technische Steuerung der Asset-Instanzen zur Laufzeit, bspw. Starten, Stoppen, Pausieren von Asset-Instanzen.
    - Zur Ressourcensteuerung wird der Produktionsprozessablaufplan benötigt.

Meta-Informationsentitäten

- Anfrage (Inquiry)
    - Enthält folgende Informationsentitäten:
        - Arbeitsplan/Rezept
    - Diese Informationsentität stellen für die Checker-Interaktionen Eingabewerte dar.
- Angebot (Offer)
    - Enthält folgende Informationsentitäten:
        - (Anonymisierter) Produktionsprozessablaufplan

- Diese Informationsentitäten stellen für die Checker-Interaktionen Ausgabewerte dar.
- Ein Angebot kann für den Verhandlungspartner eine neue Anfrage darstellen.

Informationsentitäten

- Arbeitsplan/Rezept (Work Schedule/Recipe)
  - Ein Arbeitsplan/Rezept ist eine geordnete Liste der Arbeitsschritte für die Produktion des Produktes.
  - Der Arbeitsplan/das Rezept sind Technologie- und Ressourcen-agnostisch beschrieben, d.h. unabhängig von der konkreten technischen, organisatorischen und rechtlichen Umsetzung. Der geforderte Effekt jedes Arbeitsschrittes bzw. die darin beschriebene Aktivität wird durch eine Capability dargestellt, die ggf. in einer bestimmten Qualität erfüllt werden muss.
- Capability-Modell (Capability Model)
  - Das Capability-Modell enthält formale, semantische Beschreibungen von Capabilities, die ein oder mehrere Asset-Typen anbieten.
  - Capabilities haben wesentliche Ein- und Ausgabeparameter. Zusätzlich besitzen sie die Information, durch welche Skills von Asset-Typen sie umgesetzt werden.
- Feasibility-Modell (Feasibility Model)
  - Das Feasibility-Modell schränkt das Capability-Modell basierend auf dem Umweltmodell (technologische, organisatorische, ökonomische Rahmenbedingungen) der Asset-Instanz ein.
  - Das Feasibility-Modell enthält Beschreibungen vom Umweltmodell, den Skills der Asset-Instanzen sowie andere Informationen, die zur Ausführung der geforderten Funktion erforderlich sind (z.B. mathematisch formulierte physikalische Zusammenhänge), die ein oder mehrere Asset-Instanzen anbieten.
- Asset-Typ-Daten (Asset Type Data)
  - Asset-Typ-Daten enthalten die Daten eines Asset-Typs, wie sie in Katalogen stehen bzw. bei Auslieferung der Produkte mitgegebenen werden.
  - Beispiele sind Informationen in Form von Handbüchern, Zertifikaten, Konstruktionsunterlagen etc.
  - Diese Informationen beziehen sich auf den Typ des Assets nicht auf die konkrete Instanz, mit der sie mitgeliefert werden.
- Asset-Instanz-Daten (Asset Instance Data)
  - Asset-Instanz-Daten sind eine dynamische Beobachtung der realen und/oder simulierten Asset-Instanz.
  - Sie beinhalten Informationen der Asset-Instanz zur Laufzeit und geben Auskunft bspw. über den aktuellen Betriebszustand, die Auslastung, den Standort oder den Wartungszustand.

- Ressourcenangebot (Offered Resource(s))
    - Das Ressourcenangebot beinhaltet alle vorhandenen Ressourcen, die potentiell in der Lage sind, Arbeitsschritte (Tasks) eines Arbeitsplanes/Rezeptes durchzuführen.
- Produktionsprozessablaufplan (Production Process Schedule)
    - In dem Produktionsprozessablaufplan findet eine Zuordnung sowohl von Capabilities als auch den daraus abgeleiteten Skills der Asset-Instanzen zu den Arbeitsschritten mit einer bestimmten Zeitangabe statt.
    - Der Produktionsprozessablaufplan bzw. einzelne Schritte des Produktionsablaufplans sind das Ergebnis des Feasibility Checks.
- Konfigurationsdaten (Configuration Data)
    - Die Konfigurationsdaten stellen eine bestimmte Konfiguration von Ressourcen bereits. Dazu zählt unter anderem die Parametrisierung von Ressourcen-Services.

3 Umsetzung

In Deliverable 2.7 wurde die Architektur sowie die notwendigen Arten von Applikationen definiert (Checker, Checker-Broker und Applikation). Zusätzlich wurde die Kommunikationsvermittlung zwischen Checker und Applikation beschrieben. Der Fokus lag hierbei auf der Bereitstellung von Checkern über einen Checker-Broker. Wie die Kommunikation zwischen einer Applikation und einem Checker genau funktioniert, wurde nicht erläutert. Festgelegt wurde nur, dass eine Applikation ein zu checkendes Szenario an den Checker übergibt und das Checker-Ergebnis erhält. In diesem Kapitel soll näher auf die Kommunikation zwischen der Applikation und dem Checker eingegangen werden.

Für die Kommunikation mit einem konkreten Checker können verschiedene Arten genutzt werden [DIN (2018)]. In diesem Beitrag werden eine API-basierte und Nachrichten-basierte Kommunikation vorgestellt. Die Funktionalität der Checker unterscheidet sich hierbei nicht. Lediglich wie der Aufruf dieser Funktionen durchgeführt wird, unterscheidet sich. Nachfolgend werden die beiden Arten beschrieben.

Prinzipiell gibt es mehrere verschiedene Möglichkeiten die Interaktion zwischen den Akteuren und Informationsentitäten zu gestalten. Hier werden zwei im Kontext der Verwaltungsschale eingesetzte Optionen vorgeschlagen.

3.1 Generelles Konzept

Die API-basierte und die Nachrichten-basierte Kommunikation unterscheiden sich in der Art und Weise der zu übermittelten Informationen. Während bei einer API-basierten Kommunikation die Informationen in Form von einzelnen Funktionsaufrufen ausgetauscht werden und somit sowohl die Funktionsdefinitionen als auch der Ablauf vom Client geregelt werden müssen, geschieht dies bei der Nachrichten-basierten Kommunikation in Form eines Austauschs einer konkreten Nachricht, in der alle notwendigen Informationen enthalten sind. Die konkrete Umsetzung wird in den Kapiteln 3.2 und 3.3 beschrieben.

Zunächst sollen die allgemeinen, auszutauschenden Informationen näher beschrieben werden. Das Checker-Framework soll einheitliche Schnittstellen, sowohl für Capability- als auch für Feasibility Checker, zur Verfügung stellen. Jeder Checker soll je einen Endpunkt mit einer info-Methode, die eine Selbstauskunft der Services bereitstellt, und einer check-Methode, von der der eigentliche Check durchgeführt wird, zur Verfügung stellen. Darüber hinaus kann der Checker eine schema-Methode bereitstellen, die angibt, in welchem Format Szenarienbeschreibungen von ihm verarbeitet werden können. Die ausgetauschten Informationen sind in Abbildung 5 dargestellt. In den beiden folgenden Abschnitten wird dieses generische Modell auf die beiden in Abschnitt 1.2 definierten Use Cases konkretisiert.

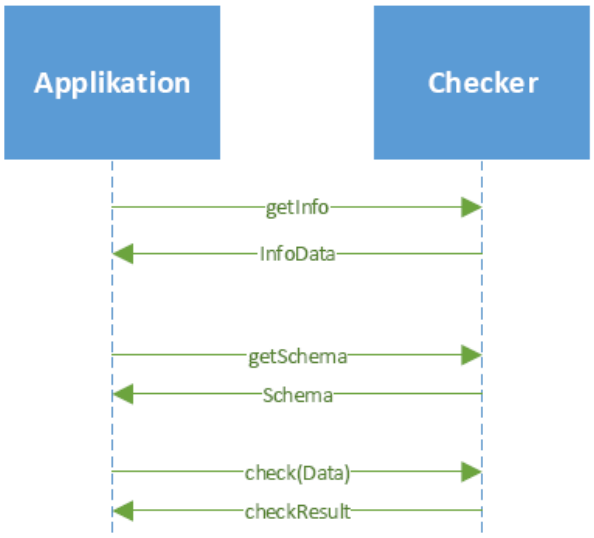

Abbildung 5: Generischer Informationsaustausch zwischen Applikation und Checker

3.1.1 Use Case 1: Verhandlung über die Herstellung eines Produktes innerhalb eines Produktionsstandortes

In diesem Use Case wird die Verhandlung für die Herstellung eines Produktes innerhalb eines Unternehmens betrachtet. Das in Abbildung 5 gezeigte Interaktionsdiagramm kann weiter verfeinert werden. Eine mögliche Verfeinerung ist in Abbildung 6 und Abbildung 7 dargestellt (Checker erhalten die VWS-IDs (siehe Abbildung 5). Die Applikation besteht in diesem Use Case aus den in Kapitel 2.3 definierten Akteuren, Meta-Informationsentitäten und Informationsentitäten. Ob die Akteure und (Meta-)Informationsentitäten jeweils in einer einzelnen Applikation oder alle in einer Applikation enthalten sind, wird nicht weiter festgelegt. Für die Interaktion reicht es zu wissen, welche Akteure und (Meta-)Informationsentitäten es gibt und wer mit wem welche Informationen austauscht. Nachfolgend ist der Ablauf dargestellt:

Eine Variation und Verfeinerung des Ablaufs kann nach Kenntnis der entsprechenden Details der entworfenen Checker und VWS erfolgen.

Es wird angenommen, dass

- das Produktionsplanungssystem den Gesamtablauf organisiert,
- der Arbeitsplan, die Assets sowie das Produktionsplanungssystem über die VWS-Schnittstelle entsprechend VDI 2193 verfügen und
- die Nachrichten (hier nur mit Namen benannt) der Definition nach VDI/VDE 2193 und entsprechend der Syntax nach VWS im Detail [BWWi Hrsg. (2020b)] z.B. im JSON-Format folgen.

Der prinzipielle Ablauf ist wie folgt:

1. Das Produktionsplanungssystem hat einen Auftrag bekommen (in Abbildung 6 nicht sichtbar) und sucht aus der dafür gültigen Informationsentität „Arbeitsplan/Rezept“ die nächste geforderte Capability heraus.
2. Mit der gefundenen Capability beauftragt das Produktionsplanungssystem den Akteur „Capability Check“ mit einer Vorauswahl der in Frage kommenden Assets.
3. Der Akteur „Capability Check“ macht eine Abfrage bei den VWS der Asset Typen, nach der gesuchten Capability. Der Call for Capability enthält die Meta-Informationsentität „Anfrage“. Diese VWS enthalten in diesem Use Case die von den Asset-Typen angebotenen Capabilities (siehe Kapitel 2.3).
4. Die VWS der Asset Typen übermitteln alle angebotenen Capabilities, falls vorhanden. Offer Capability enthält die Informationsentität „Ressourcenangebot“. Als eine besondere Möglichkeit in diesem Use Case wird gezeigt, dass eine Asset-VWS auch direkt bei einer anderen Asset-VWS um Ergänzung der eigenen Capability anfragen kann. Ist dies erfolgreich, kann diese auch die zusammengesetzte Capability anbieten.
5. In Zusammenarbeit mit dem Capability-Modell werden die Assets ausgewählt, deren VWS die benötigte Capabilities (allein oder im Verbund) anbieten können. (Capability Check und Capability Reasoning) Das Ergebnis wird dem Produktionsplanungssystem übersendet. (Asset/AAS Selected)
6. Die so vorausgewählten Assets sind detailliert zu prüfen, ob sie die Umsetzung zum gegebenen Zeitpunkt ausführen können und ob die Rahmenbedingungen stimmen. Dafür wird der Feasibility Check aktiviert. (Call for Skill)
7. Der Akteur „Feasibility Check“ fragt die benötigten Details aus den VWS der vorausgewählten Asset-Instanzen an. (Call for Skill(Anfrage) und Offer Skill(Ressourcenangebot) Wurde ein Asset beim Capability Check ausgewählt, das bei der Antwort ein weiteres Asset einbezogen hatte (siehe Schritt 4), so ist die Skill-Abfrage von der entsprechenden VWS selbstständig zu ergänzen. Dies ist mit 7a in Abbildung 7 gekennzeichnet.
8. Auf der Basis der Informationen des Produktionssystems und der Asset-VWS wird der Feasibility Check durchgeführt.
9. Rahmenbedingungen, die außerhalb des Assets liegen aber auf dieses einwirken wie z. B. das Umweltmodell eines Roboters in einer Fertigungszelle, die im Feasibility-Modell enthalten sind, müssen bei einer entsprechenden Softwarekomponente abgefragt werden. (Retrieve Information, Send Information)
10. Nach erfolgtem Feasibility Check wird das Ergebnis dem Produktionsplanungssystem zugesendet. (Feasibility Check Result)
11. Bei Bedarf können erzeugte Konfigurationsdaten an die VWS von ausgewählten Assets gesendet werden (z. B. ein Roboter soll nicht den schnellsten Weg fahren, sondern den mit dem geringsten Energieverbrauch). (Send Configuration)
12. Das Produktionsplanungssystem aktualisiert die Informationsentität „Produktionsprozessablaufplan“. (Update Production Process)

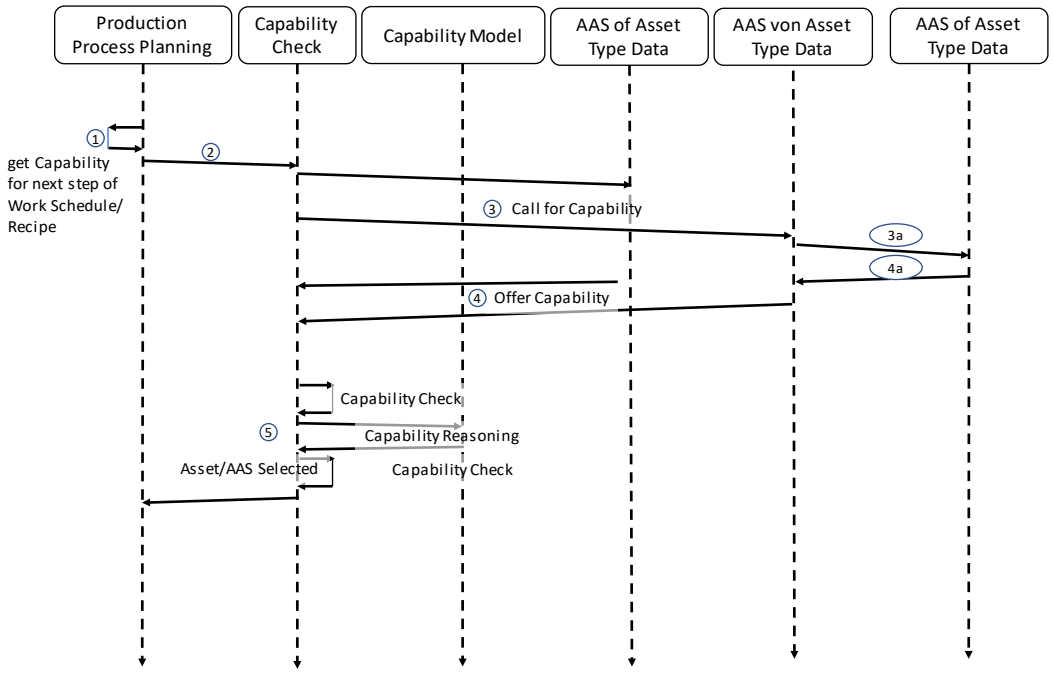

Abbildung 6: Interaktionen zum Ablauf des Capability Checks innerhalb einer Produktionslinie (produktionsstandortintern)

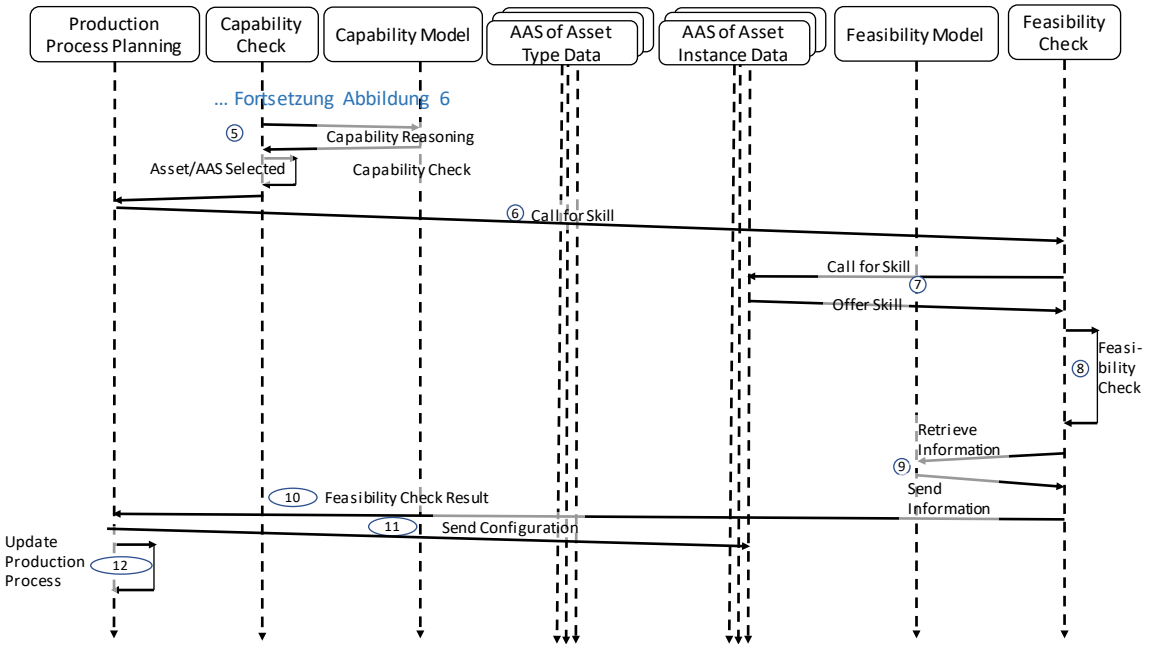

Abbildung 7: Interaktionen zum Ablauf des Feasibility Checks innerhalb einer Produktionslinie (produktionsstandortintern)

3.1.2 Use Case Production-as-a-Service-Plattform über Firmengrenzen und einzelne Produktionsstandorte hinaus

In diesem Use Case geht es um die Verhandlung für die Herstellung eines Produktes, wobei auch Leistungen von externen Unternehmen einbezogen werden können. Das in Abbildung 5 gezeigte Interaktionsdiagramm kann weiter verfeinert werden. Eine mögliche Verfeinerung ist in Abbildung 8 und Abbildung 9 dargestellt.

Die Applikation besteht in diesem Use Case aus den gleichen Akteuren und Informationsentitäten wie beim Use Case aus Abschnitt 3.1.1. Besonders ist hier, dass auch Assets mit ihren VWS einbezogen werden, die sich in anderen Unternehmen oder anderen Produktionsstandorten befinden. Die einzelnen Aktivitäten sind identisch zu denen in Kapitel 3.1.1. Deshalb wird hier auf eine erneute Auflistung verzichtet.

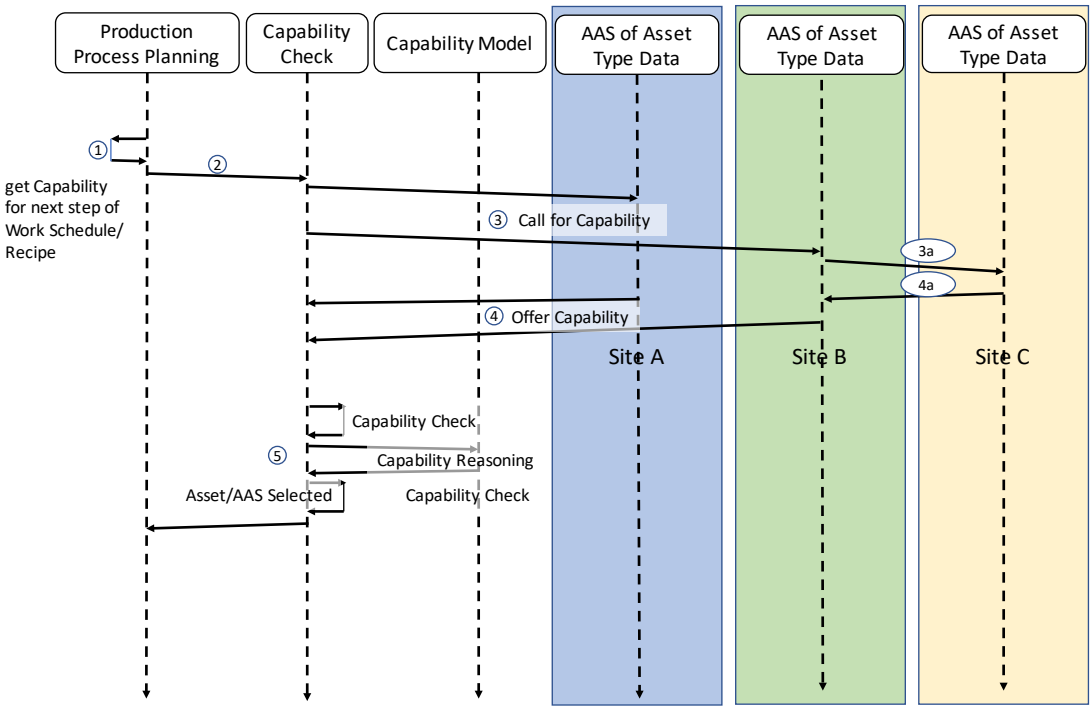

Abbildung 8: Interaktionen zum Ablauf des Capability Checks über Firmengrenzen und einzelne Produktionsstandorte hinaus

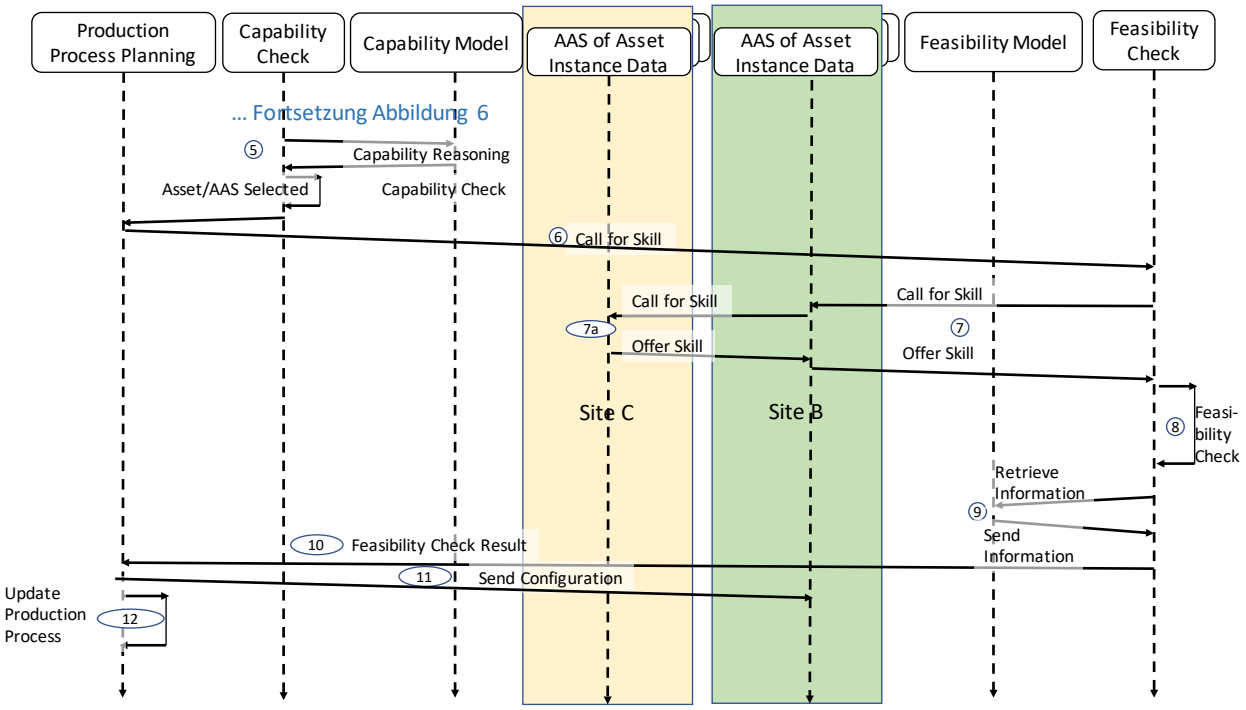

Abbildung 9: Interaktionen zum Ablauf des Feasibility Checks über Firmengrenzen und einzelne Produktionsstandorte hinaus

3.2 API-basierte Umsetzung

Bei der API-basierten Kommunikation wird das Client-Server-Pattern verwendet. Hierbei stellt ein Server eine API zur Interaktion zur Verfügung. Eine API ist eine Technologie-spezifische Beschreibung der Kommunikationsschnittstelle. Sie besteht wiederum aus Operationen, die einzelne Funktionalitäten des Servers zur Verfügung stellen. Zur Laufzeit können diese dann aufgerufen werden. Nachfolgend werden die einzelnen Funktionen beispielhaft in Form einer API-Beschreibung festgelegt:

Info-Methode

Jeder Checker stellt über die info-Methode Informationen über sich bereit, die Auskunft darüber geben, wie der Checker zu verwenden ist und in welchem Zustand er sich befindet. Dieses beinhaltet eine Freitextbeschreibung, sowie Eigenschaften des Checkers wie Echtzeitfähigkeit, die als Capabilities modelliert sind. Auch Informationen darüber, ob und welche Art von Schema der Checker verwendet, kann über Capabilities bereitgestellt werden. Es ist hierbei angedacht, dass die Menge an bereitgestellten Informationen bei Bedarf in der Zukunft noch iterativ erweitert wird.

Eingabe

Ein GET-Endpunkt.

Ausgabe

Beispiel:

{
„@id“ : „http://basys42.de/rdf/Checker/7890057a-35f9-4b86-aeb4-89a9781a540d“,
“@type”: „CapabilityChecker“,
„description“: “Checks capabilities of resources for their applicability to the given process.”,
„stateless“: “true”,
“version”: “1.0.0”,
“configuration”: “StandardConfiguration”,
“state” : “healthy”,
“started” : “Up since 2020-06-10 08:01:23”,
“capabilities”: [„AcceptsJson“, „Real-time“, „ChecksPayload“]
}

Schema-Methode

Jeder Checker hat die Möglichkeit ein Schema bereitzustellen, das das zulässige Format für Eingaben zur check-Methode definiert. Um das Framework nicht unnötig auf eine Technologie zu beschränken, sind hier beliebige Arten, ein Schema textuell zu beschreiben, zulässig. Empfohlen werden SHACL-Shapes, aber auch ein JSON-Schema wäre ein zulässiges Format. Ob und welche Schemas vom Checker verwendet werden, kann über die Capabilities der info-Methode erfragt werden.

Eingabe

Ein GET-Endpunkt.

Ausgabe

Beispiel:

```
basys:ProcessShape
        a sh:NodeShape ;
        sh:targetClass basys:Process ;
        sh:property [
                sh:path basys:products ;
                sh:class basys:Product ;
                sh:minCount 1 ;
        ] ;
        sh:property [
                sh:path basys:resources ;
                sh:class basys:Resource ;
```

```
                sh:minCount 1 ;
        ] ;
        sh:property [
                sh:path basys:steps ;
                sh:class basys:Step ;
                sh:minCount 1 ;
        ] ;
basys:ResourceShape
        a sh:NodeShape ;
        sh:targetClass basys:Resource ;
        sh:property [
                sh:path basys:payload ;
                sh:datatype xsd:integer ;
                sh:minCount 1 ;
                sh:maxCount 1 ;
        ] ;
basys:ProductShape
        a sh:NodeShape ;
        sh:targetClass basys:Product ;
        sh:property [
                sh:path basys:weight ;
                sh:datatype xsd:integer ;
                sh:minCount 1 ;
                sh:maxCount 1 ;
        ] ;
basys:StepShape
        a sh:NodeShape ;
        sh:targetClass basys:Step ;
        sh:property [
                sh:path basys:resource ;
                sh:class basys:Resource ;
                sh:minCount 1 ;
                sh:maxCount 1 ;
        ] ;
```

```
sh:property [
        sh:path basys:product ;
        sh:class basys:Product ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
] ;
```

Check-Methode

Es wird jedem Checker selber überlassen ein Format zu definieren und über den schema-Endpunkt bereitzustellen, welche er als Eingabe akzeptiert. Empfohlen wird jedoch eine Prozess-Beschreibung entsprechend dem PPR-Modell inklusive der VWS-Beschreibungen der im Prozess verwendeten Ressourcen. Da die BaSys-Prozessbeschreibung zum jetzigen Zeitpunkt noch nicht abgeschlossen ist, ist im Folgenden nur eine einfache, beispielhafte Prozessbeschreibung gegeben, die entsprechend angepasst wird, wenn die Modellierung final ist. Zusätzlich erhält der Checker optional Kontextinformationen. Diese sind nicht notwendigerweise als VWS modelliert, da Umweltfaktoren eventuell nicht über eine VWS verfügen, diese aber dennoch relevant sind.

Die Rückgabe enthält einen Konfidenzwert. Scheitert der Check und ist der Checker hierzu in der Lage, wird eine Begründung, woran der Check gescheitert ist, zurückgegeben.

Eingabe

Beispiel:

```
{
        „@context“ : „http://basys42.de/rdf/“,
        „@id“ : „http://basys42.de/rdf/Process/0567057a-35f9-4b86-aeb4-89a9781a540d“,
        „@type“ : „http://basys42.de/rdf/Process“,
        „products“ : [{
                „@id“ : „http://basys42.de/rdf/Product/4b67057a-35f9-4b86-aeb4“,
                „@type“ : „http://basys42.de/rdf/Product“,
                „weight“ : 56
        }],
        „resources“ : [{
                “@id”: “http://basys42.de/rdf/Resource/3247057a-35f9-4b86-aeb4“,
                „@type“ : „http://basys42.de/rdf/Resource“,
                “payload”: 100
        }],
        “steps”: [{
```

```
                "@id" : "http://basys42.de/rdf/Skill/LiftProduct",
                „@type“ : „http://basys42.de/rdf/Step“,
                "name" : "Lift product",
                "resource": {
                        "@id": "http://basys42.de/rdf/Resource/3247057a-35f9-4b86-aeb4“,
                        „@type“ : „http://basys42.de/rdf/Resource“
                        }
                "product": {
                         „http://basys42.de/rdf/Product/4b67057a-35f9-4b86-aeb4“
                        „@type“ : „http://basys42.de/rdf/Product“
                        }
        }]
}
```


Ausgabe

Beispiel erfolgreicher Check:


```
{
        „@context“ : „http://basys42.de/rdf/“,
        „@id“ : „http://basys42.de/rdf/CheckerResult/0b67057a-35f9-4b86-aeb4-
        89a9781a540d“,
        „@type“ : „http://basys42.de/rdf/CheckerResult“,
        „success“ : true,
        „confidence“ : 0.79
}
```


Beispiel fehlgeschlagener Check:


```
{
        „@context“ : „http://basys42.de/rdf/“,
        „@id“ : „http://basys42.de/rdf/CheckerResult/1c67057a-35f9-4b86-aeb4-
        89a9781a541c“,
        „@type“ : „http://basys42.de/rdf/CheckerResult“,
        „success“ : false,
        „confidence“ : 1.0,
        „explanation“ : [
                „Simulation software not reachable.“
        ]
```

}

3.3 Nachrichten-basierte Umsetzung

Die Use Cases aus Kapitel 3.1 werden hier mit den Nachrichten nach VDI 2193-1 dargestellt. Im Anhang Kapitel 5 sind die Grundlagen dieses Konzeptes kurz zusammengestellt und auch das hier zum Teil verwendete Ausschreibungsverfahren erläutert. Der Frame der Nachricht beinhaltet jeweils die Angaben, die zur Nachrichtenorganisation gehören, und das interactionElements-Feld entsprechend AASiD V3.0 RC01 (siehe 5.1). Der Frame für die Capability-Ausschreibung beinhaltet, neben der Quell- und Zieladressierung und den Daten zur Nachrichtenidentifizierung, auch den Protokolltyp (für die Capability- und Skill-Ausschreibung „Bidding“ nach VDI 2193-2), die dazugehörigen Akteure (Service Requester und Service Provider) und den Nachrichtentyp (hier z. B. CallForProposal und OfferProposal). Das interactionElements-Feld enthält die jeweiligen Teilmodelle. Die Abfrage zusätzlicher Informationen aus dem Feasibility-Modell und das Aktualisieren der Konfiguration wird nach dem klassischen Client/Server Protokoll durchgeführt. Dieses ist zurzeit noch nicht in VDI 2193 standardisiert, wird hier aber sinngemäß verwendet.

Um die Nachrichten besser einzuordnen, wird das prinzipielle Vorgehen, so wie in den Abbildungen Abbildung 6 bis Abbildung 9 beschrieben, hier noch einmal eigenständig in Abbildung 10 und Abbildung 11 dargestellt. Darin wird angenommen, dass die Akteure Asset-Type-Daten und Asset-Instanz-Daten den jeweiligen VWS zugeordnet werden, die das semantische Protokoll und die Nachrichten nach VDI 2193 am Interface anbieten.

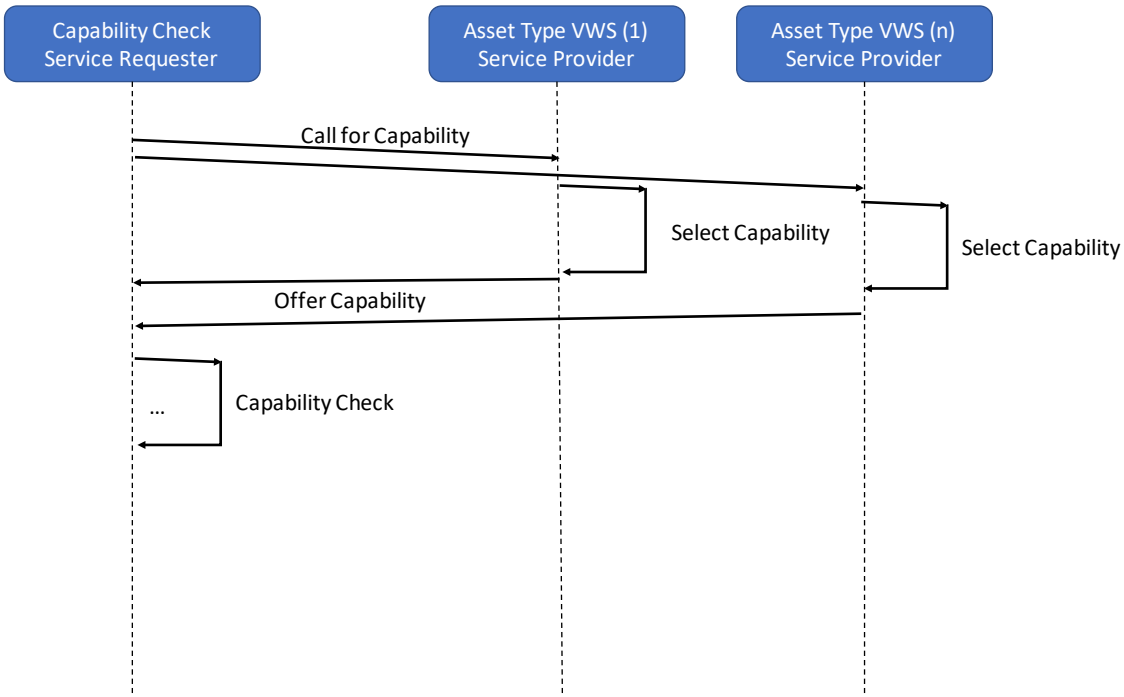

Abbildung 10: Capability Anfrage bei der VWS

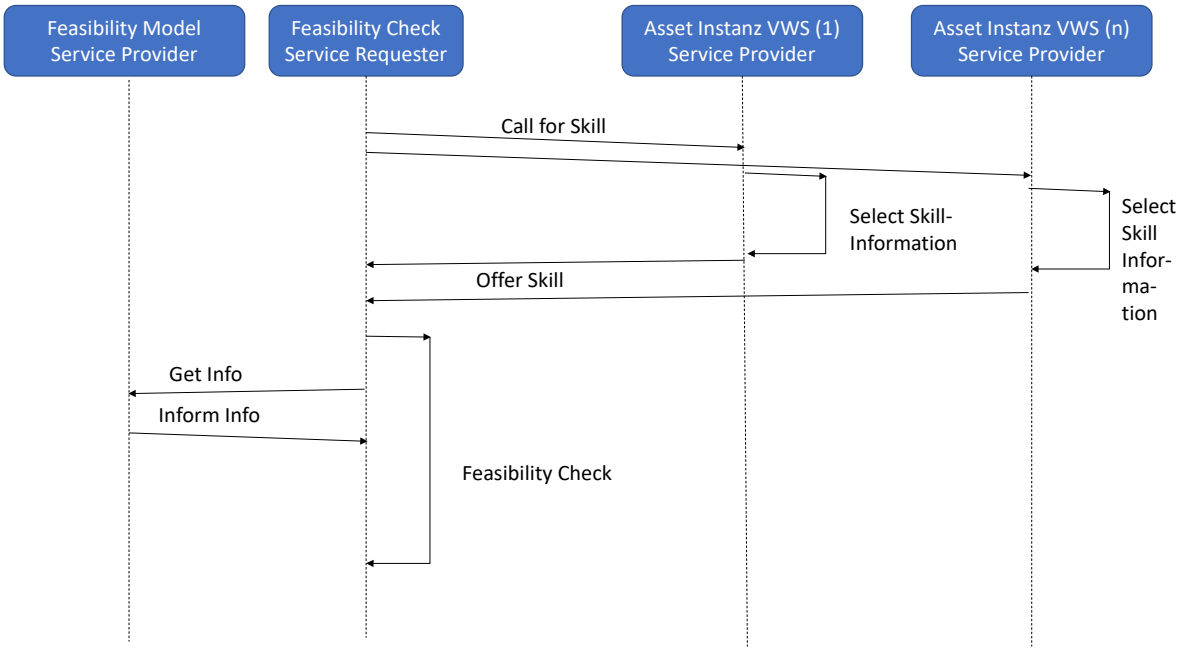

Abbildung 11: Feasibility Anfrage bei der VWS

Entsprechend der Darstellung in Abbildung 10 und Abbildung 11 werden nun beispielhaft die Nachrichten in ihrer syntaktischen JSON-Form vorgestellt.

Charakteristika der Nachrichtenbasierten Peer-to-Peer Interaktionsmuster

Vorab noch einige ergänzende Anmerkungen zur Struktur der Nachrichten. Grundprinzip der verwendeten Interaktionsabläufe ist das Peer-to-Peer Interaktionsmuster, das vollständig Kommunikationstechnologie-agnostisch ist. Es kann sowohl z.B. mit MQTT, HTTP/Rest und OPC UA durchgeführt werden. Es wird also eine Anwendung-zu-Anwendung-Interaktion beschrieben, die in Dialogen abläuft. Die Interaktion ist stateful. Dadurch entsteht der folgende Overhead:

- Definition des semantischen Protokolls
  - Hier wird das Interaktionsmuster beschrieben, das in diesem Beispiel „Bidding“ und „ApplicationInformationUsage“ ist. Es werden jeweils die Rollen der Sender und Empfänger beschrieben, da es sich in jedem Fall um ein Peer-to-Peer Interaktionsmuster handelt.
- Die InteractionElements enthalten den Submodel-Context des abzufragenden Elements
  - Es wird in der Nachricht 100% kompatibel die JSON-Serialisierung der VWS für die Submodels mit ihren SubmodelElements verwendet.
  - Hier könnte man evtl. auch reduziertere Lösungen finden.

Mit diesem Herangehen können heterarchische Systeme als Alternative zu hierarchischen Systemarchitekturen aufgebaut werden. Die heterarchische Systemarchitektur entspricht

mehr den Intensionen von I4.0-Systemen, die eine hohe Flexibilität aufweisen und autonome Komponenten enthalten. Dadurch entstehen folgende Charakteristika:

- Eine Festlegung auf Kommunikationsprotokolle für die Werkzeuge des Capability und Feasibility Checks ist nicht erforderlich.
- Die Serialisierung der VWS [BWWi Hrsg. (2020b)] kann unverändert verwendet werden.
- Peer-to-Peer-Interaktionsmuster ermöglichen eine lose Kopplung der Komponenten, es ist keine stateless Client-Server-Interaktion.

Akteure, Meta-Informationsentität und Rollen des semantischen Protokolls

In einem semantischen Protokoll nehmen die Interaktionspartner jeweils passende Rollen zueinander ein. Diese Rollen werden für ein bestimmtes Interaktionsmuster definiert. Für die unten aufgeführten Interaktionsmuster wären es folgende Rollen:

- Ausschreibungsverfahren (semantisches Protokoll „Bidding“)
  - Rolle „Service Requester“
  - Rolle „Service Provider“
- Abfragen und Senden von Daten (semantisches Protokoll „ApplicationInformationUsage“)
  - Rolle „Information User“
  - Rolle „Information Provider“

Nachfolgend wird die Nachrichten-basierte Interaktion mit beispielhaften Werten festgelegt:

Call for Capability

Die Capability-Ausschreibung wird auf das Bidding-Protokoll nach VDI 2193 abgebildet. Die Capability-Abfrage startet mit der Nachricht Call for Proposal (Mapping der Nachricht Call for Capability (Abbildung 6 bis Abbildung 9). Die gewünschte Capability, die der entsprechenden Benennung aus der gemeinsamen Ontologie oder Dictionary entspricht, wird durch das entsprechende Teilmodell in der Nachricht übergeben (hier „Drilling“).

Semantisches Protokoll

Bidding

semanticId (URI): www.admin-shell.io/interaction/bidding

Sender der Nachricht

Rolle: Service Requester

Empfänger der Nachricht

Rolle: Service Provider


```
{
   "frame": {
     "semanticProtocol": {
           “keys": [{
              "type": "GlobalReference",
              "idType": "URI",
              "value": "www.admin-shell.io/interaction/bidding",
              "local": false
           }]
           "type": "callForCapability",
           "messageId": "cfp_1",
           "sender": {
                 "identification": {
                       "id": "www.admin-shell.io/aas/1",
                       "idType": "URI"
                 },
                 "role": {
                       "name": "serviceRequester"
                 }
           },
           "receiver": {
                 "identification": {
                 "id": "www.admin-shell.io/aas/2",
                 "idType": "URI"
           },
           "role": {
                 "name": "serviceProvider"
           }
           "replyBy": "",
           "inReplyTo": "0",
           "conversationId": "Basys42AASCapabilityRequest200924"
     },
     "interactionElements": [{
           "submodels": [{
                 "hasDataSpecification": null,
                 "semanticId": {
                       "keys": []
                 },
                 "qualifiers": null,
                 "identification": {
                       "idType": "IRI",
                       "id": "www.company.com/ids/sm/1310_7132_1012_9265"
                 },
                 "administration": {
                       "version": "1.0",
                       "revision": "1.0"
                 },
                 "idShort": "RequestedCapability",
```

```
                "category": "CONSTANT",
                "modelType": {
                        "name": "Submodel"
                },
                "kind": "Instance",
                "submodelElements": [{
                        "hasDataSpecification": null,
                        "semanticId": null,
                        "constraints": null,
                        "idShort": "Drilling",
                        "category": null,
                        "modelType": {
                        "name": "Capability"
                        },
                        "kind": "Instance",
                        "descriptions": null
                }],
                "descriptions": [{
                        "language": "de",
                        "text": "enthält eine oder mehrere Capabilities, die einem Produktionsschritt benennen"
        }]
    }]
}}
```


Offer Capability

Die Antwort auf die Capability-Abfrage ist ein Ressourcenangebot mit der Nachricht Offer-Proposal (Mapping Offer Capability aus den Abbildungen Abbildung 6 bis Abbildung 9). Die angebotene Capability, die der entsprechenden Benennung aus der gemeinsamen Ontologie oder Dictionary abgeleitet ist, wird durch das entsprechende Teilmodell in der Nachricht übergeben (hier „Boring“). Außerdem wird die Referenz auf das oder die Teilmodelle mitgegeben, in denen die dazu gehörigen Skills beschrieben sind.

Anmerkung: Hier sind die Rollen von Sender und Receiver gegenüber des Call for Proposal vertauscht (serviceProvider und serviceRequester, *Anmerkung: Sevice Requester und Provider sind Rollen der Anwendung, nicht der Kommunikation*) und es wird bespielhaft nicht „Drilling“ angeboten, sondern „Boring“. Die „conversationId“ ist gleich, die „messageId“ ist um eins erhöht.

Semantisches Protokoll

Bidding

semanticId (URI): www.admin-shell.io/interaction/bidding

Sender der Nachricht

Rolle: Service Provider

Empfänger der Nachricht

Rolle: Service Requester


```
{
   "frame": {
            "semanticProtocol": {
                     "keys": [{
                              "type": "GlobalReference",
                              "idType": "URI",
                              "value": "www.admin-shell.io/interaction/bidding",
                              "local": false
                     }]
                     "type": "offerProposal",
                     "messageId": "cfp_2",
                     "sender": {
                              "identification": {
                              "id": "www.admin-shell.io/aas/2",
                              "idType": "URI"
                              },
                              "role": {
                                       "name": "serviceProvider"
                               }
                     },
                     "receiver": {
                              "identification": {
                              "id": "www.admin-shell.io/aas/2",
                              "idType": "URI"
                               },
                              "role": {
                              "name": "serviceRequester"
                              }
                     },
                     "replyBy": "",
                     "inReplyTo": "0",
                     "conversationId": "Basys42AASCapabilityRequest200924"
            },
            "interactionElements": [{
                     "submodels": [{
                              "hasDataSpecification": null,
                              "semanticId": {
                                       "keys": []
                              },
                              qualifiers": [],
                              "identification": {
                                       "idType": "IRI",
                                       "id": "www.company.com/ids/sm/0451_7132_1012_3454"
                              },
                              "administration": null,
                              "idShort": "OfferedCapability",
                              "category": "CONSTANT",
```

```
"modelType": {
        "name": "Submodel"
},
"kind": "Instance",
"submodelElements": [{
        "hasDataSpecification": null,
        "semanticId": null,
        "constraints": [],
        "idShort": "Boring",
        "category": "CONSTANT",
        "modelType": {
                “name": "Capability"
        },
        "kind": "Instance",
        "descriptions": [{
                "language": "de",
                "text": "Angebot, dass das Asset bohren kann"
        }]
        "value": {
                "keys": [{
                        "type": "Submodel",
                        "local": true,
                        "value":
                        "www.company.com/ids/sm/0110_8042_1012_1367",
                        "index": 0,
                        "idType": "IRI"
                }]
                "hasDataSpecification": null,
                "semanticId": null,
                "constraints": [],
                "idShort": "ReferenceSubmodelBoring",
                "category": "CONSTANT",
                "modelType": {
                        "name": "ReferenceElement"
                },
                "kind": null,
                "descriptions": [{
                        "language": "de",
                        "text": "beinhaltet die Referenz auf das Submodel, das die
                        technischen Daten für Boring beinhaltet"
                }]
         }
                "descriptions": [{
                        "language": "de",
                        "text": "Benennt die Capability die vom Asset, das durch
                        diese AAS
                                repräsentiert wird angeboten wird."
                }]
        }]
    }]
  ]}
}}
```

Call for Skill

Die Skill-Ausschreibung wird auf das Bidding-Protokoll nach VDI 2193 abgebildet. Die Skill-Abfrage startet mit der Nachricht Call for Proposal (Mapping der Nachricht Call for Skill aus den Abbildung 6 bis Abbildung 9). Während der Capability-Abfrage werden die Referenzen an die Teilmodelle übergeben. Diese Referenzen werden in der Abfrage-Nachricht im Bereich der „interactionElements“ eingebettet.

Semantisches Protokoll

Bidding

semanticId (URI): www.admin-shell.io/interaction/bidding

Sender der Nachricht

Rolle: Service Requester

Empfänger der Nachricht

Rolle: Service Provider


```
{
   "frame": {
          "semanticProtocol": {
                 "keys": [{
                        "type": "GlobalReference",
                        "idType": "URI",
                        "value": "www.admin-shell.io/interaction/bidding",
                        "local": false
                 }]
                 "type": "cfp",
                 "messageId": "cfp_1",
                 "sender": {
                        "identification": {
                        "id": "www.admin-shell.io/aas/1",
                        "idType": "URI"
                        },
                        "role": {
                               "name": "serviceRequester"
                        }
                 },
                 "receiver": {
                        "identification": {
                               "id": "www.admin-shell.io/aas/2",
                               "idType": "URI"
                        },
                        "role": {
                               "name": "serviceProvider"
```

```
                        }
                    },
                    "replyBy": "",
                    "inReplyTo": "0",
                    "conversationId": "Basys42AASSkillRequest210223"
                },
                "interactionElements": [{
                    "submodels": [{
                        "hasDataSpecification": null,
                        "semanticId": {
                            "keys": []
                        },
                        "qualifiers": [],
                        "identification": {
                            "idType": "IRI",
                            "id": "www.company.com/ids/sm/0110_8042_1012_1367"
                        },
                        "administration": null,
                        "modelType": {
                            "name": "Submodel"
                        },
                        "kind": "Instance",
                    }]
                }]
    }}
```


Offer Skill

Die Antwort auf die Skill-Abfrage ist ein Angebot mit der Nachricht Offer Proposal (Mapping Offer Skill aus den Abbildung 6 bis Abbildung 9). Das angebotene Teilmodell war in der Capability-Antwort mit seiner id referenziert.

("idType": "IRI", "id": "www.company.com/ids/sm/0110_8042_1012_1367")

Semantisches Protokoll

Bidding

semanticId (URI): www.admin-shell.io/interaction/bidding

Sender der Nachricht

Rolle: Service Provider

Empfänger der Nachricht

Rolle: Service Requester


```
{
   "frame": {
```

```
"semanticProtocol": {
        "keys": [{
                "type": "GlobalReference",
                "idType": "URI",
                "value": "www.admin-shell.io/interaction/bidding",
                "local": false
        }]
        "type": "offerProposal",
        "messageId": "proposal_2",
         "sender": {
                "identification": {
                "id": "www.admin-shell.io/aas/2",
                "idType": "URI"
                },
                "role": {
                        "name": "serviceProvider"
                }
        },
        "receiver": {
                "identification": {
                        "id": "www.admin-shell.io/aas/2",
                        "idType": "URI"
                },
                "role": {
                        "name": "serviceRequester"
                }
        "replyBy": "",
        "inReplyTo": "0",
        "conversationId": "Basys42AASSkillRequest210223"
},
"interactionElements": [{
        "submodels": [{
                "hasDataSpecification": null,
                "semanticId": {
                        "keys": [{
                                "type": "Submodel",
                                "local": false,
                                "value": "0173-1#01- AKG243014",
                                "index": 0,
                                "idType": "IRDI"
                        }]
                },
                "qualifiers": null,
                "identification": {
                        "idType": "IRI",
                        "id": "www.company.com/ids/sm/0110_8042_1012_1367"
                },
                "administration": {
                        "version": "0.1",
                        "revision": "0.1"
                },
```

```
"idShort": "Technical Data Bohren",
"category": "",
"modelType": {
        "name": "Submodel"
},
"kind": "Instance",
"submodelElements": [{
        "value": "5",
        "valueId": null,
        "hasDataSpecification": null,
        "semanticId": {
                "keys": [{
                        "type": "Property",
                        "local": false,
                        "value": "0173-1#02-BAB216",
                        "index": 0,
                        "idType": "IRDI"
                }]
        "constraints": [],
        "idShort": "Senkdurchmesser",
        "category": "CONSTANT",
        "modelType": {
                "name": "Property"
        },
        "valueType": {
                "dataObjectType": {
                        "name": "float"
                }
        },
        "kind": "Instance",
        "descriptions": [{
                "language": "EN",
                "text": "Diameter of the cylindrical bore of a hole in the center of a tool
                        or tool holder that can receive a screw head to create a con-
nection"
                }]
        }]
}
{
        "value": "20",
        "valueId": null,
        "hasDataSpecification": null,
        "semanticId": {
                "keys": [{
                        "type": "Property",
                        "local": false,
                        "value": "0173-1#02-AAA014",
                        "index": 0,
                        "idType": "IRDI"
                }]
        },
        "constraints": [],
```

```
                                    "idShort": "Senktiefe",
                                    "category": "CONSTANT",
                                    "modelType": {
                                            "name": "Property"
                                    },
                                    "valueType": {
                                            "dataObjectType": {
                                                    "name": "float"
                                            }
                                    },
                                    "kind": "Instance",
                                    "descriptions": [{
                                            "language": "EN",
                                            "text":
                                            "Depth of the cylindrical countersinking of a hole in the center of a
                                            tool or tool holder that can receive a screw head to create a con-
                                            nection"
                                            }]
                            }],
                            "descriptions": []
                    }]
            }]
}}
```


Get Info

Es müssen noch zusätzliche Informationen aus dem Feasibility-Modell geholt werden. Für Informationsabfragen ist das Protokoll „ApplicationInformationUsage“ vorgesehen (ist noch in Bearbeitung in GMA 7.20). Die Informationsabfrage startet mit der Nachricht Get (Mapping der Nachricht Get Info aus den Abbildung 6 bis Abbildung 9). Nachrichteninhalte sind hier nicht weiter angegeben.

Semantisches Protokoll

ApplicationInformationUsage

semanticId (URI): www.admin-shell.io/interaction/ApplicationInformationUsage

Sender der Nachricht

Rolle: Information User

Empfänger der Nachricht

Rolle: Information Provider


```
{
   "frame": {
```

```
        "semanticProtocol": {
            "keys": [{
                        "type": "GlobalReference",
                        "idType": "URI",
                        "value": "www.admin-shell.io/interaction/ApplicationInformationUsage",
                        "local": false
            }]
            "type": "Get",
            "messageId": "Get_1",
            "sender": {
                        "identification": {
                        "id": "www.admin-shell.io/aas/1",
                        "idType": "URI"
                        },
                        "role": {
                        "name": "informationUser"
                        }
            },
            "receiver": {
                        "identification": {
                                    "id": "www.admin-shell.io/aas/2",
                                    "idType": "URI"
                        },
                        "role": {
                                    "name": "informationProvider"
                        }
            },
            "replyBy": "",
            "inReplyTo": "0",
            "conversationId": "Basys42AASSkillRequest210223"
    },
    "interactionElements": [{
          "submodels": [{
          …..
      ]}
    ]}.
}}
```


Inform Info

Es müssen noch zusätzliche Informationen aus dem Feasibility-Modell geholt werden. Für Informationsabfragen ist das Protokoll „ApplicationInformationUsage“ vorgesehen (ist noch in Bearbeitung in GMA 7.20). Die Informationsantwort wird mit der Nachricht „Show“ gegeben (Mapping der Nachricht Inform Info aus den Abbildung 6 bis Abbildung 9). Nachrichteninhalte sind hier nicht weiter angegeben.

Semantisches Protokoll

ApplicationInformationUsage

semanticId (URI): www.admin-shell.io/interaction/ApplicationInformationUsage

Sender der Nachricht

Rolle: Information Provider

Empfänger der Nachricht

Rolle: Information User

```
{
   "frame": {
      "semanticProtocol": {
            "keys": [{
               "type": "GlobalReference",
               "idType": "URI",
               "value": "www.admin-shell.io/interaction/ApplicationInformationUsage",
               "local": false
            }]
           "type": "Show",
            "messageId": "show_2",
            "sender": {
                        "identification": {
                        "id": "www.admin-shell.io/aas/2",
                        "idType": "URI"
                        },
                        "role": {
                                    "name": "serviceProvider"
                        }
            },
            "receiver": {
                        "identification": {
                        "id": "www.admin-shell.io/aas/2",
                        "idType": "URI"
                        },
                        "role": {
                                    "name": "serviceRequester"
                        }
            },
            "replyBy": "",
            "inReplyTo": "0",
            "conversationId": "Basys42AASSkillRequest210223"
      },
            "interactionElements": [{
                        "submodels": [{
                        …..
                        ]}
            ]}.
}}
```

4 Literatur

BaSys (2021). Blohm, P. et al.: Deliverable 2.7 - Spezifikation Generisches Architekturrahmenwerk für Machbarkeitstests. 2021.

BaSys (2019). Projektantrag Basissystem Industrie 4.2 - BaSys 4.2. 2019.

Beier M. (2018). Beier, B.: Konzipierung und prototypische Umsetzung einer I4.0-Komponente. Masterarbeit Otto von Guericke Universität Magdeburg, März 2018

BMWi Hrsg. (2021): Glossar der Plattform I4.0. Online Veröffentlichung. Ständiges Update. eingesehen Februar 2021. https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/Glossar/glossar.html

BMWi Hrsg. (2020a). Plattform Industrie 4.0 White Paper - Describing Capabilities of Industrie 4.0 Components. Dezember 2020. https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Capabilities_Industrie40_Components.pdf?__blob=publicationFile&v=3

BMWi Hrgs. (2020b). Federal Ministry for Economic Affairs and Energy (BMWi) - Public Relations (publisher). Plattform I4.0 Text and Editing: Details of the Asset Administration Shell Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0). https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html%20

BMWi Hrsg. (2018). I4.0-Sprache – Vokabular, Nachrichtenstruktur und semantische Interaktionsprotokolle der I4.0-Sprache. November 2018.

BMWi Hrsg. (2017a). Anwendungsszenario trifft Praxis: Auftragsgesteuerte Produktion eines individuellen Fahrradlenkers. April 2017.

BMWi Hrsg. (2017b). Beziehungen zwischen I4.0-Komponenten – Verbundkomponenten und intelligente Produktion Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente SG Modelle und Standards. Ergebnispapier Juni 2017. http://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/beziehungen-%20i40-komponenten.pdf?__blob=publicationFile&v=7

BMWi Hrsg. (2017c). Anwendungsszenario trifft Praxis: Auftragsgesteuerte Produktion eines individuellen Fahrradlenkers. April 2017.

Diedrich, Ch. Hadlich, Th. Thron, M. (2015). Christian Diedrich, Thomas Hadlich, Mario Thron: Semantik durch Merkmale für I40. Beitrag in B. Vogel-Heuser et al. (Hrsg.), Handbuch Industrie 4.0, Springer NachschlageWissen, DOI 10.1007/978-3-662-45537-1_63-1. Online ISBN 978-3-662-45537-1.

Epple (2011). U. Epple: Merkmale als Grundlage der Interoperabilität technischer Systeme. In: at - Automa-tisierungstechnik 59 (2011) Nr.7, S. 440–450.

IEC 62832 (2017). IEC: Industrial-process measurement, control and automation – Digital Factory framework – Part 1: General principles. Geneva 2017.

VDI (2019a) VDE 2193 Blatt 1: *Sprache für I4.0-Komponenten - Struktur von Nachrichten* Beuth Verlag 2019.

VDI (2019b) VDE 2193 Blatt 2: Sprache für I4.0-Komponenten - Interaktionsprotokoll für Ausschreibungsverfahren. Beuth Verlag 2019.

5 Anhang

5.1 Prinzip des Ausschreibungsverfahrens nach VDI/VDE 2193[1]

5.1.1 Interaktionsprotokolle

5.1.1.1 Anforderungen an eine I4.0-Interaktionssemantik

Ein semantisches Interaktionsprotokoll definiert die Abfolge von ausgetauschten Nachrichten zwischen I4.0-Komponenten für einen bestimmten Anwendungsfall. Unter einem Interaktionsprotokoll verstehen die Autoren ein Regelwerk, nach dem einzelne Konversationen ablaufen können. Die Interaktionsprotokolle sorgen dafür, dass ein Sprecherwechsel stattfindet, die Beiträge von interagierenden Parteien auf einander bezogen sind und ein Dialog ein erkennbares Ziel hat.

Eine I4.0-Interaktionssemantik muss somit die folgenden Voraussetzungen erfüllen:

- Begrenzung der Interaktion
- Konstruktive Interaktion
- Interaktion über dieselbe Thematik
- Kontextbezug der Interaktion
- Einheitliche, standardisierte Aktionen

5.1.1.2 Interaktionsprotokoll Ausschreibungsverfahren

Das in der VDI/VDE 2193-2 definierte Interaktionsprotokoll „Ausschreibungsverfahren“ bietet die Möglichkeit einer hochflexiblen Erzeugung von Kooperationsbeziehungen zwischen I4.0-Komponenten, insbesondere wenn die Aufgaben verteilt werden müssen. Genau solche dynamischen Beziehungen sollen im Sinne der Konzepte der wandlungsfähigen Fabrik und flexiblen Auftragsfertigung kostengünstig realisiert werden, um freie, interne oder externe Fertigungskapazitäten möglichst selbstständig und automatisiert in die eigenen Unternehmensabläufe einbinden zu können. Einzelne Prozessschritte in der Produktion können so wesentlich flexibler miteinander kombiniert und ihre spezifischen Fähigkeiten besser genutzt werden [BMWi Hrsg. (2017c)].

[1] Der Text dieses Kapitels beruht auf der VDI/VDE 2193 Teil 1 und 2 ([VDI (2019a)] und [VDI (2019b)]) sowie auf dem Diskussionspapier „Interaktionsmodell für Industrie 4.0-Komponenten“ [BMWi Hrsg. (2018)].

Das Interaktionsprotokoll „Ausschreibungsverfahren“ lehnt sich an das Kontraktnetzprotokoll [Beier M. (2018)] an und ist beispielhaft auf der Abbildung 12 dargestellt. Das Kontraktnetzprotokoll ist einer der verbreitetsten Ansätze zur formalen Beschreibung von automatisierten Verhandlungen.

Sofern die I4.0-Komponenten das Interaktionsprotokoll „Ausschreibungsverfahren“ unterstützen, können sie sofort Aufgaben übernehmen, indem sie auf Ausschreibungen reagieren oder selbst Aufgaben zur Bearbeitung ausschreiben. Die Ausschreibung und das Angebot beinhalten eine technische und kaufmännische Beschreibung der zu vereinbarenden Aufgabe, die mit dem Vokabular der I4.0 verfasst wird.

Die interagierenden Komponenten werden in zwei Gruppen aufgeteilt. Eine anfragende Komponente (Auftraggeber) spaltet die zu lösende Aufgabe in die Teilaufgaben auf und sucht nach anderen Komponenten, die diese Teilaufgaben erledigen können.

Die Besonderheit gegenüber den starren, hierarchischen Strukturen besteht darin, dass keine zentrale Kontrolle über die Aufgabenausführung existiert und die Verbindungen zwischen I4.0-Komponenten nicht vom zentralen Element des Systems fest vorgegeben sind. Vielmehr kann je nach Aufgabe jede Komponente in Verbindung mit beliebigen anderen I4.0-Komponenten treten, die dann spontan die Rolle des Anbieters oder Erbringers einer Dienstleistung übernehmen.

Die angefragten Komponenten (Auftragnehmer, in der Regel die Anbieter von Dienstleistungen) übernehmen die Teilaufgaben oder werden selbst zu anfragenden Komponenten und zerlegen die Teilaufgaben weiter, um sie an andere I4.0-Komponenten zu delegieren.

Eine anfragende I4.0-Komponente hat die Möglichkeit, nach einem neuen Kooperationspartner zu suchen, falls die zuvor kooperierende Komponente nicht in der Lage ist, eine zufriedenstellende Lösung zu liefern.

Anhand von eingegangenen Angeboten kann die anfragende I4.0-Komponente entscheiden, ob die nächste Iteration endgültig ist und ein Teil der Angebote angenommen und der Rest abgelehnt wird.

Alternativ kann die anfragende Komponente eine Verhandlung starten und neue Konditionen aushandeln. Dafür wiederholt die angefragte Komponente den Prozess, indem sie eine modifizierte Ausschreibung herausgibt.

Die Absicht ist, dass die anfragende Komponente versucht, bessere Gebote von den potenziellen Kooperationspartnern zu erhalten, indem sie die Ausschreibung überarbeitet und neue Angebote anfordert.

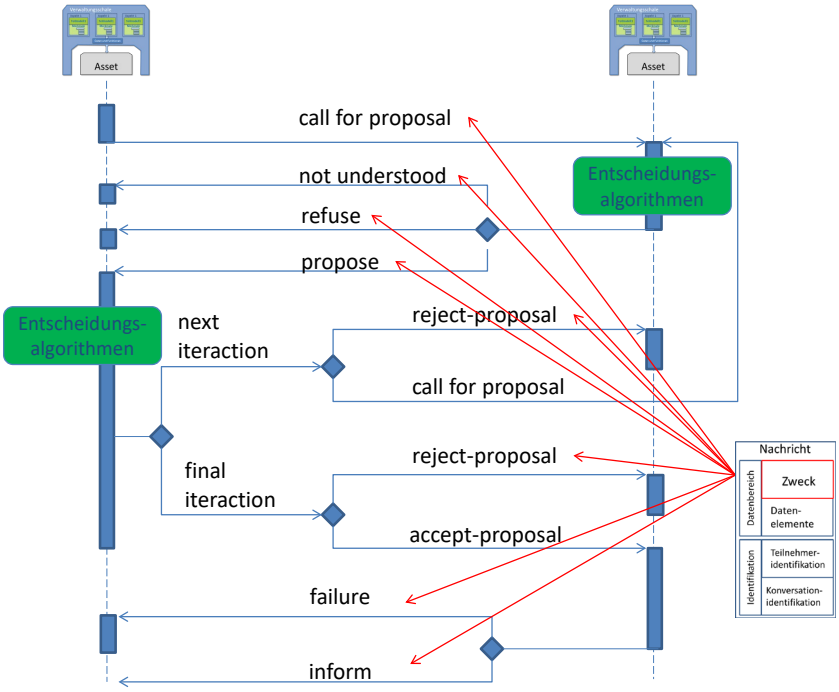

Abbildung 12: Interaktionsprotokoll "Ausschreibungsverfahren" [BMWi Hrsg. (2018)]

Legende zu Abbildung 12

call for proposal

Mit dem „call for proposal“ (kurz CFP) wird eine I4.0-Komponente aufgefordert, ein Angebot abzugeben. Dieses Angebot bezieht sich auf das übermittelte Substantiv. Antworten auf einen CFP könnten beispielsweise *refuse*, *not-understood* oder *propose* sein. [VDI (2019a)]

Die Nachrichtennamen in dem Sequenz-Diagramm verkörpern den Typ der Nachricht in dem Interaktionsdialog

refuse

Ein „refuse“ bedeutet, dass eine Komponente eine gewünschte Operation nicht ausführen kann oder will. [VDI (2019a)]

not-understood

„not-understood“ sagt aus, dass eine Komponente die andere nicht verstanden hat. In Bezug auf das Ausschreibungsverfahren wird dieses Verb verwendet, wenn eine Komponente das angefragte Teilmodell nicht besitzt und somit nicht weiß, welche Aktion durch die Anfrage ausgelöst werden soll. [VDI (2019a)]

propose

Im Ausschreibungsverfahren bedeutet ein „propose“, dass eine I4.0-Komponente ein Angebot für eine andere Komponente bereitstellt. Eine Komponente bietet somit an, eine angefragte Aktion auszuführen. [VDI (2019a)]

accept-proposal und reject-proposal

Mit „accept-proposal“ und „reject-proposal“ wird einer Komponente, welche ein Angebot abgegeben hat, mitgeteilt, ob ihr Angebot angenommen oder abgelehnt wurde. [VDI (2019a)]

inform

„inform“ sendet eine Information von einer Verwaltungsschale zu einer weiteren. Beispielsweise wird im Ausschreibungsverfahren dieses Verb benutzt, um mitzuteilen, wie der Auftragsstatus zu einem vergebenen Auftrag ist. [VDI (2019a)]

Die Interaktionsprotokolle bieten einen Kontext für die einzelnen Nachrichten und vermitteln den Nachrichten eine Bedeutung.

Es gibt zwei Referenzen der prototypischen Umsetzung, die hier referenziert werden können.

- In einem Forschungsprojekt wurde auf der Basis des Ausschreibungsverfahrens eine Produktionssteuerung umgesetzt. Die Produktionsanlage wurde simuliert und die Steuerung wurde von VWS mit Ausschreibungsverfahren realisiert. Ein Video mit einer entsprechenden Beschreibung kann unter (https://www.youtube.com/watch?v=cWT6OqxpIjU) eingesehen werden.
- Das Ausschreibungsverfahren wurde auch verwendet, um einen dezentralen Industrie-Marktplatz aufzubauen. Der Marktplatz basiert auf der Distributed Ledger Technologie „IOTA“ und verwendet die Verwaltungsschalen und das Ausschreibungsverfahren für die Angebots- und Auftragsabwicklung. Die referenzierte Webseite bietet weitere Information zum industriellen Marktplatz. Ein eingebettetes Video mit einer entsprechenden Beschreibung kann unter (https://blog.iota.org/iota-foundation-launches-industry-marketplace-the-worlds-first-autonomous-and-decentralized-80d9290318f3?gi=c25789946a59) eingesehen werden.

5.1.1.3 Entscheidungsfähigkeit

Wird die Zusammenarbeit von I4.0-Komponenten nach dem Interaktionsprotokoll „Ausschreibungsverfahren“ orchestriert, entstehen mindestens zwei Situationen, in denen Verwaltungsschalen Entscheidungen treffen müssen. Dies betrifft zum einen die Entscheidung, ob ein Asset in der Lage ist, eine Anfrage zu bearbeiten. Dafür werden technische und wirtschaftliche Aspekte betrachtet, welche eine Entscheidung erfordern. Dies ist z.B. bei der Angebotserstellung der Fall. Zum anderen muss die anfragende Komponente entscheiden, welches der eingegangenen Angebote angenommen wird. Die Algorithmen können mehrere Ziele verfolgen. Beispielsweise sollen die besten wirtschaftlichen Konditionen ausgehandelt werden. Das übergeordnete Ziel ist dabei, dass Komponenten autonom handeln und somit das menschliche Verhalten imitieren sollen.

In dem hier vorgestellten Interaktionsmechanismus können die Entscheidungsalgorithmen relativ einfach gehalten werden. Beispielsweise vergleicht ein Algorithmus in simpler Weise die Merkmale miteinander, prüft die Anforderungs- und Zusicherungsaussagen und sucht nach einem besten Preis.

5.1.2 Konzeptvorstellung der I4.0 Sprache

5.1.2.1 Generelle Struktur

Um den sich aus dem I4.0-Konzept ergebenden Anforderungen (z.B. [BMWi Hrsg. (2017a)], [BMWi Hrsg. (2017b)]) gerecht zu werden, wird die I4.0-Sprache als ein Regelsystem, bestehend aus drei Ebenen betrachtet (Abbildung 13). Der Begriff I4.0-Sprache wird als übergreifende Bezeichnung für die verschiedenen Aspekte des gegenseitigen Verstehens von I4.0-Komponenten verwendet. Er repräsentiert die Sprache mehr im linguistischen Sinne und nicht die formale Definition für die Anwendung in Softwarewerkzeugen.

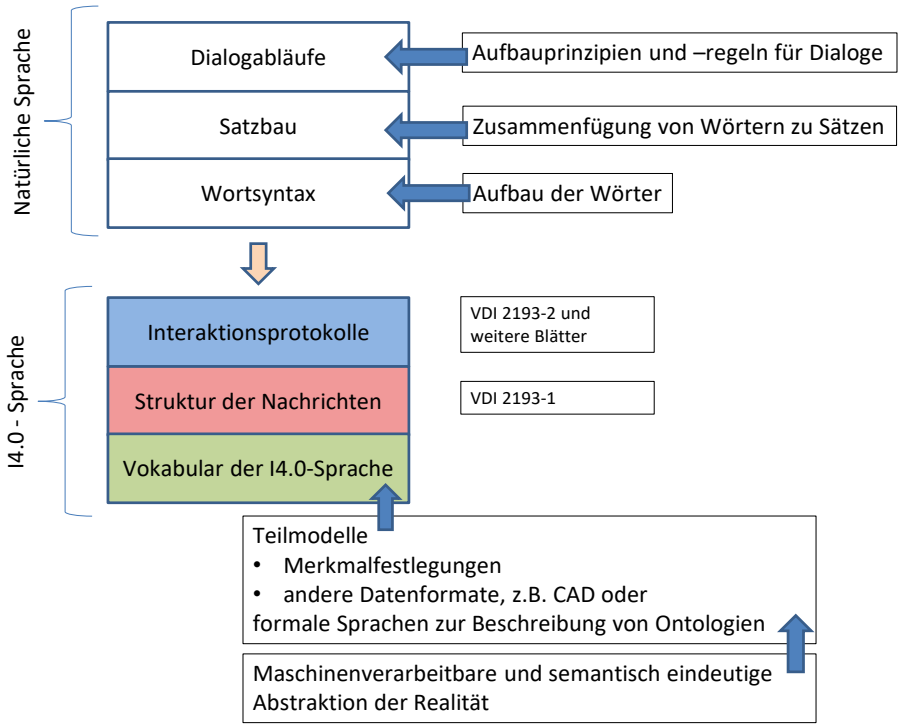

Abbildung 13: Einteilung der I4.0-Sprache [BMWi Hrsg. (2018)]

Die I4.0-Sprache zwischen I4.0-Komponenten wird untergliedert in:

- Vokabular der Sprache: Die Merkmale, Merkmallisten, oder andere Annotationsformen von Datenelementen, die in den Nachrichten verwendet werden. → entspricht der Wortsyntax (siehe 5.1.2.2)
- Struktur der Nachrichten: Struktur, die die Anordnungen der Inhalte und die für deren gegenseitige Einordnung notwendigen Elemente organisiert. → entspricht dem Satzbau (siehe 5.1.2.3)
- Interaktionsprotokolle: Abläufe in den Dialogen zwischen I4.0-Komponenten, die die zu erfüllenden Aufgaben organisieren. → entspricht dem Dialogablauf (siehe 5.1.1)

Die verschiedenen Aspekte werden in der Richtlinienserie VDI/VDE 2193 mit den folgenden Inhalten definiert Abbildung 13):

- VDI/VDE 2193-1: Beschreibt den Aufbau der Nachrichten. In diesen befinden sich die Merkmale der Teilmodelle der Verwaltungsschale. Diese bilden das wesentliche Vokabular der I4.0-Sprache. Die Definition der Verwaltungsschale und der Teilmodelle erfolgt in [BWWi Hrsg. (2020b)].
- VDI/VDE 2193-2: Beschreibt ein wichtiges Interaktionsprotokoll, das Ausschreibungsverfahren.
- VDI/VDE 2193-3: Interaktionsprinzipien für vertikale und horizontale Interaktionen zwischen I4.0-Komponenten

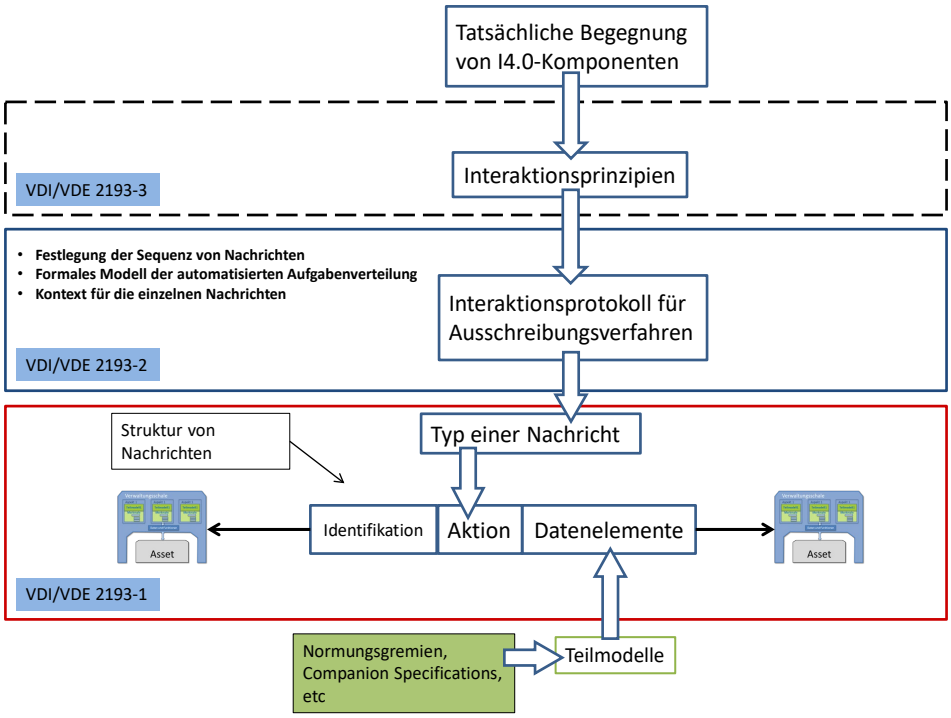

Abbildung 14: Übersicht der Richtlinienserie VDI/VDE 2193 [BMWi Hrsg. (2018)]

5.1.2.2 Vokabular der I4.0-Sprache

Will man ein Asset beschreiben, kann dies durch seine Eigenschaften und deren Beziehungen erfolgen (Abbildung 15). Eine Schraube hat z.B. eine Länge, einen Gewindedurchmesser, einen bestimmten Kopftyp (z.B. Kreuzschlitz) und einen Materialtyp. Für die Nutzung der Schraube sind dies wesentliche Eigenschaften, die beispielsweise für die Auswahl, aber auch für die Bestimmung des notwendigen Werkzeugs und die erlaubten Drehmomente, oder die Behältergröße für deren Lagerung benötigt werden. Für die Verarbeitung in digitalen Systemen müssen die Eigenschaften auch maschinenlesbar ausgewertet werden können. Dafür ist eine Notation mit eindeutigen Inhalten und Formaten notwendig. Bei Eigenschaften, die in dieser digitalen Form verfügbar sind, spricht man von Merkmalen. Für tiefergehende Information zu dem Merkmalmodell siehe auch [Diedrich, Ch. Hadlich, Th. Thron, M. (2015)] und [Epple (2011)].

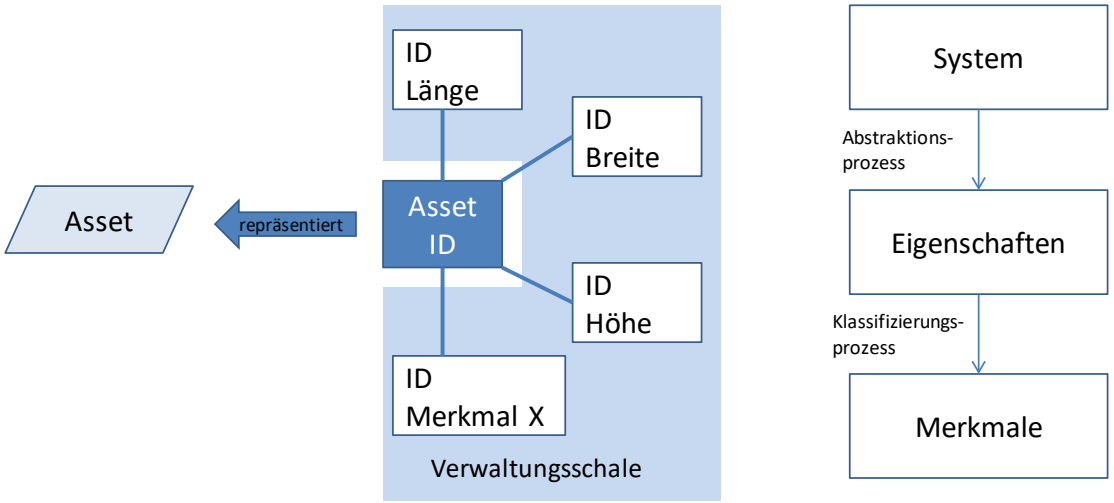

Abbildung 15: Beschreibung von Assets durch ihre Eigenschaften [BMWi Hrsg. (2018)]

Die Merkmalwerte sollen auch während des operativen Betriebs ausgetauscht werden. Nicht der Mensch nutzt die Werte und prüft deren Plausibilität, sondern das wird durch die Anwendungen selbst vorgenommen. Mit anderen Worten, die Merkmale benötigen auch instanzbezogene Attribute. Dies kommt durch die Nutzung von Merkmalen in der Interaktion zwischen I4.0-Komponenten zum Merkmalkonzept hinzu. Daraus folgt aber auch, dass diese Attribute standardisiert vorliegen müssen. Nähere Informationen zur Nutzung von Merkmalen als Typen und Instanzen sind in dem IEC-Standard Digital Factory – IEC 62832 beschrieben [IEC 62832 (2017)]. Aktivitäten zur Definition und zur Nutzung der Merkmalbeschreibung als Instanzbeschreibungen sind im Rahmen von IEC 61987 und in einer neu entstehenden DIN-SPEC 29000 vorhanden.

Für die Nutzung anderer technologischer Konzepte als lineare Merkmalskataloge für die Notation des Vokabulars ist das Arbeitspaket 2 des BaSys 4.2 Projektes zuständig.

5.1.2.3 Struktur von Nachrichten

Der Satzbau der I4.0-Sprache ist ein Regelwerk, mit dem das Vokabular der I4.0-Sprache in die Nachrichten eingebettet wird. Damit sollen die Voraussetzungen für ein gemeinsames Verständnis der Nachrichten geschaffen werden.

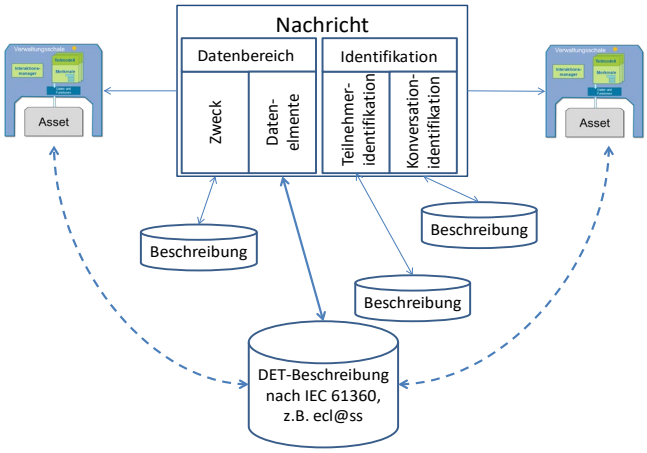

Abbildung 16: Struktur von Nachrichten [BMWi Hrsg. (2018)]

Der Informationsaustausch zwischen I4.0-Komponenten erfolgt nachrichtenbasiert. Damit die Nachrichten von den interagierenden Komponenten korrekt verstanden und bearbeitet werden können, wird in der I4.0-Sprache eine Struktur von Nachrichten festgelegt.

Eine Nachricht besteht aus einem Identifikations- und einem Datenbereich.

Identifikationsbereich

> Durch die Eingabe von Absender und Empfänger einer Nachricht werden die Teilnehmer einer Konversation identifiziert. Es besteht die Möglichkeit, die einzelnen Konversationen und Nachrichten ebenfalls zu identifizieren und zu referenzieren (analog zu dem in der Geschäftswelt üblichen: „Ich beziehe mich auf Ihre Nachricht vom…“). Dadurch wird die Ausdrucksstärke der I4.0-Sprache an den für Dialoge notwendigen Umfang angepasst.

Datenbereich

> Ein Zweck definiert die Intention der Nachricht, wie z.B. „call for proposal“, „accept-propose“ oder „inform“. Die Datenelemente referenzieren einerseits Eigenschafts-, Parameter-, Zustands-, Fähigkeits- und weitere relevanten Informationen, andererseits können sie auch Werte enthalten, die den Teilmodellen von Assets zugeordnet sind.