Vorschlag zur systematischen Klassifikation von Interaktionen in Industrie 4.0 Systemen

Hinführung zu einem Referenzmodell für semantische Interoperabilität | White Paper

www.bitkom.org

PLATTFORM INDUSTRIE 4.0 | bitkom

Herausgeber

Bitkom
Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e. V.
Albrechtstraße 10 | 10117 Berlin
T 030 27576-0
bitkom@bitkom.org
www.bitkom.org

Ansprechpartner

Dr. Katharina Eylers | Referentin Industrie 4.0 & Technische Regulierung
T 030 27576-220 | k.eylers@bitkom.org

Satz & Layout

Katrin Krause | Bitkom e. V.

Titelbild

© Scott Webb | pexels.com

Inhaltsverzeichnis

1 Kurzzusammenfassung

1 Kurzzusammenfassung

Im Rahmen der digitalen Vernetzung nähern sich die etablierte Automatisierungs- und Betriebstechnik und die moderne Informations- und Kommunikationstechnologie, unter dem Begriff der Industrie 4.0, weiter an. Neben Problemen, dass teilweise stark divergierende technologische Ansätze aufeinandertreffen, stellt insbesondere der interoperable Austausch von Informationen über Anwendungsdomänen eine Herausforderung dar. Aktuelle Ansätze adressieren bereits einige Aspekte im Kontext der Fertigung, aus Anwendungssicht bestehen jedoch noch wichtige ungeklärte Fragen.

In diesem Beitrag schlagen wir vor, die Antworten auf diese Fragen entlang eines von uns sogenannten »Referenzmodells semantische Interoperabilität« zu entwickeln. Es beruht auf der Vorstellung, dass zur Beurteilung der Interoperabilität verschiedener Systeme gewisse Festlegungen bezüglich ihres Transformationsverhaltens, also ihrer jeweiligen Verarbeitung der ausgetauschten Informationen, notwendig sind. Entsprechend beruht das Modell im Kern auf einer einfachen Klassifikation der möglichen Interaktionen bezüglich der beiden Dimensionen des Informationstransports und der Informationsverarbeitung, wobei letztere weiter aufgeteilt wird hinsichtlich derjenigen Eigenschaften, von denen wir wissen, dass sie die Gestalt von Interfaces bestimmen: Synchronität, Zustandsbehaftung und Determinismus.

Das wesentliche Ergebnis der Untersuchung ist:

1. Ein Interface zwischen zwei Systemen ist im Sinne des Referenzmodells semantischer Interoperabilität nur soweit wohldefiniert, als es eine Aussage über das Transformationsverhalten, also die Abbildungseigenschaften der Systeme macht. Denn nur dann lassen sich weitergehende Aussagen über »Interoperabilität« in einem semantischen Sinn, also bezogen auf die jeweilige Verarbeitung der ausgetauschten Informationen, machen.

2. Die Ausprägungen der Informationstransportkategorien uni-/bidirektional und der Informationsverarbeitungskategorien A-/Synchronität, Zustandsbehaftung und Nicht-/Determinismus schlägt sich in der Gestalt der Interfaces nieder, weswegen es verschiedene Interfaceformen gibt, die es technologisch zu unterstützen gilt.

3. Horizontale Interaktionen im Sinne des vorgestellten Referenzmodells semantischer Interoperabilität zeichnen sich dadurch aus, dass sich alle Interaktionsakteure grundsätzlich gleich bzgl. des Transports und v.a. in der Verarbeitung der ausgetauschten Informationen verhalten, nämlich asynchron, zustandsbehaftet und nichtdeterministisch. Vertikale Interaktion liegt vor, wenn sich bei bidirektionalem Informationsaustausch die Akteure grundsätzlich asymmetrisch bzgl. der Informationsverarbeitung verhalten.

4. Horizontale Interaktionen zwischen Applikationen unterschiedlicher Domänen bilden den Kern der immer größer und dichter werdenden Interaktionsnetzwerke.

5. Zerlegt man das Problem der aufwandsarmen Herstellung von Interoperabilität in das Problem des Informationstransports (bzw. der Kommunikation) und der Informationsverarbeitung, so stellt man fest, dass das Problem der Kommunikation vergleichsweise gut ver-

standen ist und standardisierte Lösungen hierfür verfügbar sind. Das Problem der aufwandsarmen Herstellung von aufeinander abgestimmter Informationsverarbeitung insbesondere in komplexen, zustandsbehafteten Applikationen, die an vielen (horizontalen) Interaktionen beteiligt sind, ist hingegen bisher weniger gut verstanden und ganz wesentlich ein Problem der Architektur der beteiligten Applikationen, die möglichst robust gegen Änderungen in einzelnen Interaktionen sein sollte.

6. Die technologische Unterstützung sowohl für unidirektionale Interaktionen als auch für hierarchische Interaktion ist recht ausgereift. Für die horizontalen Interaktionen im Bereich IoT im Allgemeinen oder auch I4.0 im Speziellen, sind entsprechende Standards leider noch nicht so weit verbreitet eingesetzt.

2 Einleitung

2 Einleitung

Neue Anwendungsfelder erschließen – die Transparenz existierender Prozesse ermöglichen – eine Optimierung der gesamten Wertschöpfungsnetze erreichen. Diese und viele weitere Zielsetzungen werden im Kontext der Digitalisierung der Wirtschaft genannt. Mit der zunehmenden Vernetzung tritt naturgemäß das Thema der Interoperabilität stark in den Vordergrund. Hinzu kommt, dass – im Gegensatz zur Anfangszeit des Internets, in dem semantisch agnostische Transportprotokolle wie HTTP, FTP, SMTP, etc. die Wachstumstreiber waren und die Verarbeitung der über das Internet transportierten Informationen im Wesentlichen in der Domäne des menschlichen Geistes verblieb – mittlerweile technische informationsverarbeitende Systeme mehr und mehr mit einer gewissen Autonomie in die entsprechenden inhaltlichen Interaktionen einbezogen werden.

Damit ergibt sich zum einen das technische Problem, wie die gewünschten Interoperabilität tatsächlich möglichst aufwandsarm hergestellt werden kann, aber auch das soziale Problem, dass immer mehr – primär nichtinformatische – Fachgebiete von dieser Entwicklung inhaltlich betroffen sind und es zunehmend wichtiger wird, dass sich alle Beteiligten der unterschiedlichen Wissensdomänen auf ein entsprechendes Referenzmodell im Sinne eines konzeptuellen Rahmens aus gemeinsamen Begriffen und Terminologien einigen können, wenn es um das Thema der Interoperabilität geht.

Hier ist aus Sicht der Bitkom die Informatik gefragt. Um diese Herausforderung zu adressieren wird ein Referenzmodell für die Interoperabilität von Systemen vorgeschlagen. Im Unterschied zu anderen Ansätzen beruht es auf einem System- und entsprechenden Interface-Konzept, das das Transformationsverhalten, also die Abbildungseigenschaften in den Vordergrund rückt. Darauf bezogen wird eine einfache Klassifikation der Interaktion zwischen Systemen. ins Zentrum gestellt. Insbesondere die Frage, ob die Systeme sich bezüglich der Interaktion deterministisch oder nichtdeterministisch verhalten, spielt dabei eine entscheidende Rolle.

Die Identifikation der verschiedenen Interaktionsklassen erweist sich in der Praxis als ausgesprochen hilfreich, da sie sich deutlich hinsichtlich ihrer notwendigen Vereinbarung zur Sicherstellung der Interoperabilität, wie auch bezüglich der sie unterstützenden Technologien unterscheiden.

Die Zielgruppe dieses Dokuments sind daher alle Experten aus den unterschiedlichen Wissensdomänen, die sich mit Fragen der semantischen Interoperabilität informatischer Systeme auseinandersetzen. Das Ziel des Dokuments ist, ihnen ein konzeptuelles Rahmenwerk anzubieten, mit dem sie die dabei auftretenden Probleme und Herausforderungen besser miteinander diskutieren und verstehen können.

Der weitere Aufbau des Dokuments ist folgendermaßen: In Abschnitt 3 wird das Referenzmodell semantischer Interoperabilität vorgestellt, auf dem dieser Beitrag wesentlich basiert. In Abschnitt 4 verweisen wir auf einige weitere Aspekte, nämlich Verantwortlichkeit, Sicherheit und Datentypisierung, die sich nahtlos in das vorgestellte konzeptuelle Rahmenwerk einfügen. In Abschnitt 5 stellen wir den Bezug zu anderen Referenzmodellen her, die im Kontext der Diskussionen über Industrie 4.0 eine wesentliche Rolle spielen. Dasselbe tun wir in Abschnitt 6 zu sogenannten »Software Design/Architectural Styles«. In Abschnitt 7 geben wir 2 unterschiedlich detaillierte

Beispiele, die die Relevanz und Anwendbarkeit des Referenzmodells belegen. In Abschnitt 8 diskutieren wir kurz eine Auswahl bestehender Technologien entlang des Referenzmodells. Und in Abschnitt 9 fassen wir die Ergebnisse zusammen und ordnen sie in den Gesamtkontext der gegenwärtigen Diskussion zu Industrie 4.0 ein.

3 Referenzmodell semantischer Interoperabilität

3 Referenzmodell semantischer Interoperabilität

Ein Referenzmodell ist ein konzeptueller Rahmen, das es erlaubt mit einer verhältnismäßig kleinen Anzahl von Konzepten die relevanten Beziehungen einer Wissensdomäne zu verstehen[1]. Die Relevanz eines solchen Referenzmodells bemisst sich daher nach der Mächtigkeit seiner Konzepte im Sinne der tatsächlich ableitbaren praktischen Konsequenzen.

Der Glossar Industrie 4.0[2] beschreibt Interoperabilität als: »Fähigkeit zur aktiven, zweckgebundenen Zusammenarbeit von verschiedenen Komponenten, Systemen, Techniken oder Organisationen, Interoperation ist realisierte Zusammenarbeit«. Die Interoperabilität von informationsverarbeitenden Komponenten bedeutet daher, dass sie zum einen Informationen austauschen und zum anderen, dass ihre Informationsverarbeitung »sinnvoll« aufeinander abgestimmt ist. Die Verarbeitung von Informationen durch ein System entspricht einer Transformation seiner Eingabewerte, ggfs. in Verbindung mit Werten innerer Zustände auf seine Ausgabewerte. Entsprechend kann über die Schnittstellen zwischen den Systemen innerhalb ihrer Interaktionen, die sogenannten Interfaces, nur soweit klar geredet werden, als das Transformationsverhalten der Systeme bezogen auf die Interaktionen klar ist. Nur dann lassen sich Aussagen über Interoperabilität auf einer inhaltlichen Ebene machen, also Aussagen über die Beziehung der Informationsverarbeitung verschiedener, miteinander informationsaustauschender Systeme. Um diesen inhaltlichen Aspekt der Interoperabilität zu betonen sprechen wir auch von »semantischer« Interoperabilität.

Das in diesem Dokument eingeführte Referenzmodell semantischer Interoperabilität beruht auf dem im nächsten Abschnitt beschriebenen Interaktionsmodell, das schon der VDI/VDE-Richtlinie 2193 zugrunde liegt. Tatsächlich lässt sich anhand dieses Modells die Informationsverarbeitung innerhalb einer Komponente in Schichten einteilen, weswegen sich auch bzgl. der Interoperabilität verschiedene Schichten identifizieren lassen. Dies ist im Wesentlichen eine Weiterentwicklung des OSI-Ansatzes (s. Abschnitt 5.1), in dem Sinne, dass er um ein klares Kriterium für die Identifikation der Schichten ergänzt wird, das auf der Strukturierung der Verarbeitung der Informationen in den Systemen basiert.

Damit wird es möglich, alle Interaktionen zwischen Systemen in eine überschaubare Anzahl von Klassen einzuteilen und in einem weiteren Schritt die breite Palette an unterschiedlichen Technologien zu betrachten und zu bewerten für welche Aspekte von Interoperabilität einzelne Technologien hilfreich sind.

1 z.B. OASIS: Reference model for service oriented architecture 1.0. ↗**https://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf** (2006)

2 ↗**https://www.plattform-i40.de/SiteGlobals/PI40/Forms/Listen/Glossar/DE/Glossar_Formular.html?resourceId=1022602&input_=1020846&pageLocale=de&titlePrefix=Alle#form-1022602**, Quelle: Industrie 4.0 – Technical Assets: Grundlegende Begriffe, Konzepte, Lebenszyklen und Verwaltung, VDI Statusreport Industrie 4.0 (November 2015)

3.1 Interaktionsmodell

Das Interaktionsmodell fußt auf der Vorstellung, dass kommunizierende Systeme Informationen verarbeiten und diese im Rahmen von Interaktionen austauschen. Dabei entspricht die Verarbeitung der Informationen einer Transformation von Eingabewerten, ggfs. in Verbindung mit Werten innerer Zustände auf Ausgabewerte.

Damit liegt es nahe, die Interaktion zwischen Systemen entlang dieser beiden Dimensionen Informationstransport und -Verarbeitung zu klassifizieren. Bei der Informationsverarbeitung beschränken wir uns auf diejenigen Eigenschaften, von denen wir wissen, dass sie einen unmittelbaren Einfluss auf die geeignete Form der Interfaces haben. Beispielsweise gibt es im asynchronen Fall keine unmittelbaren Rückgabeparameter; im deterministischen Fall ist es möglich, die intendierte Abbildung durch eine Operation zu beschreiben, die Zustandswerte auf Zustandswerte abbildet; bei Zustandsbehaftung und Determinismus lässt sich die intendierte Funktionalität objektorientiert beschreiben. Und zustandsbehaftete, nicht-deterministische und asynchrone Interaktionen werden durch Protokolle beschrieben.

Zusammengefasst klassifizieren wir die Interaktionen von Systemen daher vollständig wie folgt[3]:

1. **Informationsfluss**, in der Ausprägung unidirektional versus bidirektional; und
2. **Informationsverarbeitung** mit den drei Unterdimensionen Zustand, Determinismus und Synchronität mit den jeweiligen Ausprägungen zustandsbehaftet versus zustandslos, deterministisch versus nichtdeterministisch und synchron versus asynchron.

Eine Komponente verhält sich in einer Interaktion deterministisch, wenn ihre Zustandsübergänge vollständig durch die Eingaben bestimmt werden. Eine Komponente verhält sich zustandsbehaftet, wenn ihre Zustandsübergänge zusätzlich zur Eingabe auch noch von einem inneren Zustand abhängen. Eine Komponente verhält sich synchron, wenn sie in einer Interaktion als sendende Komponente ihre Weiterverarbeitung von dem Empfang einer Antwort auf eine gesendete Nachricht abhängig macht.

Der bidirektionale Informationsfluss ist die deutlich komplexere aber für die domänenübergreifende Umsetzung von Industrie 4.0 notwendige Variante. Sie erlaubt eine Einteilung in horizontale (symmetrische) gegenüber vertikale (asymmetrischen) Interaktionen und legt damit den Grundstein für eine klare Definition des Konzepts der Softwareschicht. Auf diesen Punkt wird im Folgenden detailliert eingegangen.

Mit der Unterscheidung in deterministisches und nichtdeterministisches Verhalten macht das Referenzmodell die weitreichende Festlegung, dass zur Definition eines Schnittstellenmodells immer auch eine Feststellung zum Transformationsverhalten der Systeme dazugehört.

3 Siehe auch J.Reich (2015), **↗Eine semantische Klassifikation von Systeminteraktionen**. D. Cunningham, P. Hofstedt, K. Meer, I. Schmitt (Hrsg.), Lecture Notes in Informatics (LNI), pp1545-1559. Sowie J. Reich, T. Schröder (2017), **↗A reference model for interaction semantics**, arXiv:1801.04185, v2

3.1.1 Interaktionen mit unidirektionalem Informationsfluss

Bei unidirektionalem Informationsfluss gibt es keine Synchronität bzw. Asynchronität, da per definitionem keine Informationen zurückgesendet werden.

Bezüglich der Dimension Determinismus lässt sich die folgende Unterscheidung treffen:

- Deterministisch: Wir sprechen auch von »Pipes«. In einer Pipe werden in sukzessiver Folge verschiedene Operationen auf einem »Datenstrom« ausgeführt. D.h. die Eingabe jeder Pipekomponente ist die Ausgabe ihrer Vorgängerkomponente (bis auf die erste in der Folge).

- Nichtdeterministisch: Der wichtigste Fall ist die »Anonyme Observation«, z.B. realisierbar durch Publish-Subscribe-Technologien. Bei der anonymen Observation wird der Zustandsübergang einer Komponente in dem Sinne anderen Komponenten zugänglich gemacht, dass diese Beobachtung für die Korrektheit der Verarbeitungsfunktion der beobachteten Komponente keine Rolle spielt. Insbesondere »kennt« die beobachtete Komponente damit nicht den Beobachter und macht auch keine Annahmen über seinen Zustand oder seine Verarbeitung.

3.1.2 Interaktionen mit bidirektionalem Informationsfluss

Der Bezug auf die gegenseitige Verarbeitung der Informationen in den beteiligten Systemen führt dazu, dass Interaktionen mit bidirektionalem Informationsfluss nicht einfach die Überlagerung zweier Interaktionen mit unidirektionalem Informationsfluss sind.

Anstatt sich auf die Flussrichtung der Informationen zu beziehen, lässt sich bei bidirektionalem Informationsfluss eine Richtung der Interaktion auf Grund von symmetrischer bzw. asymmetrischer Verarbeitung bestimmen. Dies ist dann tatsächlich eine »semantische Richtung« der Interaktions-Beziehung der beteiligten Systeme in unserem Sinne.

Dazu betrachten wir die Ausprägung der drei Verarbeitungsdimensionen Synchronität, Zustandsbehaftung und Determinismus aller beteiligten Komponenten. Ist die Ausprägung aller beteiligter Komponenten gleich, liegt – bezogen auf diese 3 Dimensionen – ein symmetrisches Verhalten vor, ist die Ausprägung unterschiedlich, liegt ein asymmetrisches Verhalten vor, dem man eine Richtung zuschreiben kann.

Tatsächlich sind nur ganz wenige Kombinationen sinnvoll. Das Auftreten der Kombination aus Nichtdeterminismus und Zustandslosigkeit bei einem Interaktionspartner bedeutet für andere Zufälligkeit und kann daher ausgeschlossen werden. Wechselseitiger Determinismus ist ebenfalls auszuschließen, da er einen Deadlock zur Folge hat: Spontanes Agieren ist ausgeschlossen, es erfolgt auch keine initiale Eingabe. So führt das wechselseitige Aufrufen von Operationen zu rekursiver Funktionalität, bei der der initiale Aufruf eine »äußere« Eingabe erfordert. Wechselseitiges synchrones Agieren aller Beteiligten einer Interaktion macht nur im Bereich rekursiver Funktionalität Sinn. Daher ergeben sich nur zwei wesentliche Kategorien:

- **Horizontale Interaktionen**: Hier verhalten sich die Interaktionspartner bezogen auf die 3 Unterdimension symmetrisch – nämlich zustandsbehaftet, nicht-deterministisch und asynchron. Diese Interaktionen werden durch Protokolle beschrieben, wobei die Interfaces der Protokollteilnehmer (die wir auch »Rollen« nennen) multi- oder bilateral sind in dem Sinne, dass die Kenntnis aller Rollen der Interaktionspartner notwendig ist, um wichtige Eigenschaften der Interaktion, in diesem Fall eines Protokolls, wie etwa Vollständigkeit, Konsistenz, die Freiheit von Deadlocks, Livelocks oder Starvation, zu bestimmen.

- **Vertikale Interaktionen**: Hier verhalten sich die Interaktionspartner bezogen auf die 3 semantischen Unterdimensionen asymmetrisch und erzeugen auf diese Weise eine semantische (d.h. auf die Informationsverarbeitung bezogene) Richtung der Interaktion. Diese Interaktionen werden durch unilaterale Interfaces beschrieben. Per definitionem »abwärts«-gerichtet verhält sich die »untere« Komponente deterministisch, weswegen sich ihr Verhalten durch Funktionsaufrufe (mit Ausnahmen) beschreiben lässt. Die deterministisch arbeitende Komponente macht ihrerseits keine Annahmen über die »übergeordnete« Komponente, weswegen sie nach »oben« nur über »aufwärts«-gerichtete Events kommunizieren kann. Tatsächlich wird im Interface seinem unilateralen Charakter folgend nur die Komponente beschrieben, die die Funktionalität, bzw. die Events zur Verfügung stellt. Isoliert betrachtet ähneln die aufwärtsgerichteten Events den anonymen Observationen.

3.1.3 Der Schichtenbegriff

Mit Hilfe dieser einfachen Klassifikation lässt sich für den Fall des bidirektionalen Informationsflusses der Begriff der Systemschicht definieren, der im Softwareengineering eine herausragende Bedeutung hat: Komponenten höhere Schichten interagieren mit Komponenten niedrigerer Schichten nur über vertikale Interaktionen und mit Komponenten aus ihrer eigenen Schicht nur über horizontale Interaktionen. Anonym observierte Komponenten lassen sich tieferen Schichten zuordnen, alle Komponenten einer Pipe sind derselben Schicht zuzuordnen.

Allerdings muss man sorgfältig beachten, welche Systeme man betrachtet und welcher Art die hierarchische Beziehung ist. Denn durch die deterministische Interaktion wird im vertikalen Interaktionsfall auch eine Supersystem-Subsystem-Beziehung etabliert, die ebenfalls eine Ordnungsrelation entstehen lässt. Diese Ordnungsrelation ordnet jedoch nicht die interagierenden Systeme, sondern die Super- gegenüber den Subsystemen. Ein Beispiel[4] soll diese scheinbar geringfügigen, aber dennoch sehr wichtigen Unterschiede verdeutlichen.

Abbildung 1 zeigt eine einfache Systemkomposition dreier Systeme S_1, S_2 und S_3, die zu einem Supersystem S mit der einfachen Funktion $f_S = 2x + 5$ komponieren. System S_2 trägt seine Systemfunktion $f_2 = 2x$ bei, System S_3 seine Systemfunktion $f_3 = x + 5$, und S_1 ist ein System mit mehreren Ein- und Ausgängen, das im Wesentlichen die Systeme S_2 und S_3 auf eine nicht-triviale, rekursive Weise koordiniert. Wie wir sehen können, gibt es keine Interaktion zwischen dem Supersystem S

4 Das Beispiel entstammt dem Artikel J. Reich, T. Schröder (2017), ↗**A reference model for interaction semantics**, arXiv:1801.04185, v2

und seinen drei Subsystemen S_1, S_2 und S_3. Stattdessen entsteht das Supersystem S durch die deterministische Interaktion zwischen S_1 und S_2 einerseits und S_1 und S_3 andererseits indem eine übergeordnete Systemfunktion entsteht, die dann dem Supersystem zuzuordnen ist.

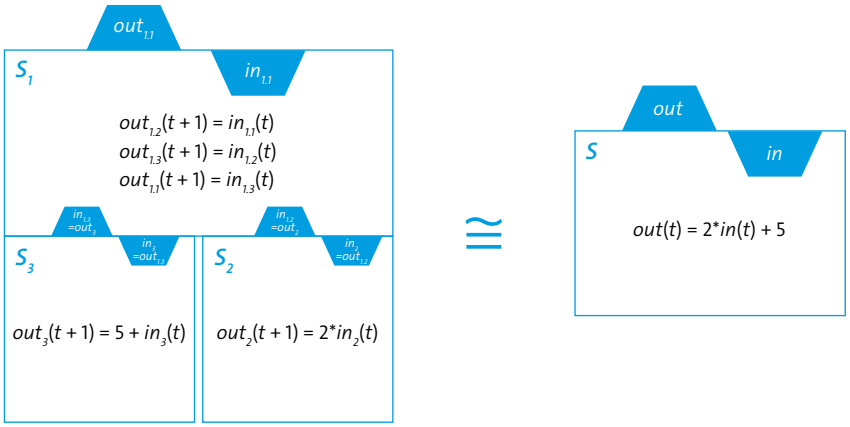

Abbildung 1: Drei Systeme S_1, S_2 und S_3 komponieren zu einem Supersystem S mit der Funktion $f_S = 2x + 5$. Jedes System wird als Box dargestellt, das über Eingangs- und Ausgangszustände *out*, *in* sowie einen inneren Zustand q verfügt, die über ihre Systemfunktion $f = (f^{ext}, f^{int})$ zu jedem Zeitschritt wie folgt abgebildet werden:

$$\begin{pmatrix} out\,(t+1) \\ q\,(t+1) \end{pmatrix} = \begin{pmatrix} f^{ext}\,(in(t), q(t)) \\ f^{int}\,(in(t), q(t)) \end{pmatrix}$$

Wie im linken Teil der Abbildung 2 dargestellt, können wir also gemäß der Interaktionsklassifikation das System S_1 einer höheren Schicht zuordnen als die beiden Systeme S_2 und S_3. Das Supersystem S taucht in dieser Beschreibung nicht auf und würde in dieser Darstellung alle Schichten seiner Subsysteme umfassen. Es ist daher nur ausgegraut dargestellt.

In der Regel, so auch im OSI-Modell, wird bei hierarchischer Darstellung jedoch die Supersystem-Subsystem-Beziehung dargestellt im Sinne einer »Ist-Teil-von«-Beziehung. Dies wird im rechten Teil von Abbildung 2 gezeigt. Jetzt ist das Supersystem S das übergeordnete System und die Subsysteme S_1, S_2 und S_3 findet sich im Sinne der »Ist-Teil-von«-Beziehung entsprechend in einer untergeordneten Schicht. In der Regel findet das Subsystem S_1 in dieser Darstellung keine Erwähnung. Es ist daher nur ausgegraut dargestellt.

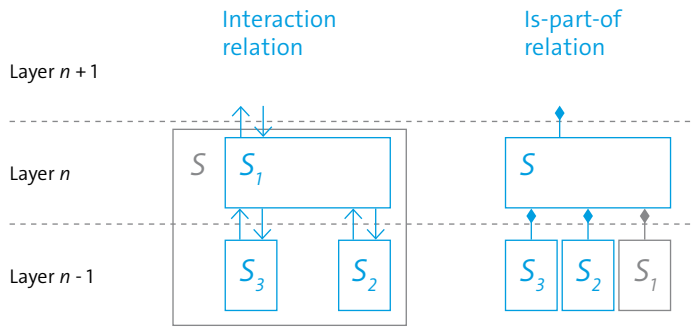

Abbildung 2: Auf Grund ihrer Interaktion können die Systeme von Abbildung 1 auf zwei verschiedene Arten geordnet werden. Auf der linken Seite sind sie bezogen auf ihre Interaktionsklasse geordnet. Die Pfeile repräsentieren den bidirektionalen Informationsfluss. Das Supersystem *S* wird nur ausgegraut dargestellt. Es umfasst in dieser Darstellung alle Layer seiner Subsysteme. Auf der rechten Seite ist die andere mögliche Ordnung der beteiligten Systeme durch ausgefüllte Rauten dargestellt, nämlich die »Ist-Teil-von«-Beziehung. Nun ist das Supersystem *S* übergeordnet und die Subsysteme S_1, S_2 und S_3 sind untergeordnet. Im Allgemeinen wird in dieser Darstellung das System S_1 nicht dargestellt, weshalb es nur ausgegraut eingezeichnet ist. Es gibt keinen Informationsfluss zwischen dem Supersystem und seinen Subsystemen.

Diese »Ist-Teil-von«-Beziehung wird in imperativen Programmen und in der Objekt-orientierten Welt mit ihrem Methoden-Konstrukt verwendet. Eine Methode repräsentiert eine Funktion, die sich – so sie denn nicht elementar ist – wiederum aus Teilfunktionen zusammensetzt, von denen sie entsprechend in diesem Sinne abhängt.

Der formale Nachweis für die Behauptung, dass eine Komponente in eine bestimmte Schicht der »Ist-Teil-von«-Relation einzuordnen ist, ist daher ein unilaterales Interface mit generischen Events und Operationen der Komponente zusammen mit dem Nachweis, dass die Komponente ihrerseits nur Operationen von Komponenten tieferer Schichten aufruft und auf generische Events tieferer Schichten reagiert.

Innerhalb der »Ist-Teil-von«-Beziehung nimmt die Abstraktion der Informationsverarbeitung mit zunehmendem Level »nach oben« hin zu.

Dieses Nebeneinander von Supersystem/Subsystem Hierarchie, die auf der »ist-Teil-von«-Beziehung beruht, und der horizontalen Interaktion zwischen Komponenten gleicher Schicht, ist in Abbildung 3 dargestellt. Es ist wichtig zu verstehen, dass in dieser Darstellung vertikal kein Informationsfluss stattfindet, sondern die Systeme (vertikal) tieferer Schichten in den Systemen der übergeordneten Schicht immer schon enthalten sind.

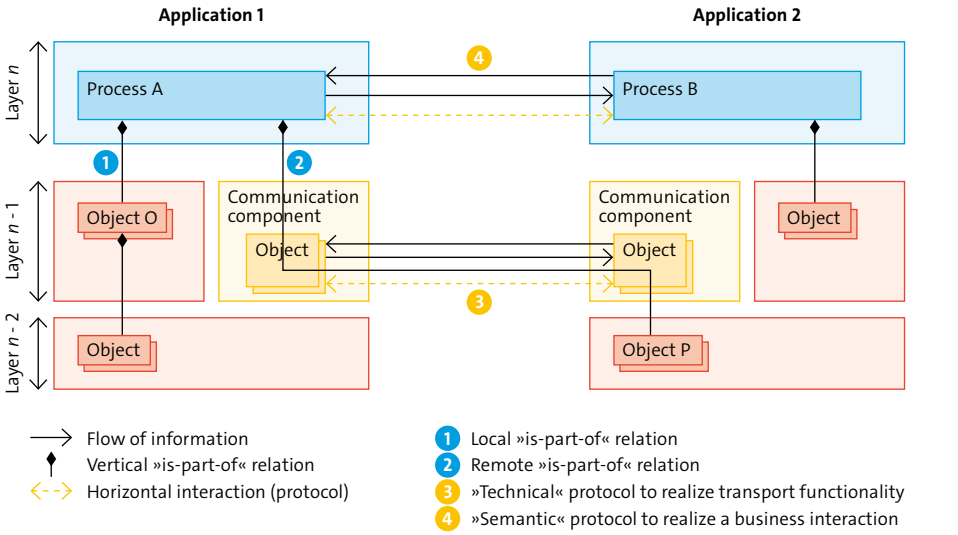

Abbildung 3: Darstellung einer geschichteten Applikationsarchitektur. Einerseits entsteht die Abhängigkeit der Objekte durch ihre Operationen im Sinne einer »ist-Teil-von«-Beziehung zwischen Supersystem und Subsystem. Und andererseits findet die Interaktion zwischen Objekten auf ein und derselben semantischen Ebene mittels Protokolle, d.h. nichtdeterministischen, zustandsbehafteten und asynchronen Interaktionen, statt, die nicht zu einer solchen »ist-Teil-von«-Beziehung führt.

3.1.4 Vertikale Interoperabilität

Für die Abstimmung der Informationsverarbeitung bei vertikal miteinander interagierenden Komponenten sind die beiden Richtungen abwärts vs. aufwärts zu unterscheiden. Dabei ist zu beachten, dass die Richtung nicht durch den Informationsfluss, sondern durch die Struktur der Verarbeitung vorgegeben wird.

Abwärts gerichtete Interaktion: Der Funktionsaufruf

Bei der abwärts gerichteten Interaktion wird ein deterministischer Verarbeitungskontext für die bereitgestellten Daten angenommen. Das ist der Grund, warum in imperativen Programmiersprachen sich diese Interaktion als Operationsaufruf beschreiben lässt.

Die aufgerufenen Operationen werden in der Regel mit Eigennamen benannt, etwa »Sinus()« oder mit Verben bezeichnet, etwa »createObject()«, was ihren Abbildungscharakter unterstreicht.

Das bleibt auch für den sogenannten »Remote«-Operationsaufruf gültig. Für diesen sind die erforderlichen Schritte von aufrufender und aufgerufener Komponente schematisch in Abbildung 4 dargestellt. Im Fall des Remote-Operationsaufrufs lassen sich die technisch notwendigen Kommunikationsschritte einer gemeinsamen Kommunikationsschicht zuordnen. Ist die Kommunikation unzuverlässig, entstehen in dieser Schicht zusätzlichen Ausnahmesituationen, die sogenannten »remote exceptions«. Je nachdem wie groß die dadurch eingeführte zusätzliche

Unzuverlässigkeit ist, kann die im Remote-Fall fehlende transaktionale Kontrolle[5] konsistente komplexe Zustandsänderungen praktisch unverhältnismäßig aufwändig machen. Zustandsändernde entfernte Funktionsaufrufe sind daher im industriellen Kontext mit einer gewissen Vorsicht zu betrachten.

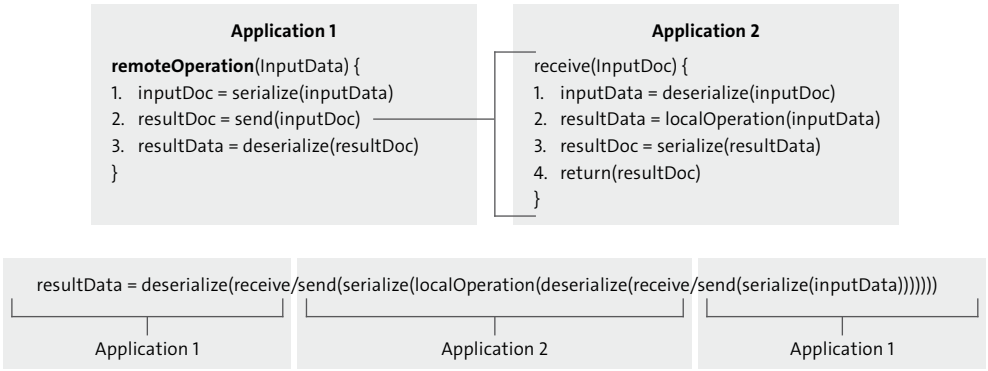

Abbildung 4: Ein entfernter Operationsaufruf verpackt die Serialisierung/Deserialisierung der transportierten Daten sowie den Datentransport und seine lokale Verarbeitung in einer Reihe von hintereinander ausgeführten Operationen. Diese werden einmal unter Verwendung von lokalen Variablen (oben) und ohne lokale Variablen (unten) dargestellt.

Grundsätzlich gilt für vertikale Interaktionen auch, dass die übergeordnete Komponente auch für die Ausnahmebehandlung der untergeordneten Komponente zuständig ist. D.h. will eine Komponente ein Filesystem auf einer Festplatte verwenden, um dort Daten zu persistieren, und ist die Platte aber voll, erhält sie von der Write-Operation eine Ausnahme signalisiert und muss nun selbst entscheiden, wie sie damit umgeht.

Aufwärtsgerichtete Interaktion: der generische Event

Wird ein Objekt durch mehrere übergeordnete Verwender bearbeitet, ist es häufig sinnvoll, die Änderung der betrachteten Daten auf Grund der Verwendung durch einen Verwender allen weiteren Verwendern anzuzeigen. Die Zuordnung kann inhaltlich anhand von Datenwerten oder mittels Registrierung erfolgen. Die »beobachteten« Daten oder der »beobachtete« Verarbeitungskontext im Sinne einer Objektorientierung enthalten selbst keine Informationen über diese Zuordnung. Der ausgelöste Event kann daher nur entlang eines allgemeingültigen Schemas, eben generisch erfolgen, etwa die Instanz 0815 des Objekts vom Typs XY ist vom (Haupt-)Zustand a in den (Haupt-)Zustand b gewechselt. In der Regel sind die Zustandsübergänge von besonderem Interesse, die das Verhalten eines Objekts grundsätzlich ändern, was in der Objektorientierung durch das »State-Pattern«[6] ausgedrückt werden kann.

5 Exemplarisch im bekannten Problem der »byzantinischen Generäle« aufgezeigt.

6 E. Gamma, R. Helm, R. E. Johnson und J. Vlissides (1994) Design Patterns. Elements of Reusable Object-Oriented Software. Prentice Hall.

3.1.5 Horizontale Interoperabilität

Systeme, die mit anderen Systemen jeweils symmetrisch interagieren, also im Wesentlichen nichtdeterministisch, asynchron und zustandsbehaftet, nennen wir »Prozesse«. Diese Interaktionen werden durch Protokolle beschrieben. Dies ist die Regel für die Interaktion zwischen komplexen Systemen unterschiedlicher Domänen, die außerdem noch jeweils mit der Umwelt oder dem Menschen interagieren oder auch eigene Entscheidungen treffen können.

Protokolle sind eine Menge von Regeln, die in Bezug auf die Zustände der beteiligten Systeme beschreiben, welche Zustandsübergänge unter welchen Umständen erlaubt bzw. verboten sind und wie diese Zustandsübergänge von den »sendenden« Akteuren dargestellt bzw. »dokumentiert« werden müssen, damit sie von den »empfangenden« Akteuren verarbeitet werden können. Daher bietet sich der Begriff »Dokument« für diese auszutauschenden Informationen im Kontext eines Protokolls an. In der Regel werden die zugehörigen Dokumentenklassen mit Hauptwörtern bezeichnet, wie etwa Auftrag, Rechnung oder Bestellung, da ihnen der imperative Charakter eines Operationsaufrufs fehlt.

Protokolle sind in dem Sinne abgeschlossen, als alle auszutauschenden Dokumente aufgeführt werden müssen und alle eventuell vorkommenden Kombinationen von Zuständen aller Beteiligter im Sinne des Protokolls zulässig sein müssen. Entsprechend komplex kann die Validierung der Konsistenz und auch der Korrektheit eines Protokolls werden.

Anschaulich entspricht ein Protokoll einem Spiel, ohne dessen Entscheidungen aufzuführen, die beim Spiel den Verlauf bestimmen oder eine Bewertungsfunktion anzugeben. Wie in Spielen lässt sich der Teil eines Protokolls, der einem einzelnen Teilnehmer zuzuordnen ist, als Rolle betrachten. Der Nichtdeterminismus in der Interaktion der Protokolle ist die Voraussetzung für den echten Entscheidungsspielraum, den komplexe Systeme in komplexen Umgebungen brauchen. Etwa ein Manufacturing Execution System in der Kontrolle komplexer Fertigungsprozesse oder ein Warehouse Management System in der Kontrolle komplexer Warenbereitstellungsprozesse oder ein Pflegeroboter in der Interaktion mit der Umwelt, den Menschen und seiner Servicestation.

Je nachdem in welcher Schicht der »ist-Teil-von«-Hierarchie sich diese Interaktion abspielt, ist ihr Inhalt mehr oder weniger »technisch«. Wir sprechen von »semantischen« Protokollen oder auch »semantischer« Interoperabilität, um zu betonen, dass es sich um eine Interaktion auf einer eher höheren semantischen Ebene handelt – etwa um eine Banküberweisung a la SEPA, oder um eine Patientenaufnahme a la HL7, oder einer Interaktion zwischen einem Staubsauger und einer Home-Automation-Anlage etc. Grundsätzlich gibt es aber keine »höchste« Schicht, weil sich – anschaulich und ohne Wertung gesprochen – alles Funktionalisieren lässt.

Die Interaktion zwischen Prozessen mittels Protokollen führt auch dazu, dass jeder Prozess die volle Verantwortung für »seine« Funktionalität trägt. D.h. tritt in einer von ihm aufgerufenen Funktion eine Ausnahme auf, ist er für ihre Behandlung zuständig. Entsprechend kommt es nur

bezogen auf die Sende-/Empfangsfunktionalität zu einer Kompetenzüberschneidung zwischen den beteiligten Systemen – was das gewünschte Designziel dieser Form von Interaktion ist.

3.2 Beschreibung von Protokollen

Da die Beschreibung von Interfaces als Protokoll häufig weniger gut geläufig ist, als die traditionellen funktionalen Interfaces eines Methodenaufrufs, gehen wir an dieser Stelle kurz darauf ein. Es gibt verschiedene Möglichkeiten, Protokolle darzustellen[7]. Auf Grund der Möglichkeit sehr subtiler Fehler, die die Welt der Nebenläufigkeit mit sich bringt, ist es wichtig, Protokolle so genau zu beschreiben, dass sie grundsätzlich auch einer formalen Validierung, etwa durch Model-checking zugänglich werden. Wir folgen an dieser Stelle dem Modell von J. Reich[8], das gegenüber anderen Modellen den Vorteil hat, das Protokoll-Interface mit dem Abbildungsverhalten der beteiligten Komponenten zu identifiziert und daher gut zu unserem Referenzmodell passt. In diesem Modell wird ein Protokoll durch das Abbildungsverhalten aller beteiligter Komponenten, die wir auch Rollen nennen, beschrieben. Jede Rolle wird dabei durch eine Menge von Transitionsklassen definiert, die jeweils einer Regel entspricht, die sich durch 5 Einträge bestimmt. Dazu wird der Zustand einer Komponente in einen Hauptanteil – oder Modus – und in einen Restanteil zerlegt. Die 5 Einträge sind dann wie folgt definiert:

- Der Modus des Startzustandes, z.B. Initial, Prepare, Execute and Final
- Der Modus des Zielzustandes.
- Die Klasse der eingehenden Dokumente, Z.B. Auftrag, Rechnung, Bestellung, etc. Ggfs. ergänzt um eine Entscheidung, die eine anderweitig unbestimmte Transition bestimmt. Wir bezeichnen die Entscheidungen mit einem '@' als ersten Buchstaben. Z.B. @CreatePlan wäre ein sprechender Name für eine Entscheidung, einen Plan zu erstellen.
- Die Klasse der ausgehenden Dokumente
- Eine Bedingung, die von der Rest-Komponente des Startzustands und den Parameterwerten des eingehenden Dokumentes abhängt.

D.h. jede Rolle kann als Tabelle der folgenden Form dargestellt werden, die wir auch im Abschnitt 6.2 im Beispiel der Beschreibung der Zusammenarbeit zwischen einem Enterprise Ressource Planning (ERP) und einem Manufacturing Execution System verwenden:

Wert des Modus des Startzustands	Dokumentenklasse des eingehenden Dokuments *Ggfs. mit Entscheidung*	Bedingung	Wert des Modus des Zielzustands	Dokumentenklasse des ausgehenden Dokuments

7 Siehe etwa G. J. Holzmann (1990), Design And Validation Of Computer Protocols. Prentice Hall. Oder: C. Baier und J. P. Katoen (2008) Principles of Model Checking. MIT Press

8 J. Reich (2020) ↗**Composition, Cooperation, and Coordination of Computational Systems.** arXiv:1602.07065

Jedes Protokoll hat weiterhin einen oder mehrere Initialzustände und ein Erfolgskriterium. Ein Protokoll als Menge aller beteiligten Rollen muss über die folgenden Eigenschaften verfügen:

- Vollständigkeit: Es gibt keine weiteren externen Eingabekanäle mehr.
- Abgeschlossen: Jedes innerhalb des Protokolls gesendetes Dokument muss vom Empfänger verarbeitet werden können.
- Konsistenz: Das Protokoll enthält keine Deadlocks, Livelocks oder Starvations.

4 Weitere relevante Aspekte der Informationsverarbeitung

4 Weitere relevante Aspekte der Informationsverarbeitung

Das Referenzmodell semantischer Interoperabilität legt nahe, sich bezüglich der Informationsverarbeitung nach weiteren hilfreichen Strukturen der Informationsverarbeitung umzuschauen, die bei der aufwandsarmen Herstellung von Interoperabilität wichtig sind.

In diesem Dokument verweisen wir dazu explizit auf drei weitere Konzepte: Zum einen die schon mehrfach angesprochene Verantwortlichkeit im Sinne der Zuständigkeit für Ausnahmebehandlungen, des Weiteren auf Sicherheitsmechanismen, mit denen die Auswahl der Systeme, die gesendete Informationen überhaupt verarbeiten können, erzwungen werden kann, sowie auf das Konzept der Datentypen.

4.1 Verantwortlichkeit

Verantwortlichkeit setzt voraus, dass klar ist, wer sich um Fehler und außergewöhnliche Umstände, sogenannter »Ausnahmen [engl. Exceptions]«, kümmern muss. Eine Ausnahme in diesem Sinne ist eine Störung einer deterministischen Informationsverarbeitung, die dazu führt, dass der übergeordnete Kontrollfluss geändert werden muss, dass also die Gesamtfunktionalität unter der Voraussetzung der eingetretenen Störung eine andere ist, als sie eigentlich angedacht war. Ein Beispiel ist eine Schreiboperation auf eine Festplatte, die in der Regel funktioniert, nur eben nicht, wenn die Platte schon vollgeschrieben ist. Das Referenzmodell semantischer Interoperabilität besagt, dass die Behandlung einer Ausnahme immer innerhalb eines Systems auf einer höheren Ebene geschieht als ihr Auftreten.

4.2 Sicherheit als Indikatoren guter Systemabgrenzungen

Sicherheit im Sinne der Schutzziele Vertraulichkeit, Integrität und Zurechenbarkeit setzt voraus, dass die Systemgrenzen zwischen dem Inneren eines Systems (»privat«) und dem Äußeren eines Systems (»öffentlich«) klar gezogen werden können.

Das wollen wir mit dem durch Signatur und Verschlüsselung auf Basis eines asymmetrisch kryptographischen Verfahrens abgesicherten Informationsaustausch zwischen zwei Systemen illustrieren. Das grundsätzlich gleiche Verhalten der beiden Akteure zeigt, dass dieses Verfahren sehr gut zur horizontalen Interaktion passt.

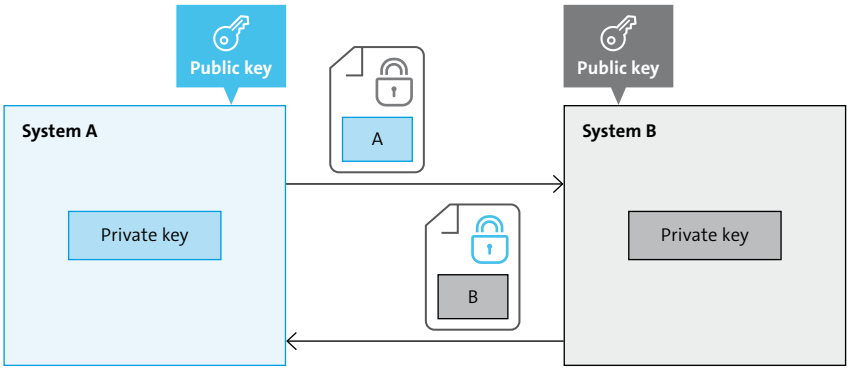

Abbildung 5: Darstellung des sicheren Informationstransports zwischen zwei Systemen mittels asymmetrischer Kryptographie. Der Sender unterschreibt die versendeten Daten mit seinem privaten Schlüssel und verschlüsselt sie mit dem öffentlichen Schlüssel des Empfängers.

Verschlüsselung: Stellt den Empfängerbezug her, als der Sender die serialisierten Daten mit dem öffentlichen Schlüssel des Empfängers verschlüsseln muss, damit ausschließlich der ausgewählte Empfänger die empfangenen Daten entsprechend mit seinem privaten Schlüssel entschlüsseln kann.

Signatur: Stellt den Senderbezug her, als der Sender zur Signatur einen »Fingerabdruck«, genauer einen Hash, der serialisierten Daten mit seinem privaten Schlüssel verschlüsselt, so dass der Empfänger diesen, von ihm mit dem öffentlichen Schlüssel des Senders entschlüsselten Hashwert, mit dem aus den empfangenen Daten selbst berechneten Hashwert vergleichen kann.

Da sich mit Sicherheitsmechanismen die Identifikation von Systemgrenzen in Interaktionen erzwingen lässt, können sie auch als empfindliche »Sonden« für die Konsistenz der angedachte Interaktionsarchitektur verstanden werden. In einer geschichteten Applikationsarchitektur lässt sich die Zuordnung vom System auf die jeweilige Schicht spezialisieren, die Kenntnis von den entsprechenden Schlüsseln, bzw. von den Operationen, die auf diesen Schlüsseln basieren, haben muss.

So stellen insbesondere die vertikale und die horizontale Interaktion unterschiedliche Anforderungen an die Sicherheitsmechanismen.

4.2.1 Sicherheitsmechanismen in horizontaler Interaktion

Die Sicherheitsmechanismen von Signatur und Verschlüsselung asymmetrischer Kryptografie können sehr einfach auf den Fall der symmetrischen Interaktion zwischen Prozessen übertragen werden, da sich im horizontalen, d.h. symmetrischen Fall alle Komponenten auch bzgl. ihrer Sicherheitsmechanismen symmetrisch verhalten können.

4.2.2 Sicherheitsmechanismen in vertikaler Interaktion

Bei der Ausführung von Funktionen muss v.a. geprüft werden, ob die entsprechende Autorisierung vorliegt. Im Remotefall heißt das, dass beide Applikationen dasselbe Identitätskonzept verfolgen müssen und die aufgerufene Applikation entscheiden muss, ob die Autorisierung gegeben ist. Verschlüsselung und Signatur können auf die Kommunikationsschicht beschränkt und damit in Bezug auf die übergeordnete Komponente der aufrufenden Schicht »technisch« bleiben.

Insbesondere besteht kein Bedarf der verwendeten Komponenten einer unteren Schicht, ihre Verwender (=Komponenten einer oberen Schicht) in dem Sinne zu kennen, dass sie sich in den Besitz deren öffentlicher Schlüssel bringen müssten.

4.3 Datentypen

Anschaulich sind Daten im Sinne dieser Darstellung Informationen von denen grundsätzlich bekannt ist, wie sie verarbeitet werden können, ohne im Detail jede mögliche Operation zu kennen. »Grundsätzlich bekannt« bedeutet konkret, dass im Rahmen eines Datentyp-Systems für jeden Datentyp zusätzlich zu einer Wertemenge (Alphabet), die die Informationen charakterisiert, eine Menge von Elementaroperationen vereinbart wird, die diese Werte verarbeiten können. Aus diesen Elementaroperationen lassen sich dann im Sinne des Berechenbarkeitskonzepts alle komplexeren Operationen zusammensetzen – die jedoch für die Definition des Datentyps nicht in einem konstruktiven Sinne bekannt sein müssen[9].

Gute Beispiele für solche Datentypen sind etwa der Datentyp Float, wie ihn die IEEE 754-Norm mit Alphabet und Elementaroperationen festlegt, oder auch die verschiedenen Charakter-Datentypen, wie sie vom Unicode Consortium definiert werden.

Datentypen spielen eine wichtige Rolle bei der aufwandsarmen Herstellung semantischer Interoperabilität. Bezieht sich die Beschreibung der Interaktion zweier Komponenten auf dasselbe Typsystem, dann basiert sie auf denselben Alphabeten als auch auf denselben Mengen an Elementaroperationen.

4.3.1 Die Bedeutung von Datentypen in vertikalen Interaktion

In vertikalen Interaktionen, die letztlich auf der Komposition berechenbarer Funktionalität beruht, lässt sich mit Datentypen garantieren, dass nur grundsätzlich geeignete Funktionen auf den dazu vorgesehenen Informationen operieren. Damit lassen sich entsprechende Laufzeitfehler schon zur Designzeit der Software vermeiden.

9 S. J. Reich (2017), ↗Data. arXiv:1801.04992

4.3.2 Die Bedeutung von Datentypen in horizontaler Interaktion

Die Semantik von Datentypen führt im Fall der horizontalen Interaktion zu einem gemeinsamen Vokabular. Einigen sich etwa zwei Akteure, Informationen als Temperaturwert anzusehen, erklären sie damit, dass beide dieselbe Menge von Elementaroperationen kennen, mit denen man Temperaturinformationen zu verarbeiten hat. Das legt offensichtlich die Bedeutung teilweise fest, in dem Sinne, dass es sich um einen Temperaturwert handelt, jedoch nicht weitergehend, ob etwa bei Überschreitung einer gewissen Schwelle bestimmte Maßnahmen zu ergreifen sind.

Diese weitergehende Semantik der Daten über ihre Typisierung hinaus ergibt sich aus dem Interaktionskontext wie er in Protokollen festgelegt wird. Da gegenseitig jedoch keine Funktionen aufgerufen werden, sondern nur wechselseitig Zustandsänderungen unter bekannten Regeln signalisiert werden, wird die Semantik der übermittelten Daten durch den jeweiligen Protokollkontext u.U. ebenfalls nicht vollständig bestimmt. So bestimmt sich bspw. die Semantik einer Versicherungsnummer eines Patienten in der Behandlungsinteraktion mit dem Krankenhaus auch durch die zusätzlichen Interaktionen, die die Krankenkasse sowohl mit dem Krankenhaus als auch mit dem Patienten hat. D.h. die Semantik der übertragenen Daten innerhalb eines Protokolls wird auf Grund der grundsätzlichen Offenheit der domänenübergreifenden Interaktionsnetzwerke ggfs. nur aspekthaft innerhalb eines Protokolls festgelegt und ist grundsätzlich nicht vollständig definierbar.

Dieser Netzwerk-Charakter der Semantik von Daten in einer horizontalen Interaktion erschwert einerseits die Herstellung von Interoperabilität, andererseits beschränkt er aber die Menge und v.a. Detailliertheit der zu vereinbarenden Bedeutungen für die horizontale Interaktion im Vergleich zur vertikalen Interaktion enorm. In letzterer müssen die übergeordneten Komponenten sogar mit den Ausnahmetatbeständen der untergeordneten Komponenten zurechtkommen.

5 Bezug zu anderen Referenzmodellen

5 Bezug zu anderen Referenzmodellen

5.1 Das Open System Interconnection (OSI) Modell

Das in diesem Dokument vorgestellte Modell lässt sich als Weiterentwicklung und Präzisierung des OSI-Modells[10] auffassen. Das OSI-Modell wollte nach eigenen Angaben nur den Austausch von Informationen zwischen offenen Systemen beschreiben und nicht die Funktion der Systeme[11]. Es besteht aus den bekannten 7 Schichten, die in Tabelle 1 aufgeführt werden. Das OSI Modell unterscheidet schon zwischen der horizontalen Interaktion zwischen den Schichten gleicher Ebene zweier Applikationen über Protokolle und der vertikalen Interaktion zwischen zwei benachbarten Schichten innerhalb einer Applikation durch sogenannte »services«.

Schicht	Beschreibung	Beispiele für realisierende Technologien
7. Anwendung (»application«)	Enthält die Anwendungsprozesse, die über Anwendungsprotokolle miteinander interagieren.	
6. Darstellung (»presentation«)	Macht die Applikation unabhängig von der konkreten syntaktischen Darstellung der Daten	
5. Sitzung (»session«)	Ermöglicht die Organisation des Datenaustausches (zustandslos/zustandsbehaftet).	
4. Transport (»transport«)	Stellt die Transportfunktionalität für die Daten zur Verfügung.	TCP, UDP
3. Vermittlung (»network«)	Erzeugt Unabhängigkeit von Routing und Relaying.	IP
2. Sicherung (»data link«)	Fehlerkorrektur für die physikalischen Schicht.	Ethernet, FDDI
1. Bitübertragung (»physical«)	Stellt mechanische, elektrische und funktionale Verbindung her.	

Tabelle 1: Die 7 Schichten des OSI-Modells

Mit Hilfe unseres Modells lässt sich verstehen, wo die Unzulänglichkeit des OSI-Modells liegt und warum entsprechend in der Praxis nur die Schichten 1-4 ihren Weg in die Realität gefunden.

Tatsächlich sind die Grundannahmen des OSI-Modells nicht ausreichend. Nämlich, dass man die Struktur einer Applikation bestimmen kann, ohne sich auf die Struktur ihrer Informationsverarbeitung zu beziehen. Ohne einen solchen Bezug fehlt dem OSI-Modell aber ein valides Kriterium für die Schichtung.

Konkret führt das OSI-Modell etwa eine Sitzungsschicht ein, deren Aufgabe es sein soll, den Zustand der Interaktion zu kontrollieren. Die Frage aber, ob die Interaktion zustandsbehaftet ist

10 ISO/IEC 7498-1 sowie ITU-T X.200 »Information Technology – Open Systems Interconnection – Basic Reference Model«, 1994.

11 So heißt es in Abschnitt 4: »OSI is concerned with the exchange of information between open systems (and not the internal functioning of each individual real open system).«

und welche Bedeutung dieser Zustandsbehaftung jeweils zukommt, lässt sich nicht in allgemeiner Form einer eigenen Schicht zuordnen, sondern nur im Falle der vertikalen Interaktion, wo die Kommunikationsschicht selbst zu einer Zwischenschicht wird. In einer horizontalen Interaktion ist die Zustandsbehaftung der Interaktion selbst zuzurechnen und nicht der Kommunikation. So ist es insbesondere möglich, dass in einer Interaktion ein Zustand wesentlich ist und die darauf aufsetzende, übergeordnete Interaktion wiederum zustandslos ist, usw. So ist etwa das TCP/IP-Protokoll zustandsbehaftet, das darauf aufsetzende HTTP-Protokoll realisiert eine zustandslose Transportfunktion und eine Geschäftsinteraktion, die beim Dokumententransport HTTP verwendet, ist wiederum zustandsbehaftet.

Weiterhin ist unsere Anmerkung aus Abschnitt 3.1.3 zu beachten, dass das OSI-Modell in vertikaler Richtung die »Ist-Teil-von«-Beziehung verwendet, die zwar durch Interaktion entsteht, selbst aber keine Interaktion repräsentiert.

5.2 Level of Conceptual Interoperability Model (LCIM)

Das Referenzmodell »Level of Conceptual Interoperability Model (LCIM)«[12] hat ebenfalls eine gewisse Verbreitung gefunden und besteht aus 7 Ebenen, die hier in den Worten des LCIM gegeben werden.

1. No IOP: Keine Interaktion
2. Technical IOP: Gestattet den Austausch auf Informationsebene (»*common understanding of signals*«)
3. Syntactic IOP: Legt Struktur der ausgetauschten Daten fest und ermöglich damit die Interpretierbarkeit der Daten. (»*common understanding of symbols*«)
4. Semantic IOP: Die Bedeutung der ausgetauschten Daten wird geteilt (»*common understanding of terms*«)
5. Pragmatic IOP: Die Systeme kennen die Handlungsrelevanz der ausgetauschten Informationen (»common understanding of the use of terms«)
6. Dynamic IOP: Die Systeme verstehen die Folgen der verursachten Zustandsänderungen (»*common understanding of effects*«)
7. Conceptual IOP: Die Systeme teilen die konzeptuellen Modelle, d.h. die Annahmen und Einschränkungen der Abstraktion der Realität (»common conceptual model«)

Dem Modell liegt kein explizites Kriterium für die Schichtenbildung zugrunde. Tatsächlich ist das Modell auf die Schichtung einer Applikation im Sinne des in diesem Artikel vorgeschlagenen Modells nicht abbildbar. So ist etwa, wie das IP-Protokoll zeigt, schon für den einfachen Informationsaustausch ein gewisses Routing erforderlich, was eine gewisse Struktur der übertragenen Information voraussetzt. Auch die Unterscheidung in Semantisch vs. Pragmatisch vs. Dynamisch ist unklar. Wie soll man eine Aktion einer Softwarekomponente beschreiben, ohne sich auf den

12 Tolk, Tunitsa, Diallo, Winters (2006), »Composable M&S WebservicesWeb-Services for Net-centric Applications«, JDMS, Volume 3, Issue 1, 2006, pp 27–44

Zustandsbegriff zu beziehen? Wie soll man etwa die Bedeutung einer Banküberweisung definieren, ohne auf ihren appellativen Charakter hinzuweisen, dass doch die Bank demnächst etwas tun soll, nämlich das Geld wie angewiesen zu überweisen. Und wie soll man eine Banküberweisung beschreiben, ohne der Bank eine Zustandsänderung zu attestieren? D.h. semantische, pragmatische oder dynamische Dinge im Sinne des LCIM beleuchten Aspekte der Semantik der ausgetauschten Informationen im Sinne des vorgestellten Modells, aber sie lassen sich nicht im Sinne einer Schichtung innerhalb von Applikationen separieren.

5.3 Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)

Das RAMI4.0[13] beschreibt nach eigener Darstellung strukturiert die wesentlichen Elemente eines für eine Organisation wertbehafteten Gegenstandes (Asset) mittels eines aus drei Dimensionen bestehenden Modells. Die drei Dimensionen sind:

1. Architektur (Layers) mit sechs Schichten zur Darstellung der für die Rolle des Assets relevanten Informationen (siehe Tabelle 2).

2. Verlauf (Life Cycle & Value Stream) zur Darstellung des Lebenslaufs eines Assets und des Wertschöpfungsprozesses in Anlehnung an IEC 62890.

3. Hierarchie (Hierarchy) zur Zuweisung funktionaler Modelle zu einzelnen Ebenen in Anlehnung an die Normen DIN EN 62264-1 und DIN EN 61512-1.

RAMI4.0 Schicht	Beschreibt ...
Business	... die geschäftliche Sicht.
Functional	... (logisch) Funktionen eines Assets (fachliche Funktionalität) im Hinblick auf seine Rolle im Industrie 4.0-System.
Information	... die von der fachlichen Funktionalität des Assets genutzten, erzeugten bzw. modifizierten Daten.
Communication	... den Industrie 4.0-konformen Zugriff auf Informationen und Funktionen eines vernetzten Assets von anderen Assets aus.
Integration	... die zur Realisierung der Funktion eines Assets vorhandene Infrastruktur (Ressource).
Asset	... die dingliche Realität, deren virtuelle Abbildung in den darüber liegenden Schichten erfolgt.

Tabelle 2: Die 6 sogenannten »Schichten« der Architektur-Dimension des RAMI4.0

13 ↗https://www.beuth.de/en/technical-rule/din-spec-91345/250940128

Das RAMI4.0 Modell bezieht sich in seinen Schichten nicht auf ein Komponentenmodell und gibt auch kein explizites Schichtenkriterium an. Damit sind die RAMI4.0-Schichten nicht auf die Schichten abbildbar, die durch die Interaktion von Komponenten entstehen, wie sie das vorgestellte Referenzmodell beschreibt. Das hat zur Folge, dass in Bezug auf das in diesem Artikel vorgestellte Referenzmodell die RAMI4.0 »Schichten« eher Aspekte der betrachteten Assets im Sinne einer Projektion darstellen.

5.4 Das IIC Connectivity Framework

Mit seinem Connectivity Framework[14] adressiert das IIC ebenfalls das Problem der sehr heterogenen Connectivity-Technologien. Die Grundidee zur Vereinfachung ist eine kleine Anzahl von »Core«-Standards zu definieren, die untereinander interoperabel sein sollen und alle anderen Connectivity-Technologien nur gegen einen dieser Core-Standards operieren zu lassen. Damit will man das n^2-Problem lösen, das entstünde, wenn nun jede Technologie mit jeder anderen Technologie interoperabel zu sein hätte.

Das zugrunde gelegte konzeptuelle Interoperabilitäts-Modell ist dem LCIM entlehnt und besteht in der Darstellung des IICs aus den drei Ebenen:

1. Technische Interoperabilität: Informationen im Sinne von Bits und Bytes können ausgetauscht werden.

2. Syntaktische Interoperabilität: Daten im Sinne von getypten, strukturierten Informationen können ausgetauscht werden.

3. Semantische Interoperabilität: Es ist möglich, die Daten eindeutig zu interpretieren.

Die betrachtete Connectivity soll lediglich »syntaktische Interoperabilität« ermöglichen im Sinne eines verteilten Zugriffs auf Daten im Sinne typisierter Informationen. Dazu wird eine Connectivity Referenzarchitektur mit zwei Kommunikationsschichten oberhalb der OSI-Schicht 3 definiert als

1. Transportschicht [vergleichbar der OSI-Schicht 4]: Stellt technische Interoperabilität her, d.h. den Austausch der Bits & Bytes. Die Funktionalität der Transportschicht wird durch das verwendete Nachrichtenprotokoll implementiert und umfasst Endpunkt-Adressierung, Kommunikationsmodi, Netzwerktopologie, Verbindungsmanagement, Priorisierung, Timing und Synchronisation, sowie Nachrichtensicherheit.

2. Framework-Schicht [spannt lt. Darstellung die OSI-Schichten 5-7 auf]: Stellt syntaktische Interoperabilität her, d.h. sorgt für den Austausch strukturierter Daten zwischen den »End-

14 ↗IIoT Volume G5: Connectivity Framework, IIC 2019

punkten«. Die Funktionalität des Datenaustauschs basiert u.a. auf einem Daten-Ressourcen-Modell (mit Datentyp-System, Id-& Adressmanagement, Lifecycle CRUD-Funktionalität zu den Daten-Objekte, Verwalten von Zustandshistorien), Publish-subscribe- und Request-Reply-Datenaustauschmustern, Ausnahmebehandlung, sowie Discovery-Funktionalität.

Die Eigenschaften der Framework-Schicht werden als Anforderungen an einen Core-Standard Connectivity betrachtet.

Das IIC-Dokument versucht ähnlich wie das vorliegende Dokument einen konzeptuellen Rahmen aufzustellen, mittels dessen sich verschiedene Kommunikationstechnologien hinsichtlich ihres Anwendungsnutzens bewerten lassen. Jedoch ist der verwendete Maßstab, nämlich die Unterscheidung zwischen technischer, syntaktischer und semantischer Interoperabilität nicht ohne weiteres geeignet, eine Schichtung in einer Applikation zu rechtfertigen und entsprechend Schichten-bezogene Interoperabilitätskriterien anzugeben. Stattdessen ist, zumindest im Sinne des Referenzmodells semantischer Interoperabilität, die Schichtung von Komponenten selbst »semantisch«, also sinnvoll nur über die Art und Weise ihrer Interaktion zu definieren.

Dadurch wird übersehen, dass Interoperabilität im Allgemeinen nicht durch den verteilten Zugriff auf Daten erreicht wird, sondern im Bild von Informationstransport und Verarbeitung eine adäquat angepasste Informationsverarbeitung aller Beteiligter erfordert. Sollen »Endpunkte« getypte Daten austauschen, müssen sie sich auch bzgl. des Datentyps der ausgetauschten Entität festlegen, also ob es ein Event darstellt, einen Funktionsaufruf, oder ein Geschäftsdokument im Rahmen eines Geschäftsprotokolls. Das sind alles schon sehr weitreichende »semantische« Festlegungen, also Festlegung, die sich auf die Verarbeitung der Informationen beziehen.

Im IIC-Dokument wird die Auffassung vertreten, dass Publish-subscribe auch für bidirektionale Interaktion eingesetzt werden kann. Das ist nach Auffassung des Referenzmodells semantischer Interoperabilität falsch. Tatsächlich basiert Publish-subscribe wesentlich darauf, dass der Sender den Empfänger nicht kennt, was im bidirektionalen Fall in der Regel nicht der Fall ist.

Die gravierendste Konsequenz ist die Auslassung des für netzwerkartige Interaktionen bestimmenden Falls, nämlich dass komplexe, (teil-)autonom agierende Komponenten asynchron, nichtdeterministisch und zustandsbehaftet interagieren und ihre Interaktion demnach als Protokolle darzustellen sind – und sie ihre Datenmodelle nur bezüglich ihrer Interaktion abzugleichen haben, nicht aber bezüglich ihres intern verwendeten Objektmodells im Sinne einer reuse-orientierten Kapselung der Daten mit Operationen.

6 Bezug zu sogenannten Design bzw. Architecture Styles

6 Bezug zu sogenannten Design bzw. Architecture Styles

In diesem Abschnitt gehen wir kurz auf die Service Orientierte Architektur (SOA), wie sich durch OASIS definiert wurde und den sogenannten REpresentational State Transfer (REST) ein. Beide werden in der Literatur als sogenannte (Software) »Architectural Styles« bzw. »Design Styles« bezeichnet.

6.1 Service Orientierte Architektur (SOA) / Web-Services

OASIS definiert eine SOA[15] recht allgemein als »paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains«. Ein Service wird definiert als »mechanism by which needs and capabilities are brought together«. WSDL 2.0[16] spricht in Abschnitt 2.2.1 von einer »Interface component« als einer »sequences of messages that a service sends and/or receives«. Nach WSDL 2.0 ist eine »operation« eine »interaction with the service consisting of a set of (ordinary and fault) messages exchanged between the service and the other parties involved in the interaction«.

Die Spezifikationen der Web-Services nehmen an keiner Stelle Bezug auf die Abbildungseigenschaften der Web-Services. Stattdessen wird ein »Web-Service« wenig differenzierend als etwas dargestellt, in das Informationen hineingehen und herauskommen. Tatsächlich sind die über WSDL zur Verfügung gestellten syntaktischen Mittel allerdings nur geeignet, Operationen zu beschreiben. Damit sind Web-Services im Sinne der WSDL-Spezifikationen nicht für die Darstellung horizontaler Interaktionen geeignet. Diese semantische Vagheit hat wesentlich zur Komplexität der Welt der WS-Spezifikationen beigetragen.

6.2 REST

Die Idee des REpresentational State Transfer (REST) basiert auf der Dissertation von Roy T. Fielding[17]. Ein REST-Service soll bestimmte Eigenschaften haben. Roy T. Fielding selbst nennt

- Adressierbarkeit: jede Ressource hat eine eindeutige URL, und
- Zustandslosigkeit: jede REST-Nachricht enthält alle Informationen, die zu ihrer Verarbeitung notwendig sind.

Manchmal wird auch genannt

- Idempotenz: identische Nachrichten entfalten an identischen Endpunkten identische Wirkungen.

15 ↗http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

16 ↗https://www.w3.org/TR/wsdl20/

17 R. Fielding, »Architectural styles and the design of network-based software architectures«, Ph.D. dissertation, University of California, Irvine, 2000

Dies entspricht den HTTP-Operationen von GET, HEAD, PUT und DELETE. Eine einheitliche Standardisierung besteht nicht.

REST kann als Versuch angesehen werden, die Prinzipien des im Internet so erfolgreichen HTTP-Transportprotokolls, zustandslose Kommunikation zusammen mit semantischer Agnosie auf die Beschreibung von Komponenteninteraktionen zu übertragen. Beides steht in direktem Widerspruch zu dem vorgestellten Referenzmodell Interoperabilität. Horizontale Interaktion ist per Definitionem zustandsbehaftet, die empfangenen Informationen werden in der Regel nicht idempotent verarbeitet und der Interaktionspartner erhält in der Regel keinen detaillierten Einblick oder gar direkten Zugriff auf die eigenen Ressourcen.

Aus der Sicht des vorgestellten Referenzmodells ist REST eine methodische Chimäre mit Bestandteilen aus der Objekt-orientierten-, als auch aus der Protokoll-Welt. Es wird ein Ressourcen-orientiertes Briefkastensystem definiert, bei der die Verfügungsgewalt über den Lebenszyklus ungünstiger Weise an den Sender delegiert wird, und es werden keinerlei Annahmen über die eigentliche Verarbeitung der »REST-Services« gemacht. Insbesondere lassen sich keine Zusammenhänge zwischen verschiedenen »REST-Services« formulieren. Etwa, dass eine Rechnung, bevor sie gestellt wird, eine Lieferung erfordert und anschließend auch bezahlt werden sollte.

Es ist zu erwarten, dass dieser »Architekturstil« zu erheblichen Problemen führen wird, wenn mit dieser Form von Interfaces im Bereich der horizontalen Interaktionen gearbeitet wird, weil sie keine Analyse hinsichtlich der vielen Fallstricke nebenläufiger Interaktionen wie etwa Konsistenz, Deadlocks, Livelocks, etc. zulässt.

7 Beispiele

7 Beispiele

Nach dem formulierten Referenzmodell semantischer Interoperabilität erfolgt die Domänenübergreifende Kommunikation horizontal. D.h. Applikationen, die ihre jeweilige Domäne kompetent repräsentieren, interagieren über eine Dokumenten-basierte Protokoll-Schnittstelle asynchron, zustandsbehaftet und nichtdeterministisch miteinander. Sie kümmern sich jeweils eigenständig um die Behandlung eventueller Ausnahmen, die bei ihrer inhaltlichen Verarbeitung der empfangenen Daten auftreten. Damit sind ihre Zuständigkeitsbereiche klar gegeneinander abgrenzbar, und ggfs. mittels kryptografischer Methoden auch durchsetzbar.

Um sich gegenseitig zu verstehen müssen sich alle beteiligten Parteien bezüglich der Interaktion auf ein gemeinsames Datenmodell, sowie die Protokollregeln verständigen. Da dieses Datenmodell nur die Interaktion betrifft, kann es sich von dem Datenmodell, das zur internen Repräsentation der zu verarbeitenden Informationen verwendet wird, unterscheiden.

Beide Datenmodelle werden in der Regel voneinander abweichen, da sie unterschiedliche Designziele verfolgen. Das Design des Datenmodells einer horizontalen Interaktion richtet sich nach den Anforderungen des Empfängers wie auch den Fähigkeiten des Senders: Der Sender muss seine Zustandsübergänge so repräsentieren, dass der Empfänger seinerseits die Zustandsübergänge ausführen kann, an der alle Beteiligte ein genuines Interesses haben.

Das Design des internen Datenmodells wird hingegen vom Applikationsdesigner autonom festgelegt und orientiert sich an der Frage, wie das Objektmodell am besten geschnitten werden kann, um einen möglichst hohen Wiederverwendungsgrad der verwendeten Funktionalität innerhalb einer Applikation sicherzustellen.

Zwei Beispiele sollen die Relevanz der Protokoll-basierten Interaktionen für die Domänenübergreifende Interaktion illustrieren. Im ersten Beispiel wird am Betrieb eines Elektro-Mähdreschers das komplexe Zusammenspiel vieler verschiedener Komponenten aufgezeigt. Im zweiten Beispiel beschreiben wir die Interaktion zwischen einem Enterprise Ressource Planning (ERP) System und einem Manufacturing Execution (ME) System bei der Auftragsfertigung im Detail, um den Protokollbegriff weiter zu veranschaulichen.

7.1 Beispiel Produktion und Betrieb eines Elektro-Mähdreschers

Zur Veranschaulichung seien exemplarisch die Produktion und der Betrieb eines Elektro-Mähdreschers genannt. Im Rahmen der Energiewende besteht eine Herausforderung in dem Ausgleich der Energieproduktion (bspw. Windkraft oder Photovoltaik) mit dem Energieverbrauch. Durch eine geeignete Vernetzung zwischen der Produktion und dem Energiemarkt (4) kann durch die Verschiebung von industriellen Lasten geeignete Flexibilitätsoption geschaffen werden. Die lokalen sogenannten Bilanzkreise sind wiederum untereinander selbst mit eigenen Protokollen vernetzt (5). Auch erfolgt eine Kommunikation zwischen dem Stromnetz mit dem Elektro-Mähdrescher während zukünftiger Ladevorgänge (6).

Um die Effizienz in der Landwirtschaft zu steigern, erfolgt darüber hinaus eine Kommunikation zwischen dem Mähdrescher und Komponenten in dem landwirtschaftlichen Betrieb (7). Zur hochpräzisen Flächenbearbeitung beispielsweise werden Standorte bestimmt und mit anderen Fahrzeugen geteilt und über eine zentrale Steuerung koordiniert (8). Durch optimierte Fahrrouten wird der Energieverbrauch gesenkt und durch einen weiteren Austausch mit dem Energiemarkt (9) können auch hier Energielasten flexibel aufgefangen werden, indem der Fuhrpark zu geeigneten Zeiten aufgeladen wird.

Diese Beispiele zeigen, dass ein Gegenstand den Anwendungsbereich innerhalb des Lebenszyklus einmal oder mehrfach wechseln kann. Demzufolge muss das intelligente Produkt mit externen Systemen unterschiedlicher Anwendungsbereiche kommunizieren. Wenngleich der Gegenstand – und eine etwaige digitale Repräsentation – stets dieselbe Entität ist, erfolgt die Kommunikation in diesen Bereichen wiederum ganz unterschiedlichen Paradigmen, Protokollen und Datenmodellen.

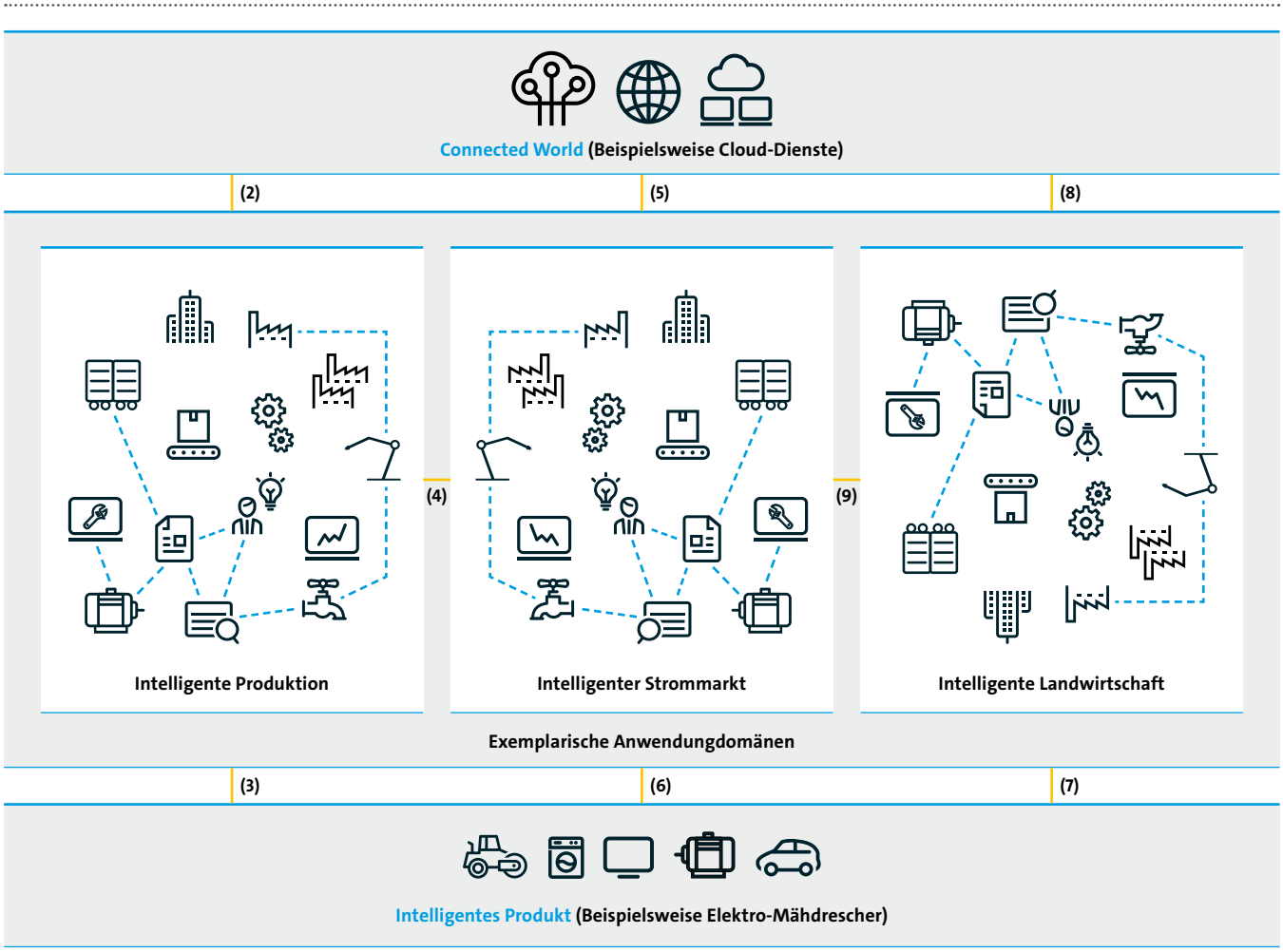

Abbildung 6: Schematische Darstellung der Beziehungen und Interaktionen im Rahmen einer Fertigung und des Betriebs eines Elektro-Mähdreschers

7.2 Beispiel der Zusammenarbeit von ERP und MES bei der Fertigung

Enterprise Ressource Planning-Systeme (ERP) und Manufacturing Execution Systeme (MES) unterstützen in ihrer jeweiligen Domäne komplexe Geschäftsprozesse. In diesem Beispiel beschreiben wir die Interaktion zwischen dem ERP und einem MES bei der Vergabe eines Fertigungsauftrags.

Die Aufgabe des ERP-Systems ist die bedarfsgerechte Steuerung und Koordinierung der betrieblichen Ressourcen wie Personal, Betriebsmittel, Kapital, etc. und damit für effiziente betriebliche Wertschöpfungsprozesse zu sorgen. Die Aufgabe des MES (auch ‚Produktionsleitsystem') ist die Steuerung und Koordinierung der Produktion in Echtzeit.

Um ihren Aufgaben gerecht zu werden, verfügt sowohl das MES als auch das ERP über eine Vielzahl von Verbindungen zu anderen Systemen. So interagieren MES und ERP wegen der Produktionsbeauftragung. Aber darüber hinaus interagiert etwa das MES mit den Systemen der Prozessautomatisierung, mit dem HR-System etwa wegen der Schichtplanung und weiteren Systemen, die unmittelbare Auswirkung auf den Produktionsprozess haben.

Es ist zu beachten, dass in der sogenannten Automatisierungspyramide das ERP zwar über dem MES steht. Aus Sicht des Referenzmodells semantischer Interoperabilität aber ERP und MES mittels einer horizontalen Interaktion interagieren, in der beide Systeme sich gleichartig, nämlich asynchron, nichtdeterministisch und zustandsbehaftet verhalten. D.h. wenn zwischen ERP und MES eine hierarchische Beziehung gesehen wird, stützt sich diese auf andere Kriterien, als diejenigen, die die Form des Interfaces zwischen beiden Systemen bestimmt.

Die folgende Interaktionsbeschreibung ist als eine Möglichkeit zu betrachten, wie die beiden Rollen des ERPs und des MES zusammen wirken könnten. Es sind natürlich auch andere denkbar.

7.2.1 Die ERP-Rolle

Die ERP-Rolle wird durch die folgenden Dokumente und Hauptzustände (Modi) bestimmt.

- Gesendete Dokumente: Fertigungsauftrag (Production Order Request) ProdOrdReq.
- Empfangene Dokumente: Arbeitsbestätigung (Confirmation of Work) ConfWork.
- Hauptzustände: Initial, ReleasedAndDistributed, PartlyConfirmed, Confirmed, Final.

Die Transitionen werden in Abbildung 6 grafisch illustriert und im Weiteren beschrieben.

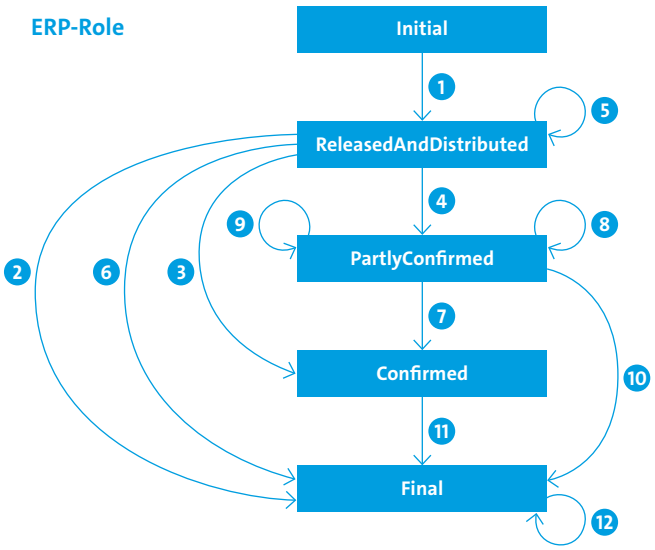

Abbildung 7: Zustandsdiagramm für den Hauptzustand der ERP-Rolle in der ERP-MES-Interaktion.

Initial: Das ERP initiiert die Interaktion (1), wenn der Produktionsauftrag intern freigegeben wurde und die Entscheidung getroffen wurde, dass das MES einzubeziehen ist. Diese Entscheidung des ERPs erfordert eine komplexe Bestimmung der Stückliste, des Arbeitsplans, der Kostenkalkulation und des Scheduling, der Materialbedarfsplanung, usw., die dem MES verborgen bleiben. U.U. impliziert ein Produktionsauftrag mehrere weitere Produktionsaufträge.

Dann wird ein Fertigungsauftrag (ProdOrdReq) an das MES gesendet und das ERP geht in den Zustand ReleasedAndDistributed über.

ReleasedAndDistributed: Abhängig von den Einstellungen sendet das MES Arbeitsbestätigungen (ConfWork) nach jedem Arbeitsschritt oder erst am Ende der Produktion an das ERP zurück. Empfängt das ERP eine solche Bestätigung, registriert sie diese und frischt seine Inventarliste entsprechend auf.

Ist der Produktionsprozess abgeschlossen, geht das ERP in seinen Zustand Confirmed (3) über, ansonsten in den Zustand PartlyConfirmed (4). Wurde der Fertigungsauftrag abgelehnt, geht es in seinen Zustand Final über (2).

Ändern sich in diesem Zustand die Produktionsanforderungen, dann wird dies dem MES mit einem neuen Fertigungsauftrag mitgeteilt (5).

Wird die Produktionsanforderung innerhalb des ERPs zurückgenommen, wird eine entsprechender Lösch-Fertigungsauftrag gesendet und das ERP geht in den Zustand Final über (6).

PartlyConfirmed: Wenigstens ein Fertigungsauftrag wurde versendet und bisher wurde keine finale Vollständigkeit des Auftrags vom MES vermeldet.

Wird eine Arbeitsbestätigung erhalten mit dem Vermerk »completed = full«, dann wird das Inventar aufgefrischt und das ERP geht in den Zustand Confirmed (7) über. Ohne diesen Vermerk verbleibt das ERP im Zustand PartlyConfirmed (8).

Ändern sich in diesem Zustand die Produktionsanforderungen, dann wird dies dem MES mit einem neuen Fertigungsauftrag mitgeteilt (9).

Wird die Produktionsanforderung innerhalb des ERPs zurückgenommen, wird eine entsprechender Lösch-Fertigungsauftrag gesendet und das ERP geht in den Zustand Final über (10).

Confirmed: Das MES hat dem ERP mitgeteilt, dass der Fertigungsauftrag abgearbeitet wurde. Es kann keine weitere Arbeitsbestätigung empfangen werden. Das ERP setzt seinen internen Produktionsauftrag auf »finished« und geht in den Zustand Final über (11).

Final: Die Interaktion zur Produktion ist nun beendet. U.U. trudeln noch ausstehende Arbeitsbestätigungen ein (12).

7.2.2 Die MES-Rolle

Die MES-Rolle wird durch die folgenden Dokumente und Hauptzustände (Modi) bestimmt.

- Gesendete Dokumente: Arbeitsbestätigung (Confirmation of Work) ConfWork
- Empfangene Dokumente: Fertigungsauftrag (Production Order Request) ProdOrdReq
- Hauptzustände: Initial, Prepare, Execute, Final

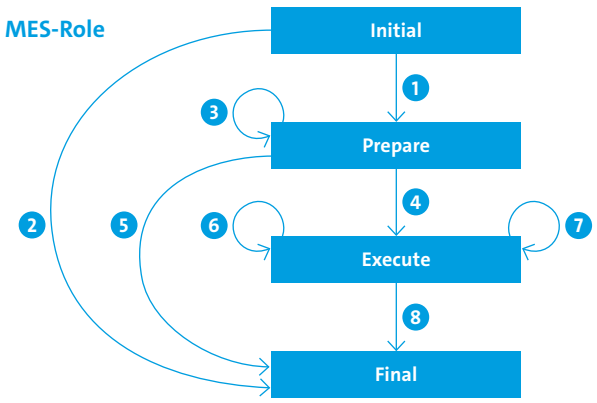

Abbildung 8: Zustandsdiagramm für den Hauptzustand der MES-Rolle in der ERP-MES-Interaktion.

Initial: Das MES wartet auf einen Fertigungsauftrag des ERP, um einen Produktionsprozess zu starten. Empfängt es einen solchen, überprüft das MES, ob es ihn ausführen kann. Wenn ja geht es über in den Prepare Modus (1). Wenn nicht, meldet es das an das ERP zurück durch eine als Absage formulierte Arbeitsbestätigung (ConfWork) und geht in seinen Final-Modus (2).

Prepare: Das MES bereitet den Produktionsprozess vor. Änderungen am bestehenden Fertigungsauftrag werden darauf geprüft, ob sie berücksichtigt werden können. Kann es die Änderung des Fertigungsauftrags berücksichtigen, tut es das (3). Führt die Änderung hingegen dazu, dass der Fertigungsauftrag nicht mehr ausgeführt werden kann, meldet es das an das ERP zurück durch eine als Absage formulierte Arbeitsbestätigung (ConfWork) und geht in seinen Final-Modus (5).

Ist das MES mit seinen Vorbereitungen fertig, geht es ansonsten in den Execute-Modus über (4)

Execute: Der Produktionsprozess wird durchgeführt. Kommt nun ein geänderter Fertigungsauftrag ein, muss Einzelfall-geprüft werden, ob und wie dieser noch Berücksichtigung finden kann (6). In Abhängigkeit von den Einstellungen wird das ERP über den Arbeitsfortschritt mittels entsprechender Arbeitsbestätigungen informiert (7). Ist die Produktion beendet, geht das MES in den Final-Modus über und setzt das ERP davon in Kenntnis (8).

Final: Der Produktionsprozess ist MES-seitig abgeschlossen.

8 Bewertung bestehender Technologien zur Unterstützung elektronischer Interaktionen

8 Bewertung bestehender Technologien zur Unterstützung elektronischer Interaktionen

Gemäß des Referenzmodells semantischer Interoperabilität lassen sich interaktionsunterstützende Technologien unterscheiden in solche, die sich ausschließlich auf den Transport der Informationen beschränken und keine Annahmen über deren Inhalte oder Struktur treffen, und solche, die sich auf diese Inhalte beziehen. Erstere nennen wir »transportbezogene«, letztere »verarbeitungsbezogene« Interaktionstechnologien, bzw. kürzer Kommunikationstechnologien und Anwendungstechnologien. Mittels kryptografischer Methoden, nämlich der Verschlüsselung der transportierten Inhalte lässt sich diese Unterscheidung erzwingen. Die Kommunikationstechnologien bieten u.a. die Elementarfunktionen für den Datentransport zwischen den Applikationen an und legen damit u.a. die Datentyp-Semantik des Datentyps »Adresse« fest.

8.1 Kommunikationstechnologien

Nach dem Referenzmodell semantischer Interoperabilität sind an Kommunikationstechnologien, je nachdem welche Interaktionsklasse sie unterstützen sollen, unterschiedliche Anforderungen zu stellen. In horizontalen Interaktionen müssen Dokumente an bekannte Empfänger versendet werden, die einem Verarbeitungskontext, der zuständigen Prozessinstanz im Empfänger zugeordnet werden müssen. In vertikalen Interaktionen ist ein Verwendungskontext zu etablieren, innerhalb dessen Operationen aufgerufen und Ergebnisse, ggfs. ergänzt um Ausnahmen, zurückgeliefert werden und Events der verwendeten Objekte den Verwendern zugeordnet werden.

Entsprechend haben sich recht unterschiedliche Kommunikationstechnologien etabliert, von denen wir einige wenige exemplarisch in Bezug auf unser Referenzmodell einordnen.

Beispiele von Kommunikationstechnologien im beschrieben Sinn sind etwa HTTP(S), MQTT[18] oder AMQP[19]. Sie unterscheiden sich v.a. hinsichtlich der Klasse der unterstützen Interaktionen.

HTTP(S) wurde ursprünglich entworfen, um den Zugriff auf Hypertext-Dokumente des World Wide Web zu ermöglichen. Sein Aufruf-Antwort-Schema kann jedoch auch dazu verwendet werden, Daten an einer http-Adresse abzuliefern. Die Informationsflussrichtung kann also grundsätzlich in Richtung des Aufrufs, entgegen der Aufrufrichtung oder auch in beide Richtungen erfolgen. Das macht den Informationstransport über HTTP(S) einerseits recht vielseitig, andererseits aber auch mehrdeutig im Sinne des Referenzmodells.

18 ↗http://mqtt.org/

19 ↗https://www.amqp.org/

MQTT ist ein Beispiel für eine Kommunikationstechnologie, die mittels Publish-Subscribe dediziert Szenarien mit unidirektionalem Informationsfluss im Sinne eines anonymen Beobachtens unterstützt.

Auch **AMQP** ist ein Protokoll zur Kommunikation und betrachtet die zu übermittelnden Daten entsprechend als Blackbox. Es definiert die Regeln für den Nachrichtenaustausch zwischen Clients und einem Message-Broker bzw. zwischen verschiedenen Message-Brokern, ohne sich auf eine bestimmte Netzwerktopologie festzulegen. Die im Vergleich zu MQTT bewusste Topologie-Agnosie von AMQP trägt wesentlich zu seiner höheren Komplexität bei. Insbesondere unterstützt es Workqueues, Publish-Subscribe, selektives Message-Routing, den auf der Auswertung von Mustern basierten Empfang (»Topics«) oder auch Request-Reply-Muster für Remote Procedure Calls.

8.2 Anwendungstechnologien

Verarbeitungsorientierte Interaktionstechnologien oder auch Anwendungstechnologien unterstützen Anwendungen im Zugriff, in der Verarbeitung und der Bereitstellung von Informationen, etwa durch die Unterstützung der Definition von Datenmodellen[20], Objektmodellen, Dokumentenstandards, Bibliotheken von Hilfsoperationen etc. Sie setzen die Einbindung geeigneter Kommunikationstechnologien voraus. Aus dieser allgemeinen Beschreibung lässt sich schon ableiten, dass es nicht »die eine« Anwendungstechnologie geben kann, die den Aufwand zur Erreichung von domänenübergreifender Interoperabilität von IT-Applikationen auf ein Minimum reduziert. Stattdessen gibt es mehrere dieser Technologien die in der Regel komplementär, d.h. sich ergänzend eingesetzt werden. Beispielhaft greifen wir OPC UA und oneM2M heraus. Während OPC UA primär für die Interaktion zwischen industriellen Geräten innerhalb von Industrieanlagen entwickelt wurden, zielt oneM2M auf die Kommunikation von IoT Geräten über drahtgebundenen und drahtlosen Weitverkehrsnetzwerk Technologien und die sich daraus ergebenden Herausforderungen ab.

OPC UA[21] ist ein typischer Standard zur Verwaltung und Verwendung von remote, Client-Server-basierten Objektmodellen im Rahmen vertikaler Interaktionen mit remote Methodenaufrufen, das aus dem DCOM-Standard abgeleitet wurde. In einem Informationsmodell werden Adressierungsschemata der Instanz- und Typhierarchie, die Basistypen und -objekte, sowie das Serverobjekt festgelegt. UA Applikation müssen dann definierte Untermengen der OPC UA Fähigkeiten, sogenannte Profile, implementieren um interoperabel zu sein. Ebenso werden Events innerhalb eines Verwendungskontextes unterstützt. Weiterhin wird Funktionalität bereitgestellt, um historische und aggregierte Daten zu verwalten. In seiner neuesten Version wird auch ein Publish-subscribe Interaktionsmodell definiert.

20 Siehe auch RFC3444 zur Unterscheidung zwischen Informations- und Datenmodell.

21 ↗https://opcfoundation.org/

oneM2M[22] ist ein globales »Partnership Project« der weltweit 8 ICT SDOs[23]. oneM2M spezifiziert herstellerunabhängige Technologie für den sicheren und zuverlässigen Transport von IoT- und Prozess-Daten, zwischen beliebigen Feldgeräten (Sensoren, Gateways etc..) und Backend/Server/Cloud-Infrastrukturkomponenten über Weitverkehrsnetzwerke (Wide Area Networks –WAN). Der herstellerunabhängige »gemeinsame Software Service Layer« mit global standardisierten APIs unterstützt darüber hinaus Funktionen die domänenübergreifend im IoT-Umfeld benötigt werden, wie Authentifizierung, Autorisierung und Registrierung von Applikationen, das Zwischenspeichern von Daten, sowie das Planen und Steuern der Datenübertragung. oneM2M stellt unter anderem auch ein standardisiertes Device Management inklusive beispielsweise FTOA Funktionalität (Firmware Updates Over The Air) bereit.

Zusätzliche Funktionen gestatten die Anreicherung von Geräten und Daten mit Informationen zur semantischen Suche (sog. »Semantic Support«; Bsp. »Finde alle Geräte, die zum Messen der Temperatur eingesetzt werden«). Sie bieten einer Applikation die Möglichkeit eine Vielzahl von Datenstrukturen mit einem Kommando anzusprechen bzw. zu verändern (sogenannte Group Management Funktionen; Bsp. »alle Lampen der Gruppe ‚Erdgeschoß' ausschalten«). Applikationen können über den oneM2M Service Layer über die Veränderung von Daten verständigt werden (sog. »Subscription Notification«). Daten des oneM2M Service Layer können durchsucht werden. (sog. »Discovery«; Bsp. »welche Lampen mit dem Label ‚Erdgeschoß' sind an das Gerät angebunden«) und die Position von Geräten kann abgefragt werden (sog. »Location«-Funktion). Datenmodelle werden durch Firmen in oneM2M eingebracht und im Smart Device Template (SDT) dokumentiert.

Beispiele für die aufwandsarme Herstellung von Kommunikation zwischen Applikationen unterschiedlicher Domänen mittels oneM2M speziell im Smart City Bereich wurden im französischen Bordeaux[24] oder im koreanischen Busan[25] umgesetzt. Ein anderes Beispiel für den Einsatz von M2M ist die domänenübergreifende Kommunikation zwischen »Agriculture« und »Automotive«[26].

22 ↗http://www.onem2m.org/

23 ↗http://onem2m.org/about-onem2m/partners

24 ↗https://aioti.eu/blog-future-proof-smart-cities-the-case-of-bordeaux/

25 ↗https://www.itu.int/en/ITU-T/Workshops-and-Seminars/bsg/20180506/Documents/BSG%20Session%20on%20IoT%20(Training%20on%20IoT)%206May2018/20180506%20Cairo%20ITU-T%20BSG%20v3.pdf

26 siehe [»ETSI TR 103 545 SmartM2M; Pilot test definition and guidelines for testing cooperation between oneM2M and Ag equipment standards«]

9 Ergebnisse und deren Einordnung im Gesamtkontext

9 Ergebnisse und deren Einordnung im Gesamtkontext

Die wesentliche Bedeutung des vorgestellten Referenzmodells semantischer Interoperabilität liegt in einem besseren Verständnis der Natur der Interaktionsnetze, die die Basis des Internet of Things darstellen. Basierend auf diesem Referenzmodell lassen sich Technologien in Transport-orientierte und Applikations-orientierte Technologien unterscheiden. Und weitergehend lassen sich die Komponenten hinsichtlich ihres Potentials, die verschiedenen Klassen der Interaktion zwischen Komponenten zu unterstützen, beurteilen. Auch die neuere Entwicklung in der Diskussion um die Verwaltungsschale sehen wir als Bestätigung des vorgestellten Modells, als dort mittlerweile drei Arten unterschieden werden, die sich in unserem Modell als Datenmodell, als Komponente mit hierarchischem Interface sowie als Komponenten mit horizontalem Interface darstellen.

Das wesentliche Ergebnis der Untersuchung ist:

1. Ein Interface zwischen zwei Systemen ist im Sinne des Referenzmodells semantischer Interoperabilität nur soweit wohldefiniert, als es eine Aussage über das Transformationsverhalten, also die Abbildungseigenschaften des Systems macht. Denn nur dann lassen sich weitergehende Aussagen über »Interoperabilität« in einem semantischen Sinn, also bezogen auf die jeweilige Verarbeitung der ausgetauschten Informationen, machen.

2. Die Ausprägungen der Informationstransportkategorien uni-/bidirektional und der Informationsverarbeitungskategorien A-/Synchronität, Zustandsbehaftung und Nicht-/Determinismus schlägt sich in der Gestalt der Interfaces nieder, weswegen es verschiedene Interfaceformen gibt, die es technologisch zu unterstützen gilt.

3. Horizontale Interaktionen im Sinne des vorgestellten Referenzmodells semantischer Interoperabilität zeichnen sich dadurch aus, dass sich alle Interaktionsakteure grundsätzlich gleich bzgl. des Transports und v.a. in der Verarbeitung der ausgetauschten Informationen verhalten, nämlich asynchron, zustandsbehaftet und nichtdeterministisch. Vertikale Interaktion liegt vor, wenn sich bei bidirektionalem Informationsaustausch die Akteure grundsätzlich asymmetrisch bzgl. der Informationsverarbeitung verhalten.

4. Horizontale Interaktionen zwischen Applikationen unterschiedlicher Domänen bilden den Kern der immer größer und dichter werdenden Interaktionsnetzwerke.

5. Zerlegt man das Problem der aufwandsarmen Herstellung von Interoperabilität in das Problem des Informationstransports (bzw. der Kommunikation) und der Informationsverarbeitung, so stellt man fest, dass das Problem der Kommunikation vergleichsweise gut verstanden und standardisierte Lösungen hierfür verfügbar sind. Das Problem der aufwandsarmen Herstellung von aufeinander abgestimmter Informationsverarbeitung insbesondere in komplexen, zustandsbehafteten Applikationen, die an vielen (horizontalen) Interaktionen beteiligt sind, ist hingegen bisher weniger gut verstanden und ganz wesentlich ein Problem der Architektur der beteiligten Applikationen, die möglichst robust gegen Änderungen in einzelnen Interaktionen sein sollte.

6. Die technologische Unterstützung sowohl für unidirektionale Interaktionen als auch für hierarchische Interaktion ist recht ausgereift. Für die horizontalen Interaktionen im Bereich IoT im Allgemeinen oder auch I4.0 im Speziellen, sind entsprechende Standards leider noch nicht so weit verbreitet eingesetzt.

Dann wäre es die Aufgabe der Applikationsentwickler, ihre Applikationen so zu strukturieren, dass es möglichst einfach wird, die für die Interaktion notwendigen Rollen zu implementieren bzw. diese entsprechend robust gegen Änderungen der einzelnen Interaktionen werden. D.h. der Aufwand zwischen den Änderungen in einzelnen Interaktionen und der Aufwand zur Adaptation der Applikationen sollten in einem möglichst gutartigen Verhältnis zueinander stehen. Dazu gehört auch entwicklungsseitig, die notwendigen Entitäten der Zustandsmodelle und verwendeten Dokumente zur Verfügung zu stellen.

Die möglichst einfache Abbildung von wohldefinierten Protokoll-Schnittstellen in den Applikationen ist insbesondere auch deswegen wichtig, als die Welt der Nebenläufigkeit und parallelen, nichtdeterministischen Interaktionen durch die Möglichkeit sehr subtiler Fehler uns vor neue Herausforderungen stellt. Experten wie Gerard J. Holzmann gehen so weit zu sagen, dass ein nicht-validiertes Protokoll im Zweifelsfall fehlerhaft ist. Soll die Entwicklung weiter in Richtung bidirektionale, automatisierte Interaktion zwischen Asset-Instanzen gemäß RAMI4.0 (u.a. Maschinen, Fabriksystemen, Unternehmen) gehen und auch weitergehende, nicht-funktionale Aspekte wie z.B. Autonomie und Souveränität gemäß dem Leitbild 2030 der Plattform Industrie 4.0[27] berücksichtigt werden, müssen wir in der Lage sein, solche Fehlerquellen systematisch auszuschließen.

27 Leitbild 2030 für Industrie 4.0, vgl. ↗**https://www.plattform-i40.de/PI40/Redaktion/DE/Standardartikel/leitbild.html**

Danksagung

Besonderer Dank gilt der Projektgruppe Kommunikationsprotokolle, insbesondere den folgenden Akteuren:

- Andreas Neubacher, Deutsche Telekom / T-Mobile International Austria
- Andreas Kraft, Deutsche Telekom / T-Systems
- Dr. Alexander Willner, Fraunhofer FOKUS Institut für offene Kommunikationssysteme
- Johannes Reich, SAP SE
- Jörg Wende (Leiter der Projektgruppe), IBM Deutschland GmbH
- Matthias Lieske, Hitachi Europe
- Lukas Klingholz, Bitkom e.V.
- Roland Steinke, Fraunhofer FOKUS Institut für offene Kommunikationssysteme
- Stefan Grieß, Asseco Solutions AG
- Dr. Thomas Usländer, Fraunhofer IOSB

Bitkom vertritt mehr als 2.700 Unternehmen der digitalen Wirtschaft, davon gut 1.900 Direktmitglieder. Sie erzielen allein mit IT- und Telekommunikationsleistungen jährlich Umsätze von 190 Milliarden Euro, darunter Exporte in Höhe von 50 Milliarden Euro. Die Bitkom-Mitglieder beschäftigen in Deutschland mehr als 2 Millionen Mitarbeiterinnen und Mitarbeiter. Zu den Mitgliedern zählen mehr als 1.000 Mittelständler, über 500 Startups und nahezu alle Global Player. Sie bieten Software, IT-Services, Telekommunikations- oder Internetdienste an, stellen Geräte und Bauteile her, sind im Bereich der digitalen Medien tätig oder in anderer Weise Teil der digitalen Wirtschaft. 80 Prozent der Unternehmen haben ihren Hauptsitz in Deutschland, jeweils 8 Prozent kommen aus Europa und den USA, 4 Prozent aus anderen Regionen. Bitkom fördert und treibt die digitale Transformation der deutschen Wirtschaft und setzt sich für eine breite gesellschaftliche Teilhabe an den digitalen Entwicklungen ein. Ziel ist es, Deutschland zu einem weltweit führenden Digitalstandort zu machen.

Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e.V.

Albrechtstraße 10
10117 Berlin
T 030 27576-0
F 030 27576-400
bitkom@bitkom.org
www.bitkom.org

bitkom