http://dx.doi.org/10.25673/39550

Distributed Asset Management in Industrie 4.0

Nico Braunisch, Frank Hilbert, Martin Wollschlaeger
Institut für Angewandte Informatik - TU Dresden
Professur für Prozesskommunikation

Nöthnitzer Str. 46
01187 Dresden
nico.braunisch@tu-dresden.de
frank.hilbert@tu-dresden.de
martin.wollschlaeger@tu-dresden.de

Abstract: Damit beliebige Assets in Industrie 4.0-Systemen verwendet werden können, müssen deren Informationen, Merkmale und Verhalten digital abgebildet werden. Dies wird über das Konzept der Verwaltungsschale realisiert. In der Umsetzung dieses Konzeptes stößt man jedoch auf mehrere Herausforderungen. Die Verwaltungsschale bildet dabei die verschiedenen Aspekte der Assets in Teilmodellen ab. Die verschiedenen fachlichen Zuständigkeiten der Teilmodelle führt einerseits zu sehr unterschiedlichen Eigenschaften im Hinblick auf die Dynamik dieser Teilmodelle, andererseits können Leistungs- und Verbindungseigenschaften von I4.0 Assets unterschiedlichen Beschränkungen unterliegen. So können die Assets beispielsweise bezüglich der Leistungsfähigkeit und somit der möglichen Operationen als auch hinsichtlich ihrer Erreichbarkeit im Netzwerk beschränkt sein. Daher bietet es sich an, die Verwaltungsschale und deren Services auf unterschiedlichen Systemebenen zu verteilen. Auf diese Weise lassen sich die verschiedenen Anwendungsszenarien eines Assets gezielter adressieren. Dynamische Aspekte können so in Teilmodellen in direkter Nähe des Assets realisiert werden, während aufwendigere Operationen an rechentechnisch mächtigere Systemebenen, wie zum Beispiel die Cloud, ausgelagert werden können. Ebenso können relativ statische Informationen von der Erreichbarkeit des Assets entkoppelt werden, indem diese Informationen von einer zuverlässig erreichbaren Ebene, zum Beispiel einem Gateway, verwaltet werden. In dieser Arbeit wird ein verteilter Ansatz zur Bereitstellung und Ausführung von (pro-)aktiven Verwaltungsschalen für I4.0 Komponenten vorgestellt.

1 Einleitung

Das Management der Assets in Industrie 4.0 (I4.0) wird über die Modellierung der Verwaltungsschalen (engl. Assets Administration Shell, kurz AAS) realisiert [1]. Durch diese wird in Verbindung mit dem Asset die jeweilige I4.0 Komponente gebildet. Betrachtet man zukünftige I4.0 Industrieanlagen als verteilte Systeme, wie in Abbildung 1 dargestellt, im Sinne des Industrial Internet of Things (IIoT) so müssen die Verwaltungsschalen ebenfalls eine aktive Rolle in der Kommunikation spielen. Daher sind sie als (pro-)aktive Verwaltungsschalen auszuführen [2]. In dieser Arbeit wird eine adaptive Laufzeitumgebung für (pro-)aktive Verwaltungsschalen mit variablen Schnittstellen für unterschiedliche Kommunikationswesen vorgestellt. Die Laufzeitumgebung ist dabei plattformübergreifend und modular entsprechend der Service orientierten Architektur von I4.0 umgesetzt [2]. Um die Laufzeitumgebung möglichst flexibel und ressourcenschonend zu gestalten wird auf dezentrale (Micro-)Services auf der Basis von .NET zurückgegriffen.

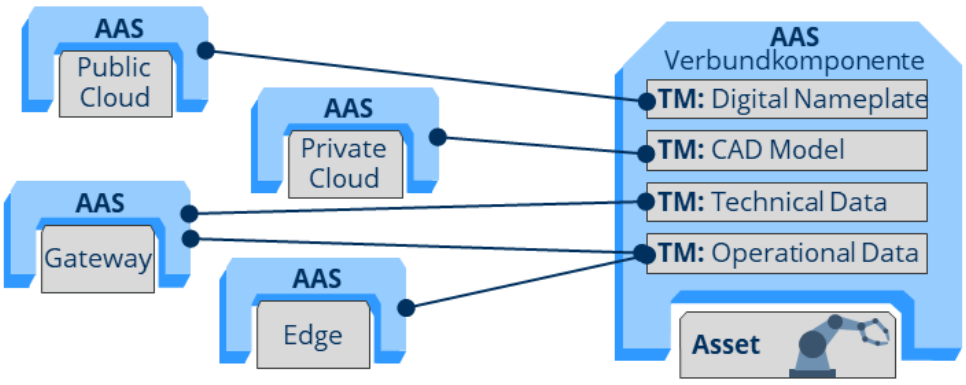

Abbildung 1: Distributed Asset

2 Problemstellung

Die verschiedenen Assets in I4.0 Systemen verfügen allerdings im Allgemeinen über stark variierende Leistungs- und Verbindungseigenschaften. Um die Verwaltungsschalen dennoch zuverlässig erreichbar bereitzustellen, muss ebenfalls die Laufzeitumgebung der Verwaltungsschalen dezentral gestaltet werden und somit ebenfalls, wie in Abbildung 1 dargestellt, die Verwaltungsschalen an unterschiedlichen Orten oder Ebenen im I4.0 System zur Verfügung stehen [3]. Eines der bestimmenden Kriterien für eine (pro-)aktive Verwaltungsschale für I4.0 Komponenten ist die Leistungsfähigkeit der eingesetzten Hardware und Software. Verfügt ein Asset nicht über hinreichende

Rechenleistung für eine (pro-)aktive Verwaltungsschale, so muss die Verwaltungsschale an einer Stelle mit mehr Leistungsfähigkeit ausgeführt werden. In diesem Fall bieten sich Hubs oder Gateways, wie in Abbildung 2 dargestellt, an.
Ein weiteres stark einschränkendes Kriterium ist die verfügbare Konnektivität. Nicht jedes Asset verfügt über eine große Bandbreite und die bestmögliche Übertragungsart. Zudem kann die Verfügbarkeit der Assets im IIoT zeitweise stark eingeschränkt sein. So muss out-of-band Kommunikation und das Handover von Verbindungen berücksichtigt werden. Beide zuvor genannten Einschränkungen können auch durch eine IIoT typische Limitierung im Energieverbrauch zustande kommen, zum Beispiel bei Verwendung batteriebetriebener Assets. In diesem Falle werden Bandbreite und Verfügbarkeit im Energiesparbetrieb genauso gedrosselt wie die Rechenleistung des Assets.

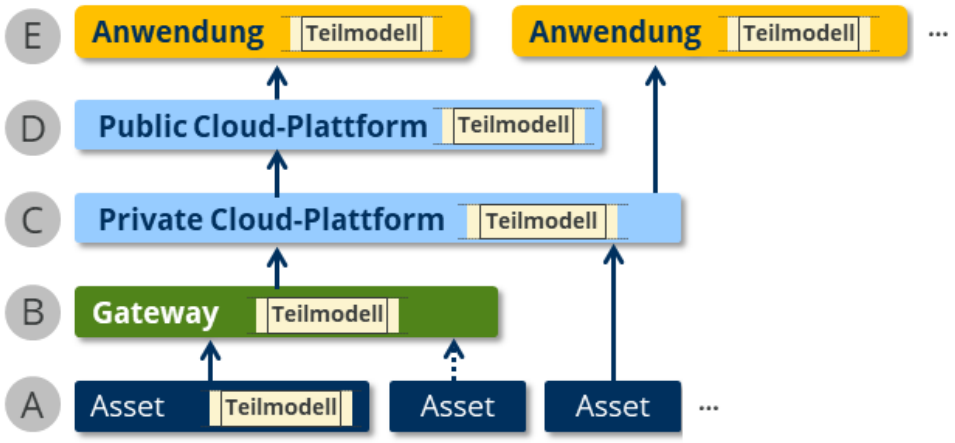

Abbildung 2: Verteilte Verwaltungsschale

Aus den zuvor genannten Gründen müssen Assets je nach gegebener Situation selbsttätig oder durch andere, übergeordnete Komponenten verwaltet werden. Zusätzlich erzeugen die verschiedenen Aspekte der Assets in den Teilmodellen sehr unterschiedliches Verhalten im Hinblick auf die Dynamik und der Kommunikationsparadigmen. So müssen dynamische Aktualwerte kontinuierlich und mit minimaler Latenz abgerufen werden können, wohingegen zum Beispiel Operations und Events in unregelmäßigen Abständen und Informationen zur Identifizierung, Versionsnummer oder Produktdatenblätter relativ selten übertragen werden.
Daher dient eine Verwaltungsschale nicht nur als statische Referenz und Datenquelle für die I4.0 Merkmale. Sie muss darüber hinaus den Zugriff auf Aktualdaten und sich ändernde Relationen erlauben. Dies führt zu verschiedenen Abhängigkeiten der Relationen und Referenzen in den Teilmodellen der Verwaltungsschale. Diese werden bereits in der Modellierung der Schale mitgestaltet. Durch die Verteilung der Verwaltungsschale in Form einer Verbundkomponente (siehe Abbildung 1) und die Verwendung der Bill-Of-Material werden die Relationen und Referenzen der Assets in einem I4.0

System abgebildet [4]. Die Laufzeitumgebung muss diese wiederum auflösen können. Durch den dezentralen Charakter dieser verteilten Verwaltungsschale ergibt sich auch eine Vielfalt in den Schnittstellen der Laufzeitumgebungen. Der Zugriff auf die Verwaltungsschale wird auf unterschiedlichen Systemebenen bereitgestellt Dies wird in Abbildung 2 schematisch dargestellt. Viele der dort benötigten sowie angebotenen Kommunikationsendpunkte genügen je nach Systemebene unterschiedlichen Charakteristika. So können Aktualwerte direkt am Gerät abgerufen werden, während relativ statische Informationen wie Versionsnummern oder Produktdatenblätter hingegen als Datenbank in der Cloud bereitgestellt werden können. Die entstehende Multi-Interface-Problematik muss durch die Laufzeitumgebung aufgelöst werden. Bei der Verteilung kommen die Aspekte der I4.0-Verbundkomponente zum Tragen. Hier sind spezielle Referenzen und Relationen notwendig, um die betreffenden Teilmodelle und deren Merkmale für die unterschiedlich dynamischen Interaktionen abzubilden.
Zur Erfüllung der angeführten Anforderungen stellt der Beitrag das Konzept, Entwurf und Umsetzung einer Laufzeitumgebung für eine verteilte Verwaltungsschale vor. Diese Laufzeitumgebung ist durch eigenständige Kommunikation und kooperative Interaktion in der Lage den Kriterien einer (pro-)aktiven und dezentralen Verwaltungsschale zu genügen. Mit dieser Herangehensweise lassen sich für die verschiedenen Anwendungsszenarien eines I4.0 Systems die Dynamiken der Teilmodelle gezielt adressieren.

3 Anforderungen an Management in Industrie 4.0

In Device Management in Industrial IoT wurden bereits die verschiedenen Anforderungen im Hinblick auf Industrial IoT betrachtet [5]. Für das Asset Management in verteilten System der Industrie 4.0 treffen diese Kriterien ebenfalls zu. Diese Erstrecken sich nach wie vor über die funktionalen Aspekte wie zum Beispiel Überwachung, Diagnose, Wartung und Fehlerbehandlung bis hin zu organisatorischen Aspekten wie zum Beispiel Bereitstellung, Security, Membership und Accounting. Für die Laufzeitumgebung der Verwaltungsschale spiegeln sich besonders für die funktionalen Aspekte die Anforderungen in den Charakteristiken der Schnittstellen wieder. Dabei wird, wie in Abbildung 3 dargestellt, zwischen unären Aufrufen und Streaming unterschieden. Unäre Aufrufe sind besonders im Bereich von Alarmen und Monitoring interessant. Das Streaming Konzept adressiert den kontinuierlichen Abruf von dynamischen Aktualwerten mit minimaler Latenz. Die Aufrufe können wiederum synchron, für feste Funktionsabfolgen wie zum Beispiel im Fall der Logging Aktivierung oder asynchron, zum Beispiel für Updates, erfolgen. In beiden Fällen wird jedoch der Kommunikationsfluss von der aufrufenden Verwaltungsschale (AAS1) gesteuert. Ein wesentlich flexiblere Art der Interaktion stellt die Kommunikation über Streams dar. Hierbei wird zwar von der aufrufenden Verwaltungsschale (AAS1) weiterhin der Kommunikationskanal bereitgestellt, jedoch erfolgt das Senden bzw. Empfangen von Nachrichten beider Kommunikationspartner (AAS1 und AAS2) unabhängig vom der aufrufenden Verwaltungsschale (AAS1). Dies ist besonders im Bereich von Alarmen und Monitoring interessant. Darüber hinaus ist Streaming von Relevanz wenn dynamische Aktualwerte kontinuierlich und mit minimaler Latenz abgerufen werden.

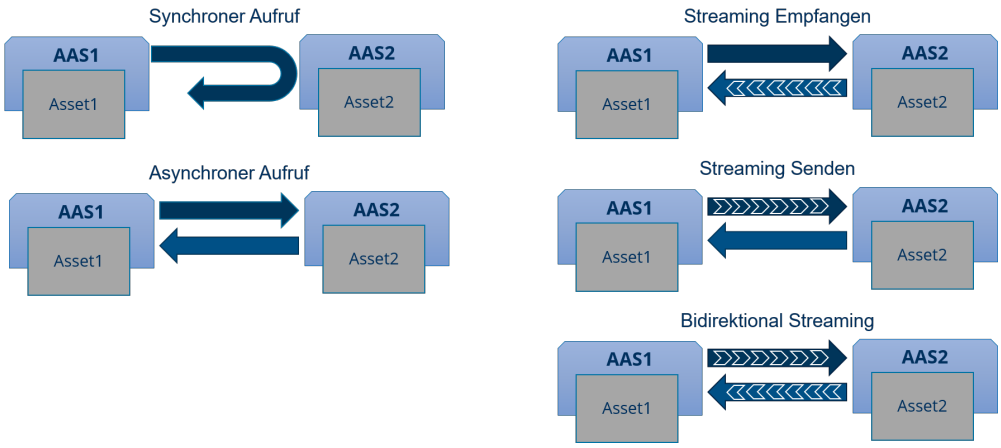

Abbildung 3: Anforderung an API der Laufzeitumgebung

4 Design Distributed Asset Management

Die Verwaltung der Assets im I4.0 System wird über ihre Verwaltungsschale organisiert. Diese stellt die benötigten Dienste im Sinne einer Serviceorientierten Architektur (SOA) bereit. Um die einzelnen Dienste der Verwaltungsschale aktiv nutzen zu können wird eine geeignete Laufzeitumgebung benötigt. Die Umsetzung dieser Dienste stellt jedoch für den Entwickler einen erheblichen Aufwand dar. Um den Mehraufwand bei der Serviceimplementierung möglichst gering zu halten, beruht die hier vorgestellte Umsetzung auf den nun vorgestellten Designprinzipien.
In der SOA bieten die einzelnen Assets ihre Fähigkeiten über Services an und nutzen zur Erfüllung ihrer Aufgaben die Services anderer Assets. Dabei soll die Zuständigkeit beziehungsweise der Anwendungsbereich eines Dienstes möglichst minimal sein. Im vorgestellten Design wird dabei die Umsetzung mittels Microservices eingeführt. Dies ermöglicht es die einzelnen Services unabhängig voneinander zu entwickeln und Abhängigkeiten zur vermeiden. Durch die Modellierung der Eigenschaften der Assets in der Verwaltungsschale werden dabei die Schnittstellen der einzelnen Services klar definiert. Komplexere Eigenschaften und Funktionen können dabei durch eine geeignete Servicekomposition aus anderen Services zusammensetzt werden.
Die Umsetzung der (Micro-)Services ist jedoch weiter stark implementierungsabhängig. Dies bezieht sich sowohl auf die Wahl der Entwicklungsumgebung, Programmiersprache und Ausführungsumgebung. Für die Laufzeitumgebung der Verwaltungsschale ergeben sich dabei noch Abhängigkeiten der jeweiligen Betriebssysteme sowie deren Laufzeitumgebung und Bibliotheken der verwendeten Programmiersprachen. Aus diesem Grund soll die Laufzeitumgebung der Verwaltungsschale plattformübergreifend gestaltet werden. In unserer vorgestellten Laufzeitumgebung wird daher das .NET Fra-

mework als plattform- und programmiersprachenübergreifende Umsetzung verwendet. Weiter können die (Micro-)Services in einer extra Virtualisierungsumgebung wie zum Beispiel Docker oder Kubernetes betrieben werden.
Um den Aufwand für den Entwickler der (Micro-)Services zu minimieren, soll bei der Umsetzung der Service der Großteil des Arbeitsaufwandes durch die automatisierte Generierung von Quellcode beziehungsweise Codegerüsten erfolgen. In der vorgestellten Lösung wird aus diesem Grund eine Interfacebeschreibungssprache (IDL) verwendet, um die Definition der Services, ihrer Endpunkte sowie deren Input- und Out-Parameter zu definierten.
Anschließend können für die einzelnen Services gezielt, entsprechend der gewünschten beziehungsweise benötigten Programmiersprachen, Frameworks und Ausführungsumgebungen, die entsprechenden Servicestubs generiert werden. Diese liefern unabhängig von der Umsetzung jedoch immer die jeweiligen Nachrichtenkanäle mit aufrufbaren Serviceendpunkten, definierten Nachrichten, sowie die dafür benötigten Kommunikationsdienste.

5 Multi API

Um den verschiedenen Charakteristika der unterschiedlichen Systemebenen Rechenschaft zu tragen, müssen die einzelnen (Micro-)Services über unterschiedliche Schnittstellen verfügen. Diese werden, wie in Abbildung 4 schematisch dargestellt, in der jeweiligen Laufzeitumgebung (hellgrüner Kreis) bereitgestellt.

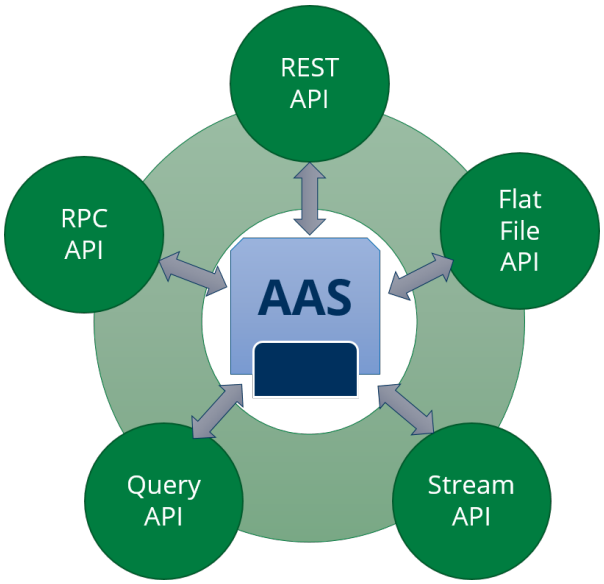

Abbildung 4: Unterschiedliche APIs der Laufzeitumgebung

Wie in Verwaltungsschale im Detail Teil 2 ausgeführt, muss die Laufzeitumgebung eine *REST API* für die Kommunikation mit der Verwaltungsschale bereitstellen [6]. Diese wird vorwiegend für die Kommunikation mit der I4.0 Infrastruktur wie Registry, Directory, Repositories beziehungsweise der jeweiligen Laufzeitumgebungen und Engines der Teilmodelle untereinander verwendet.
Für die Schnittstelle in die passive Verwaltungsschale muss die Laufzeitumgebung eine *Flat File API* bereitstellen. Diese wird verwendet um komplette AASX Pakete [1] oder Modellbestandteile auszutauschen. Der Zugriff auf die Modellbestandteile erfolgte dabei in serializierter Form als JSON oder XML Repräsentation. Weiter werden über die Flat File API die Zugriffe auf Anhänge (Supplementary Files) in der Verwaltungsschale wie Handbücher in PDF Dateien, Engineeringdokumente wie EPLAN oder Bilddateien ermöglicht. Diese API kann ebenfalls als Schnittstelle für assetsspezifische Konfigurationsdateien, wie ISO Images oder INI-Files, genutzt werden.
Für die Abfrage der Eigenschaften und Fähigkeiten der Assets wird in der Laufzeitumgebung eine *Query API* benötigt [7]. Diese kann über eine geeignete Abfragesprache (Query Language) die bestimmten Charakteristika der Assets erfassen und wiedergeben. Für das Auffinden der Assets im I4.0 System anhand eines Directory bzw. Repository wird diese API ebenfalls verwendet.
Für den Zugriff auf die funktionalen Eigenschaften der Assets werden in der Laufzeitumgebung je nach Charakteristik des Zugriffs zwei unterschiedliche APIs benötigt. Für den Aufruf von Funktionen, abgebildet in der AAS durch Operations, wird eine *RPC API* verwendet. Dies ermöglicht es, die Zugriffe auf die Funktionen der Assets auch remote auszuführen. Dabei können die Funktionen sowohl spezifische Eigenschaften am Asset ändern als auch komplexere Operationen ausführen, welche z.B. für die Berechnung oder Akkumulierung von Aktualwerten vor der Übermittlung genutzt werden.
Sollen Werte kontinuierlich erfasst und übermittelt werden, wird eine *Streaming API* verwendet. Für diese wird zwischen dem sendenden und dem empfangenden Asset ein Nachrichtenkanal aufgebaut. Die Übermittlung der Nachrichten über diesen Kanal erfolgt dann kontinuierlich und zeitunabhängig vom Aufruf. Dabei ist besonders auf einen minimalen Overhead der Nachrichten und eine geringe Latenz bei der Übertragung zu achten. Dies ist besonders bei der Übermittlung von Events zwischen den Assets wichtig.

6 Generative Implementierung der (Micro-)Services

Für die Umbesetzung der Laufzeitumgebung wird ein generatives Vorgehen eingeführt, um den Implementierungsaufwand der (Micro-)Services zu minimieren. Die generellen Schritte werden dabei in Abbildung 5 dargestellt.

6.1 Modellierung der Merkmale der Assets

Die Endpunkte der Service müssen als Merkmale der Verwaltungsschale modelliert werden (1). Dies erfolgt beispielsweise über den AAS Package Explorer [8]. Dadurch

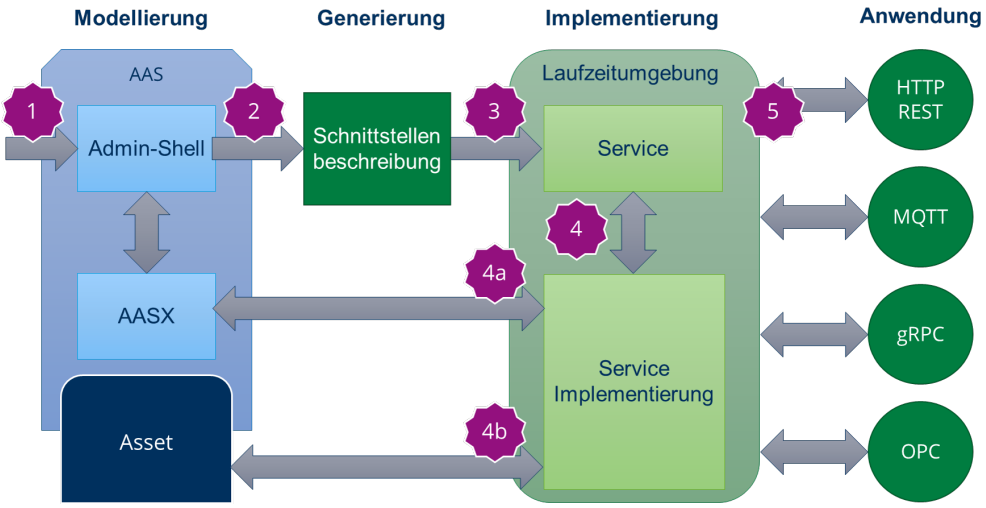

Abbildung 5: Generative Entwicklung der Laufzeitumgebung

werden zugleich die Verwaltungsschale sowie deren verbundenen bzw. eingebetteten Dateien in dem Paketformat der AASX Datei gepackt.

6.2 Generierung der Schnittstellen

Ein weiterer Vorteil in der Verwendung des AAS Package Explorer für die Modellierung der Verwaltungsschale liegt in der Generierung der Schnittstellen für die Endpunkte. Durch die von uns geschaffene Erweiterung des AAS Package Explorer kann die Schnittstellenbeschreibung mittels Protocol Buffers aus der Modellierung direkt exportiert werden (2). Dabei handelt es sich um eine einfache, leichtgewichtige und vielseitige Schnittstellenbeschreibungssprache zur Definition von Nachrichten und Services. Protocol Buffers ist sprachunabhängig und kann für verschiedene Laufzeitumgebungen genutzt werden. Für die aktive Verwaltungsschale werden die verschiedenen Merkmale auf Nachrichten bzw. Serviceaufrufe abgebildet. Diese bilden die Voraussetzung für die Services im Industrie 4.0 System. Für jedes Teilmodell wird dabei ein eigener Service in Protocol Buffers definiert. Für den Anwendungsfall wird sich auf die Abbildung von Dataelements und Operations beschränkt. Dabei werden für jede Property eine Serviceaufrufe für lesenden und schreibenden Zugriff definiert. Die Ausführung der Operations erfolgt direkt über die Serviceaufrufe. Für die Inputparameter der Operation und die Schreibzugriffe auf die Properties werden einzelne Nachrichten definiert. Ebenso werden die Rückgabewerte des Lesezugriffs auf Properties und die Outputparameter der Operations als Nachrichten codiert. Sollten dabei in der Modellierung Referenzen angewendet werden, können diese ebenfalls als eine einzige Nachricht codiert werden. Andernfalls werden dedizierte Nachrichtendefinitionen erzeugt.

6.3 Generierung der Services

Durch die Verwendung der Protocol Buffers als Beschreibungssprache können die Services direkt aus der Schnittstellenbeschreibung generiert werden (3). Dies kann für unterschiedliche Programmiersprachen und Laufzeitumgebungen erfolgen. Daher können verschiedene Teilmodelle in unterschiedlichen Sprachen realisiert werden. Dies ermöglicht es, für jeden Anwendungsfall die geeignetste Sprache und beste Toolchain zu nutzen.

6.4 Implementierung der Services

Leider bleibt die Generierung der Services durch die Schnittstellenbeschreibung auf das Erzeugen von Stubs beschränkt. Daher müssen die funktionalen Aspekte der Assets durch den Entwickler in der, zur Erzeugen der Stubs gewählten Sprache, implementiert werden (4). Dabei können zwei grundlegende Implementierung unterschieden werden. Zum einem werden die Zugriffe auf die passive Schale in der Form von Zugriffen auf die AASX Pakete betrachtet (4a). Dies kann zum Erfassen von festen Werten in Form von Properties oder Abfrage von Elementen der Verwaltungsschale wie topologische Relationen oder externe Referenzen dienen. Zum andrem müssen die Zugriffe auf die funktionalen Aspekte der Assets über die Services abgebildet werden (4b). Aufgrund fehlender Definition der Umsetzung obliegt es in diesem Fall dem Entwickler der Services, diese zu implementieren. Dies kann beispielweise über externe Prozessaufrufe, Lese- bzw. Schreibzugriff auf Programmparameter oder die Ausführung von Steuerungscode erfolgen. In jedem Fall hängt die Umsetzung stark vom jeweiligen Asset ab.

6.5 Service Endpunkte bereitgestellt

Schlussendlich müssen nur noch die Endpunkte erzeugt werden, um die Dienste nutzen zu können (5). Diese können allerdings durch eine geeignete Laufzeitumgebung der Verwaltungsschale selbsttätig erzeugt und bereitgestellt werden.

6.6 Deployment

Für die Anwendung einer SoA in Industrie 4.0 fehlt nur noch das Deployment der I4.0 Komponente. Diese I4.0 Komponente mit (pro-)aktiver Verwaltungsschale müssen die instanziierten Services des jeweiligen Assets bereitstellen. Wie im vorangegangenen Absatz beschrieben, können diese generell in 5 Schritten erstellt werden, wobei die Schritte wesentlich vereinfacht werden, wenn eine günstige Toolchain für das Deployment gewählt wird. Die Laufzeitumgebung der (pro-)aktiven Verwaltungsschale ist das Kernstück der I4.0 Komponente. Diese bildet alle pro- und reaktiven Teile der Verwaltungsschale ab. Die Laufzeitumgebung stellt für die Verwaltungsschale die Ebene dar, welche ein Betriebssystem für eine Applikation darstellt. Somit bildet die Laufzeitumgebung die Grundlage für die Kooperation und Koordination der Verwaltungsschale.

6.7 Bootstrap der Laufzeitumgebung

Bevor die Verwendung der (Micro-)Services beginnen kann, muss zuvor jedoch die Inbetriebnahme der Verwaltungsschale erfolgen. Diese beinhaltet eigenen Schritte um die Kommunikations- und Reaktionsfähigkeit herzustellen. Dadurch wird die Verwaltungsschale mit Hilfe der Laufzeitumgebung zu einer (pro-)aktiven Verwaltungsschale. Zunächst muss aber eine grundlegende Kommunikationsfähigkeit der Laufzeitumgebung hergestellt werden. Diese ist unter Anderem nötig um die Auffindbarkeit im I4.0 System zu gewährleisten oder die Konfiguration der Laufzeitumgebung zu ermöglichen.

7 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde die Problemstellung von Distributed Systems im Hinblick auf Industrie 4.0 erläutert. Dabei wurde im Speziellen auf das Assetmanagement der I4.0 Komponente als Einheit von Asset und Verwaltungsschale eingegangen. Für die in der SOA von I4.0 benötigten Services wurde die Servicegenerierung, -Implementierung sowie -Komposition vorgestellt. Ein besonderes Augenmerk erfordert dabei die Laufzeitumgebung, welche die Managementfunktionalitäten der I4.0 Assets realisiert.
Um diese möglichst flexibel und ressourcenschonend zu gestalten, wird in dem vorgestellten Konzept auf dezentrale (Micro-)Services auf der Basis von .NET zurückgegriffen. Um für den Entwickler dieser Services den Aufwand zu minimieren wird neben der platformübergreifenden und sprachunabhängigen Entwicklung von .NET Microservices ebenfalls die Servicegenerierung aus der modellierten Verwaltungsschale vorgestellt. Dies ermöglicht es, die in der Verwaltungsschale modellierten Merkmale, über die Generierung einer IDL bis hin zu Serviceendpunkten und -Stubs, eine gezielte Entwicklung der Services ohne den wiederholten Implementierungsaufwand für jeden Service erneut absolvieren zu müssen. Zum momentanen Zeitpunkt muss nach der Servicegenerierung noch die konkrete Serviceimplementierung für die (pro-)aktiven Verwaltungsschalen erfolgen. Der Zugriff auf die passiven Elemente der Verwaltungsschale kann über eine geeignete Abbildung ebenfalls generiert werden. Für den Zugriff auf die hardwarenahen Dienste der Assets erfolgt das Mapping momentan noch manuell. Gleiches gilt für die Servicekomposition und die Kooperation des Assets im I4.0 System.
Als ein weiterer entscheidender Punkt für Distributed Systems im Hinblick auf Industrie 4.0 wurde dabei die Bereitstellung unterschiedlicher Endpunkte für die einzelnen I4.0 Services herausgestellt. Im vorgestellten Konzept wurde zur Lösung eine Multi-API präsentiert, welche unterschiedliche Protokolle und Kanäle realisieren kann. Nach dem bisherigen Stand werden dabei Endpunkte über HTTP/REST, gRPC und OPC UA bedient [9]. Für zukünftige Anwendungsfälle wie fähigkeitsbasiertes Engineering [7], digitale Marktplätze und der Erweiterung der Multi-API für Abfragen der Vewaltungsschalen, kann die Laufzeitumgebung zukünftig um eine Query-API mittels GraphQL oder SPARQL ergänzt werden. Jedoch bedarf es dafür eine abgeschlossene Spezifikation der I4.0 API Definition.

Viele der vorgestellten Probleme und vorgeschlagen Lösungen befinden sich noch im Stadium der Entwicklung von Industrie 4.0. Eine Umsetzung der (pro-)aktiven Verwaltungschale als plattform- und implementierungsunabhänge (Micro-)Services stellt jedoch momentan einen der vielversprechendsten Ansätze dar. Die (Micro-)Services verfügen durch ihre lose Kopplung, minimale Zuständigkeit und maximale Vielseitigkeit über die größte Flexibilität, um auf die sich ändernden Anforderungen reagieren zu können.

Danksagung

Dieser Beitrag resultiert aus dem erfolgreich abgeschlossenen Projekt „Funktionsnachweis der Interoperabilität von fluidtechnischen Komponenten am Beispiel von Plug-and-Produce“ (FKM-Nr.: 7046610), das vom Forschungskuratorium Maschinenbau des VDMA gefördert wurde.

Literaturverzeichnis

[1] S. Bader u. a., *Details of the Asset Administration Shell. Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01)*. Nov. 2020.

[2] T. Miny u. a., *Functional View of the Asset Administration Shell in an Industrie 4.0 System Environment*. Apr. 2021.

[3] S. Bader u. a., *What is the Asset Administration Shell from a technical perspective?* Apr. 2021.

[4] H. Bedenbender u. a., *Beziehungen zwischen I4.0-Komponenten – Verbundkomponenten und intelligente Produktion*. Juni 2017.

[5] N. Braunisch, S. Soler Perez Olaya und M. Wollschlaeger, „Device Management in Industrial IoT“, Nov. 2020.

[6] S. Bader u. a., *Details of the Asset Administration Shell. Part 2 -Interoperability at Runtime - Exchanging Information via Application Programming Interfaces (Version 1.0RC01)*. Nov. 2020.

[7] A. Bayha u. a., *Describing Capabilities of Industrie 4.0 Components*. Nov. 2020.

[8] A. Belyaev u. a., *Diskussionspapier VWS-Referenzmodellierung Exemplarische Modellierung einer fertigungstechnischen Anlage mit AASX Package Explorer auf Basis des VWS-Metamodells*. Apr. 2021. DOI: 10.13140/RG.2.2.17848.47368.

[9] H. Bagci und A. Kara, „A Lightweight and High Performance Remote Procedure Call Framework for Cross Platform Communication“, in *Proceedings of the 11th International Joint Conference on Software Technologies - Volume 1: ICSOFT-EA, (ICSOFT 2016)*, INSTICC, SciTePress, 2016, S. 117–124, ISBN: 978-989-758-194-6. DOI: 10.5220/0005931201170124.