Einsatz einer service-orientierten Architektur zur Orchestrierung eines dezentralen Intralogistiksystems

Jan-Philipp Schmidt, Timo Müller und Michael Weyrich

Zusammenfassung

Softwarearchitekturen sind mit Blick auf die Komplexität von Softwaresystemen ein zentraler Ansatzpunkt, um die Entwicklung, spätere Erweiterungen und die Wartung durchführen zu können. Daher wird in diesem Beitrag wird eine service-orientierte Architektur (SOA) vorgestellt, die dezentrale Logistikmodule in Form von dezentralen Intralogistiksystemen organisiert, sodass Werkstücke auf ihrem individuellen Weg durch das Produktionssystem gesteuert werden können. Dabei bieten einzelne Logistikmodule ihre Transportfähigkeiten in Form von sogenannten Services an. Der individuelle Transport wird durch die Koordinierung der Logistikmodule in einer service-orientierten Architekturermöglicht. Um die Koordinierung zu ermöglichen, wird hierfür ein schlankes und dadurch echtzeitfähiges, busunabhängiges, Kommunikations-Protokoll beschrieben.

1 Flexible Produktionssysteme Kontext Industrie 4.0

1.1 Intralogistik für flexible und rekonfigurierbare Produktionssysteme

Um dem Bedarf an neuartigen Konzepten für eine immer höhere Flexibilität in der Produktion bis hin zur Individualproduktion gerecht zu werden, muss neben den Bearbeitungsprozessen des Produktionssystems auch dessen Intralogistik flexibel gesteuert werden können. Hierzu wurde in der aktuellen Forschungsagenda der Plattform Industrie 4.0 Forschungsbedarf hinsichtlich Logistikkonzepten identifiziert, um die Dynamikanforderungen der flexiblen und kleinteiligen Produktion

J.-P. Schmidt (✉) · T. Müller (✉) · M. Weyrich
Institut für Automatisierungstechnik und Softwaresysteme (IAS), Universität Stuttgart, Stuttgart, Deutschland
E-Mail: ias@ias.uni-stuttgart.de; ias@ias.uni-stuttgart.de; michael.weyrich@ias.uni-stuttgart.de

B. Vogel-Heuser et al. (Hrsg.), *Handbuch Industrie 4.0*, Springer Reference Technik,
https://doi.org/10.1007/978-3-662-45537-1_124-1

bedienen zu können. Neben der Forderung nach erhöhter Flexibilität kommt es zukünftig in der Produktion absehbar immer öfter zu Anforderungsänderungen zur Betriebszeit, welche zur Entwicklungszeit des Produktionssystems nicht vorhersehbar sind (Hansson et al. 2017; Järvenpää et al. 2016; Müller-Schloer et al. 2012). Dies resultiert in einem erhöhten Rekonfigurationsbedarf zur Betriebszeit und somit auch in der Forderung nach einer erhöhten Rekonfigurierbarkeit des Produktionssystems. Bezogen auf die Intralogistik bedeutet dies sowohl einen Bedarf an einer erhöhten Rekonfigurierbarkeit des Intralogistiksystems selbst als auch an einem Intralogistiksystem dessen Funktionalität auch nach einer Rekonfiguration der Bearbeitungsmodule noch zur Verfügung steht.

Die Produktorientierung adressiert die genannten Flexibilitätsanforderungen und ermöglicht mit Hilfe der „intelligenten Produkte“ (IP) die Individualproduktion. Der Forderung nach erhöhter Rekonfigurierbarkeit hingegen, kann durch die Modularität der Anlagensteuerung, bestehend aus Hard- und Software, welche gemeinsam funktionale Einheiten bilden, begegnet werden.

Hieraus ergibt sich der Bedarf an einer Koordination der Module, welche sowohl, potenziell heterogene, Bearbeitungsmodule als auch Logistikmodule sein können. In Bezug auf die Intralogistik des Produktionssystems muss gewährleistet sein, dass die intelligenten Produkte auf ihrem individuellen Weg durch das Produktionssystem befördert werden können um von den, produktspezifisch, benötigten Bearbeitungsmodulen bearbeitet werden zu können. Um insgesamt die Forderungen nach Flexibilität und Rekonfigurierbarkeit erfüllen zu können sollte der Verbund der Logistikmodule seine Transportfunktionalität in Form eines dezentral koordinierten Intralogistiksystems zur Verfügung stellen. Die hierfür notwendige Architektur sollte diese Funktionalität ohne menschlichen Eingriff und auch nach einer Rekonfiguration zur Verfügung stellen.

1.2 Architekturen für flexible Produktionssysteme in der Forschung

Um die steigenden Flexibilitätsanforderungen zu adressieren wurden in den letzten Jahren einige Konzepte und die daraus resultierenden Architekturen präsentiert. Agentenbasierte Produktionssysteme sollen diesen Anforderungen gerecht werden (Faul et al. 2018). Des Weiteren wurde in (Regulin und Vogel-Heuser 2017) und (Pantförder et al. 2017) die Architektur eines agentenbasierten Produktionssystems präsentiert, welches die Produktion und den Transport auch über Unternehmensgrenzen hinweg koordiniert. Evaluiert wurde diese Architektur z. B. durch verschiedene Demonstratoren wie den „My-Joghurt“- bzw. den „RIAN“-Demonstrator.

In (Hompel 2006) wird eine erhöhte Flexibilität und Adaptivität erreicht indem, basierend auf dem Paradigma der SOA, das Konzept für eine Steuerungsarchitektur vorgestellt wird um den Materialfluss von Paketen zu organisieren. Insbesondere wird dabei die Vision einer „Zellularen Materialflusstechnik“ verfolgt welche sich, inspiriert von einem Nervensystem, organisch verhält wodurch die gesamte Steuerung während der Laufzeit entsteht.

Im Rahmen des H2020 PERFoRM-Projekts wurde eine Drei-Schichten-Architektur präsentiert, welche ebenfalls auf dem Paradigma der SOA basiert. Die Architektur enthält als Schlüsselelemente eine industrielle Middleware, Standardschnittstellen, Technologie-Adapter sowie integrierbare Tools, um eine nahtlose Integration der heterogenen Hardware-Geräte und der Software-Tools zu ermöglichen. Damit soll den Anforderungen an ein flexibles und rekonfigurierbares Produktionssystem begegnet werden (Hennecke und Ruskowski 2018).

2 Koordinierungskonzepte der dezentralen Intralogistik

In der zentralen Steuerung können alle steuerungstechnischen Abhängigkeiten zwischen den Modulen direkt im gelöst werden, in dem Steuerungscode programmiert wird, der alle Sensoren und Aktoren über Feldbusse ansteuert. Für dezentrale Lösungen muss hingegen ein Koordinierungskonzept realisiert werden, da die Modulsteuerungen über die Durchführung ihrer dedizierten Funktionalität hinaus koordiniert werden müssen um die gesamte Produktionssequenz durchführen zu können. Für die Koordinierung sind ein Kommunikationsnetzwerk, sowie ein entsprechendes Protokoll erforderlich.

2.1 Anforderungen an die Koordinierung der Intralogistik

Folgende Anforderungen werden an eine Koordinierung der dezentralen Intralogistik gestellt und in den folgenden Absätzen erläutert:

- Heterogenität
- Flexibilität
- Rekonfigurierbarkeit
- Echtzeitfähigkeit

Zudem müssen Softwaresysteme heute so strukturiert werden, dass eine Erweiterung und spätere Wartung mit einem überschaubaren Aufwand durchführbar bleiben.

2.1.1 Heterogenität

Das Koordinierungskonzept muss berücksichtigen, dass die modulare Anlagensteuerung heterogen aufgebaut sein kann. Sensoren, Aktoren und Steuergeräte, die im Netzwerk kommunizieren, werden in der Regel von unterschiedlichen Herstellern geliefert und laufen entsprechend auf unterschiedlicher Hardware (Kagermann et al. 2016).

Je nach Anforderung an Datenrate, Echtzeitfähigkeit und Rechenleistung muss das Konzept der Koordinierung auf unterschiedlichen Bussystemen für die Kommunikation und Hardwareplattformen für die Steuerung der Module anwendbar sein.

2.1.2 Flexibilität

Im Kontext Industrie 4.0 spielt Flexibilität bis hin zur Individualproduktion eine große Rolle (Pantförder et al. 2017). Die Flexibilität beschreibt dabei die Fähigkeit eines Produktionssystems seinen In- oder Output zu verändern ohne dass das System selbst verändert wird (Zaeh et al. 2006). Die so umfasste Funktionalität kann auch als Flexibilitätskorridor bezeichnet werden. Entsprechend muss die Koordinierung der Logistikmodule mit dieser Flexibilität umgehen können. Individuell produzierte Produkte nehmen unter Umständen unterschiedliche Transportwege durch die Anlage und halten an unterschiedlichen Bearbeitungsmodulen, um dort jeweils bearbeitet zu werden.

2.1.3 Rekonfigurierbarkeit

Die Rekonfiguration wird immer dann notwendig, wenn neue Produktionsaufträge nicht im Flexibilitätskorridor der Anlage liegen. Der Erfolg eines rekonfigurierbaren Produktionssystems (RMS) misst sich vor allem am Aufwand den ein Rekonfigurationsvorgang verursacht und dem daraus resultierenden Nutzen (Hees 2017; Stehle und Heisel 2017). Entsprechend sollte die Intralogistik der Anlage einfach rekonfigurierbar sein. Optimal wäre ein Plug&Produce-Konzept, bei dem einfach neue Logistikmodule ergänzt, entfernt oder in der Position in der Anlage verändert werden können.

2.1.4 Echtzeitfähigkeit

Echtzeit wurde in der DIN 44300 wie folgt definiert:

> „*Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.*“

Echtzeit-Bussysteme können die Anforderungen an eine Kommunikation im Bereich weniger Millisekunden erfüllen. Das deterministische Verhalten dieses Kommunikationsprozesses ist eine wichtige Voraussetzung für die Ausführung von regelungstechnischen Abläufen (Regulin und Vogel-Heuser 2017).

Die Koordinierung der Logistikmodule innerhalb einer Anlage muss in Echtzeit erfolgen. Insbesondere bei Produktionsanlagen mit einer hohen Taktung der Werkstücke müssen die Intralogistikmodule sehr schnell und präzise mit den wertschöpfenden Modulen kooperieren. Aber auch die Koordinierung zwischen Intralogistikmodulen muss echtzeitfähig umgesetzt werden, sodass Werkstücke schnell zu Modulen transportiert, abtransportiert und sortiert werden können.

Beispiele für Intralogistiksysteme mit hohen Echtzeitanforderungen sind automatisierte Logistikzentren, Gepäcksysteme bei Flughäfen und Produktionsanlagen mit hohen Taktraten.

2.2 Koordinierungskonzepte für die Intralogistik

Für die Koordinierung verteilter Funktionalität in der Automatisierungstechnik werden im Bereich der Forschung gegenwärtig folgende Paradigmen diskutiert (Bloch et al. 2017; Caridi und Cavalieri 2004; Cucinotta et al. 2009; Ribeiro et al. 2008):

- SOA
- Softwareagenten
- Microservices

Da die Koordinierungsmechanismen zusätzliche Softwarefunktionalität beinhalten steigt die Komplexität des Gesamtsystems (Geisberger und Broy 2012; Msadek et al. 2015) . Die erhöhte Komplexität der Software wirkt sich negativ auf Entwicklungsaufwand, Rekonfigurierbarkeit und Wartbarkeit aus. Die nach außen wahrgenommene Komplexität kann durch Kapselung von Funktionalität und durch die Verwendung von einfachen und stabilen Schnittstellen verringert werden.

Die hier diskutierten Konzepte werden alle in ISO/OSI-Layer 7, der Applikation umgesetzt. Das bedeutet, dass sie auf unterschiedliche Kommunikationsstacks aufgesetzt werden können und zunächst unabhängig vom eingesetzten Kommunikationsmedium sind. Bezogen auf die Heterogenität ist wichtig, dass die Hersteller der Komponenten, die zusammengestellt werden, Schnittstellen für gängige und Standardisierte Bussysteme anbieten. Der Anlagenbauer muss dann, wenn er die spezifische Automatisierungslösung für seine Anlage entwickelt innerhalb der Applikation die Koordinierung auflösen. Dies geschieht in der Regel nicht direkt in der Applikationslogik, sondern über ein zusätzliches Protokoll oder eine Middleware, die auf dem Kommunikations-Stack aufgesetzt werden. Gängige Protokolle für Software-Agenten und SOAs sind immer in der Applikations-Schicht angesiedelt und übernehmen die Koordinierungsaufgaben für die Anwendung, sodass sich der Automatisierungsingenieur auf die Applikationslogik der Automatisierungsfunktionen konzentrieren kann.

2.2.1 Verteilung von Funktionalitäten in der Software

Statische Abhängigkeiten bedeutet, dass Informationen, die für ein anderes Modul relevant sind über ein Kommunikationsmedium gesendet werden. Die Reaktion darauf kann dann im anderen Modul implementiert werden (Abb. 1).

Die Norm IEC 61499 definiert ein Modell für die Verteilung von Funktionsbausteinen auf mehrere Steuergeräte. Sie ermöglicht eine zunächst hardwareunabhängige Entwicklung der Funktionsbausteine (FB) und ein späteres Mapping auf unterschiedliche Rechenressourcen. Die Abbildung zeigt wie die Verteilung von Funktionalität auf unterschiedliche Steuergeräte in IEC 61499 vorgesehen ist. Die Schnittstellen zwischen den Funktionsbausteinen (FB) können gemeinsame Daten oder Funktionsaufrufe sein. Im Falle von Anwendung B werden diese Informationen über das Kommunikationsinterface und ein Bussystem kommuniziert.

Eine Änderung der Signale ist immer mit großem Aufwand verbunden, da sich Änderungen auf Sender, Empfänger und auf die Bus-Auslegung auswirken. Da alle Signalwege statisch ausgelegt werden, lässt sich das Systemverhalten sehr gut vorhersagen. Es ist also eine sehr gute Echtzeitfähigkeit gewährleistet. Aufgrund des hohen Aufwandes für Änderungen an der Kommunikation ist dieses Konzept nicht geeignet, um eine hohe Flexibilität und Rekonfigurierbarkeit zu gewährleisten.

Eine Kapselung von Funktionalität ist zunächst nicht vorgesehen. Die Abhängigkeiten werden klassisch in der Applikation aufgelöst. Die Sensor- und Aktor-Signale durchlaufen die Prozessinterfaces. Je nachdem, auf welchem Gerät die Steuerung gerechnet wird, werden entweder Sensor-Signale oder Stellgrößen für die Aktoren über den Bus gesendet.

Die weiteren Koordinierungsmechanismen kapseln Funktionalität als Services. Diese können in der Applikations-Schicht als Wrapper umgesetzt werden oder als zusätzliche Middleware für Applikationen in den Geräten umgesetzt werden. Service- oder agentenbasierte Koordinierungskonzepte können so durch Kapselung und Spezifikation der FB-Schnittstellen IEC 61499-konform umgesetzt werden.

2.2.2 Softwareagenten

Softwareagenten sind ein Konzept, welches für die Koordinierung verteilter Funktionalität eingesetzt werden kann. Die Funktionalität wird dabei von Agenten gekapselt, die auf einer Agentenplattform kommunizieren. So kann die Funktionalität im Agentennetzwerk angeboten werden (Siehe Abb. 2). Weitere Agenten können als

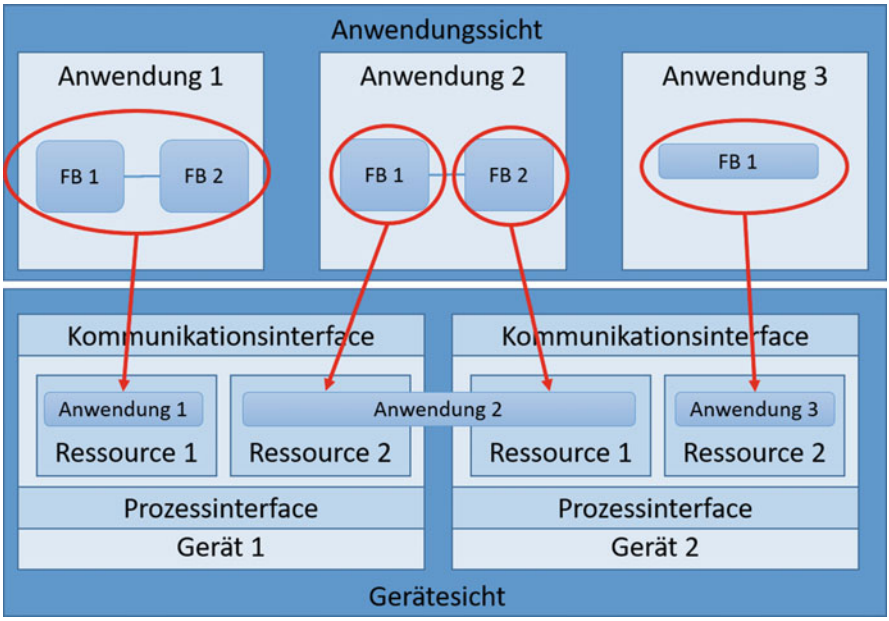


Abb. 1 Verteilung von Funktionsbausteinen nach IEC 61499

Konsumenten auftreten und die Funktionen nutzen. So können ähnliche Mechanismen realisiert werden, wie sie bereits in Abschn. 2.2.1 beschrieben wurden. Ein Produktagent kann für das Produkt, das er vertritt, individuell Funktionalitäten der Anlage, durch Kommunikation mit den entsprechenden Agenten, abrufen.

Hilfreich sind hierbei die Eigenschaften, die Softwareagenten zugeschrieben werden (VDI 2653, Blatt 1). Diese Eigenschaften zeigen bereits, dass Softwareagenten das Potenzial bieten deutlich mehr Intelligenz umsetzen zu können, als für die einfache Koordinierung der Module notwendig ist.

- Autonomie
- Interaktion
- Kapselung
- Persistenz
- Reaktivität
- Zielorientierung

So eignen sich Agentensysteme insbesondere dafür Mechanismen für die Selbstorganisation und Selbstoptimierung wertschöpfender Systeme zu realisieren (Vogel-Heuser et al. 2019). Die für die Koordinierung geforderte Flexibilität, Rekonfigurierbarkeit sowie der Umgang mit Heterogenität kann mit Hilfe eines Agentensystems realisiert werden. Für eine echtzeitfähige dezentrale Anlagensteuerung ist es allerdings nicht wünschenswert, dass Agenten eigenständig entscheiden ob, wann und wie lange sie rechnen.

Harte Echtzeitfähigkeit ist daher für Agenten nur unter Beschränkung ihres Kommunikationsverhaltens realisierbar (Fay und Wassermann 2018). Hierzu ist es möglich rein reaktive Agenten einzusetzen. So kann ein echtzeitfähiges Agentennetzwerk aufgebaut werden. Das

2.2.3 Microservices

Microservices sind ein servicebasiertes Konzept zur Modularisierung von Applikationen. Sie haben ihren Ursprung im Bereich der Web-Services. Das Ziel beim Einsatz von Microservices ist eine hohe Skalierbarkeit. So können Rechen- und Speicherbedarf durch Instanziierung neuer Services dynamisch angepasst werden (Abb. 3).

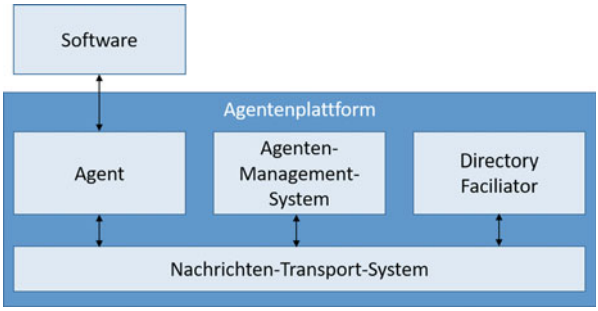


Abb. 2 Agentensystem (nach FIPA 2004)

Microservices basieren auf drei Grundideen:

1. Ein Programm soll nur eine Aufgabe erfüllen
2. Die Programme sollen zusammenarbeiten können
3. Programme sollen eine universelle Schnittstelle nutzen.

Damit stellen Microservices ein Modularisierungskonzept für große Systeme dar. Das besondere an Microservices im Vergleich zu anderen Modularisierungskonzepten ist, dass jeder Service unabhängig von anderen in Produktion gebracht werden kann. Wenn für eine Aufgabe, die ein Microservice eine bessere Implementierung oder eine bessere Technologie bereitstehen, dann dieser Service unabhängig von den anderen neu ausgeliefert werden. Die unabhängige Auslieferung der Microservices bedeutet, dass sie als virtuelle Maschinen oder unabhängige Applikationen umgesetzt werden müssen. Nur dann sind die Services als eigenständige Programme lieferbar (Wolff 2016).

Gegenwärtig werden Microservices als Architektur-Konzept für die Automatisierung von modularen Prozess-Anlagen diskutiert (Bloch et al. 2017).

Microservices gliedern sich in Functional-Services, die wiederum auf Infrastructure-Services zugreifen. So kann nicht nur die Software, sondern auch die notwendige Hardware dynamisch skaliert werden. Dies ist im Falle von Rechenleistung oder Speicherbereich auf Servern vergleichbar einfach möglich. Im Falle der Automatisierungstechnik bedeutet diese Skalierung allerdings eine Hardware-Rekonfiguration, die gegenwärtig nicht vollständig automatisiert werden kann. Grundsätzlich könnte eine Linienproduktion so durch Instanziierung von Microservices zwar, was die Software anbelangt mit geringem Aufwand skaliert werden. Aufgrund des geringen Automatisierungsgrades der Hardware-Rekonfigurationsmaßnahmen wird der große Vorteil der Micro-Services in dieser Domäne allerdings stark eingeschränkt.

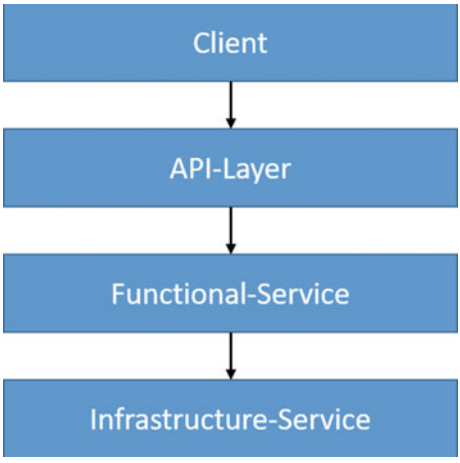


Abb. 3 Micro-Services

2.2.4 Service-orientierte Architekturen

Die SOA haben ihren Ursprung im Bereich der Geschäftsprozesse und stellen ein Paradigma zur Koordinierung verteilter Funktionalität dar (MacKenzie et al. 2006). Funktionalität wird dabei in Services gekapselt und im Netzwerk den Service-Konsumenten bereitgestellt. Services und Konsumenten finden sich durch den Discovery-Server, der die entsprechenden Meta-Daten kennt und entsprechend Abb. 4 ein Serviceverzeichnis darstellt.

Durch den Einsatz einer SOA wird eine Ad-hoc-Vernetzung zur Laufzeit im Sinne des Plug & Produce-Gedanken ermöglicht. Besonderheit der SOA sind die wohldefinierten Schnittstellen der Services. Wohldefiniertheit bedeutet im Software-Kontext, dass die Schnittstellen nicht nur syntaktisch, sondern auch semantisch definiert sind. Diese Schnittstellenspezifikation dient einer Kompatibilität zwischen allen Komponenten, die das entsprechende SOA-Protokoll beherrschen. Eine Registrierung der Services beim Eintreten in das Netzwerk kann so realisiert werden, dass die Verwaltung der Services im Discovery Server vollständig automatisiert werden kann. Position in der Anlage, Fähigkeiten des Moduls und weitere Metadaten müssen so nur einmal im Service konfiguriert werden. Die Rekonfiguration wird softwareseitig also maximal unterstützt, was zu einer hohen Rekonfigurierbarkeit des Gesamtsystems führt.

Eine hohe Flexibilität wird durch das Service-Konsument-Konzept erreicht. Ein Produkt, das sich durch die Anlage von Bearbeitungsmodul zu Bearbeitungsmodul bewegt kann mit Intelligenz ausgestattet werden und im SOA-Netzwerk als Konsument auftreten. So kann jedes Produkt individuelle Wege und Bearbeitungsschritte in Anspruch nehmen.

2.2.5 Gegenüberstellung der Koordinationskonzepte

In der nachfolgenden Tabelle werden die Konzepte für die Koordinierung gegenübergestellt (Tab. 1).

Die statische Koordinierung ist nicht flexibel und rekonfigurierbar, da die Signale bei jeder Änderung bei Sender und Empfänger angepasst werden müssen. Eine Entkopplung findet nicht statt.

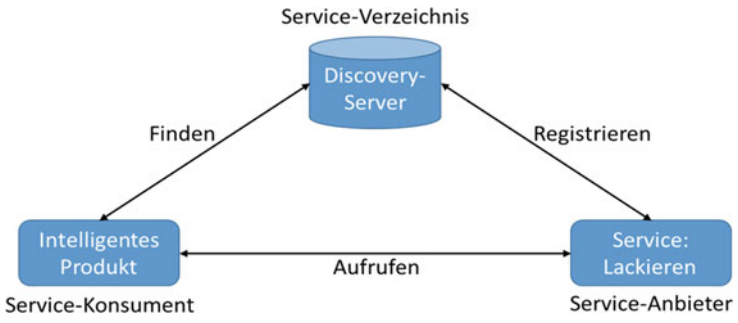


Abb. 4 SOA-Mechanismen

Microservices und SOA sind beide servicebasierte Paradigmen, haben jedoch fundamentale Unterschiede. Microservices gliedern ein Programm in Module, die über ihre API zu einem Programm zusammengeführt werden. Die SOA stellt eine Architektur dar, die eine flexible Koordinierung verteilter Funktionalität bietet. Damit befinden sich Microservices und die SOA auf unterschiedlichen Ebenen, nämlich innerhalb eines Programmes und zwischen Programmen. Da Microservices unabhängig in Produktion gebracht werden sollen, werden sie häufig in Form von virtuellen Maschinen umgesetzt. Alternativ wären Docker-Container-Konzepte in Java denkbar. Die Anwendung von Microservices im Embedded-Umfeld für die Automatisierungstechnik ist daher nur über eine Java-Laufzeitumgebung denkbar. Daher ist die Echtzeitfähigkeit bei Microservices nur bedingt gegeben. Die Flexibilität ist nur im vordefinierten Flexibilitätskorridor gegeben. Microservices unterstützten keine Rekonfigurierbarkeit, da einzelne Services zwar ausgetauscht werden können, diese dann aber nur neue Technologien oder Implementierungen enthalten, jedoch keine neuen Features. Die ersetzten Services müssen über die gleichen Funktionen und Schnittstellen verfügen.

Die SOA betrachtet im Gegensatz zu den Microservices die Koordinierung zwischen den Funktionen. Der Discovery-Server ermöglicht für jeden Client eine dynamische, individuelle Zuordnung von Services. Die Echtzeitfähigkeit ist umsetzbar, indem die Services auf eingebetteten Steuergeräten ihre jeweilige Automatisierungsfunktion umsetzten. Die Koordinierung erfolgt auf ISO/OSI-Layer 7 und muss auf echtzeitfähigen Kommunikationsprotokollen aufsetzen. Da alle Services über die gleichen Schnittstellen aufgerufen werden, ist es sehr einfach Services mit neuen Features in das Netzwerk einzubinden oder auszutauschen. So ist eine hohe Rekonfigurierbarkeit gegeben.

Wird ein Agentensystem echtzeitfähig und reaktiv eingesetzt, unterscheidet es sich nach außen hin nicht mehr von einer SOA. Tatsächlich ist es so möglich mit einem Agenten-Framework eine SOA zu realisieren.

Aus der Gegenüberstellung folgt, dass sich die SOA für die Koordinierung der verteilten Anlagensteuerung am besten eignet. Daher wird im Folgenden die Entwicklung einer SOA für die verteilte Anlagensteuerung inklusive der Steuerung der Intralogistik betrachtet.

Tab. 1 Koordinierungskonzepte

	Echtzeit-Fähigkeit	Heterogenität	Flexibilität	Rekonfigurierbarkeit
Statische Koordinierung	++	0	–	–
Software-Agenten	– bis +	++	++	++
Micro-Services	0	0	+	0
SOA	+	++	++	++

3 Orchestrierung der dezentralen Intralogistik

3.1 Service-Hierarchie der Organisationsstruktur

Die Architektur der dezentralen Intralogistik wurde derart konzipiert, dass sich das Intralogistiksystem für die Koordination der Logistikprozesse als ein unabhängig agierendes Teilsystem des Produktionssystems verhält. Es stellt den restlichen Bestandteilen des Produktionssystems die Funktionalität dieser Logistikprozesse in Form von aufrufbaren Services zur Verfügung. Deshalb wird nun zunächst die in (Schmidt et al. 2018) vorgestellte hierarchisch aufgebaute Organisationsstruktur eingeführt. Diese wurde für die service-orientierte Steuerung von modularen Produktionssystemen (MPS) entwickelt und ist in Abb. 5 dargestellt.

Dieser Organisationsstruktur liegt die Entkopplung von Produkt, Prozess und der benötigten Intralogistik zugrunde. Die Entkopplung wird durch die Einführung der drei Layer „Produkt Layer“, „Prozess Layer“ und „Logistik Layer“ erreicht.

Dem „Produkt Layer“ werden hierbei die zu produzierenden intelligenten Produkte zugeordnet.

Ein intelligentes Produkt ist ein gefertigter Gegenstand, der mit der Fähigkeit ausgestattet ist, seinen gegenwärtigen oder zukünftigen Zustand zu überwachen, zu analysieren und zu bewerten und gegebenenfalls sein Ziel zu beeinflussen (McFarlane et al. 2002).

Das intelligente Produkt ist also eine Software, die ein physisches Produkt vertritt und aufgrund seiner Fähigkeiten dessen Produktion vertreten kann. Das intelligente Produkt kennt dabei alle zur Produktion notwendigen Prozessschritte und den aktuellen Zustand des physischen Produkts (Werkstück). Außerdem kann das intelligente Produkt in der SOA mit den Bearbeitungsmodulen kommunizieren, um deren Services in Anspruch zu nehmen. So kann das intelligente Produkt die Koordinierung der Produktion seines Produkts übernehmen. Jedes Produkt wird von einem eigenen, individuell modellierten intelligenten Produkt vertreten. So kann eine Individualproduktion erreicht werden.

Dem „Prozess Layer“ werden die Bearbeitungsservices zugeordnet, welche die einzelnen Prozessschritte, der für die intelligenten Produkte benötigten Produktionssequenzen, anbieten. Bearbeitungsservices können nicht nur klassische Prozessschritte, wie Drehen, Bohren oder Fräsen sein, sondern auch Prozessschritte, die der Qualitätssicherung dienen und in die Produktionssequenz der Produkte eingeplant werden können.

Dem „Logistik Layer“ werden entsprechend die Logistikservices, welche für die Transportaufgaben der Intralogistik zuständig sind, zugeordnet. Die Logistik wird vollständig dezentral aufgebaut und organisiert, um eine hohe Flexibilität und Rekonfigurierbarkeit zu gewährleisten. Logistikmodule bewegen die Werkstücke in der Anlage. Dazu gehören Förderbänder, fahrerlose Transportsysteme oder Roboterarme, die neben Bearbeitungs- auch Logistikaufgaben übernehmen können. Das „Logistik Layer“ umfasst somit alle Bestandteile des Intralogistiksystems und verwendet den Discovery Server um dem Restsystem seine Funktionalität anbieten zu können.

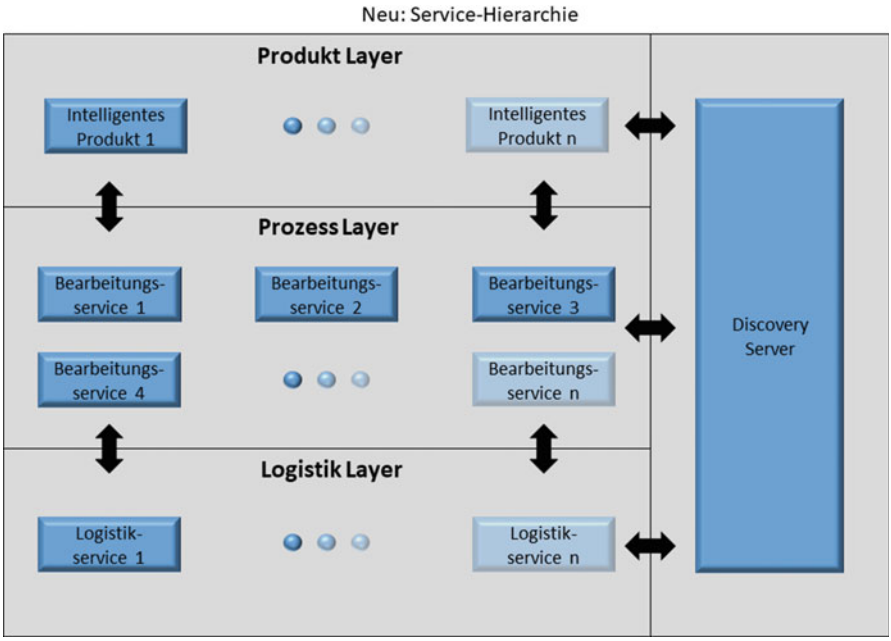


Abb. 5 Service-Hierarchie

Die Kommunikation mit dem Discovery Server ist von allen drei Layern aus möglich und dient dem Auffinden der jeweils benötigten Services und dem Abfragen von Informationen, um mit diesen Services kommunizieren zu können.

Der Discovery Server, stellt hier zunächst betrachtet einen Single Point of Failure dar. Während der Ausfall eines Logistikmoduls in der SOA bei redundant ausgelegten Transportwegen durch neue Service-Calls ein anderer Weg gewählt werden kann, muss beim Discovery Server eine hohe Verfügbarkeit gewährleistet sein. Ist dies durch eine einzelne Netzwerkkomponente nicht möglich, muss der Discovery Server redundant ausgelegt werden.

3.2 Koordinations-Ablauf

Der Ablauf der Koordination zur Durchführung eines Prozessschrittes lässt sich wie folgt beschreiben:

Die *intelligenten Produkte* kennen die benötigten Prozessschritte und die Reihenfolge, in welcher diese benötigt werden entsprechend ihrer Bill of Process (BOP). Somit können sie mit Hilfe des Discovery Servers den jeweils benötigten Service des Bearbeitungsmoduls aufrufen, der diesen Prozessschritt anbietet.

Wird der Service eines *Bearbeitungsmoduls* aufgerufen und von diesem akzeptiert und bestätigt, so ruft das Bearbeitungsmodul das geeignete Logistikmodul auf, welches dann den Transport des Produkts zum Bearbeitungsmodul organisiert. Dies

geschieht wiederum unter Zuhilfenahme des Discovery Servers, welcher dem Bearbeitungsmodul das geeignete Logistikmodul vermittelt.

Sobald der Transportservice eines Logistikmoduls aufgerufen wird, agiert der Logistik Layer als dezentrales Intralogistiksystem. Der Transport des Produkts von seiner aktuellen Position zum aufrufenden Bearbeitungsmodul wird innerhalb dieses Layers organisiert. Dies geschieht, falls nötig unter Zuhilfenahme der Services von weiteren Logistikmodulen. In Abschn. 3.3 werden verschiedene Möglichkeiten zur Wegfindung für typische Layouts von Produktionssystemen genauer beschrieben.

So wird nicht nur die Bearbeitung dezentral koordiniert, stattdessen ist auch die Intralogistik nicht mehr nur eine monolithische Logik die zentral Wegfindungen berechnet, sondern jedes Logistikmodul kann durch sein Wissen und seine Fähigkeiten zur gesamten Intralogistik beitragen. Dabei benötigt das Logistikmodul Wissen über folgende Aspekte:

1. Welche Werkstücke kann ich aktuell transportieren?
2. Welche weiteren Logistikmodule können mir Werkstücke zuliefern?

Die Informationen über aktuell transportierbare Werkstücke ist notwendig um entsprechende Transportservices abzuarbeiten. Das Wissen über Logistikmodule, die zuliefern können ist notwendig, um Transportwege über mehrere Module hinweg realisieren zu können. Da die Transportservice-Beauftragung vom Bearbeitungsmodul, also dem Transportziel ausgeht, werden von dort aus bis zum Standort des Werkstücks mögliche Wege aufgespannt. Daher ist es für die Logistikmodule nicht notwendig, nachfolgende Module zu kennen. Diese werden die Services des Moduls gegebenenfalls aufrufen.

3.3 Realisierung einer Wegfindung mit einer Service-orientierten Architektur

3.3.1 Beschreibung von Intralogistik-Layouts

Die Realisierung der Wegfindung durch die Anlage wird maßgeblich durch das zugrunde liegende Layout des Produktionssystems beeinflusst. Die wohl einfachste Layout-Form stellt die Linienproduktion dar. In dieser Variante gibt es keine redundanten Transportwege oder Abzweigungen. Trotzdem muss sich die Intralogistik an die individuellen Produkte anpassen, da die Bearbeitungsstationen für jedes Produkt variieren können.

Für die Beschreibung der Wegfindung eines Werkstücks durch die Anlage werden hier zunächst typische Anlagenlayouts (Smriti 2014) vorgestellt:

In der *Linienproduktion* gibt es einen fest vorgegebenen Weg durch die Anlage. Das Werkstück bewegt sich entlang der Linie. Eine Flexible Produktion ist in diesem Fall auch ohne redundante oder alternative Wege möglich, indem die Produkte an unterschiedlichen Bearbeitungsmodulen halten oder an einem Modul produktspezifisch unterschiedliche oder unterschiedlich parametrierte Services abgerufen werden. Die Intralogistik, die die Werkstücke bewegt, muss auch in diesem Fall indivi-

duell auf jedes Produkt reagieren, da die Produkte an unterschiedlichen Stellen halten (Abb. 6).

Eine Alternative stellt das *Functional Layout* dar. In diesem Layout haben die Bearbeitungsmodule feste Positionen in der Anlage. Die Werkstücke bewegen sich allerdings nicht entlang eines fest definierten Weges, sondern müssen je nach Produktionssequenz in unterschiedlicher Reihenfolge unterschiedliche Module anfahren. Die Positionen sind in diesem Szenario häufig in Form einer Matrix angeordnet. Herausforderung hierfür sind für die Intralogistik nicht mehr nur individuelle Stopps, sondern auch die Wegfindung der Werkstücke jeweils zum nächsten Modul. Diese muss dynamisch für jeden Schritt berechnet werden, da sich der Anlagenzustand mit den Belegungen der Module durch Werkstücke und die Positionen der Werkstücke ständig ändern (Abb. 7).

Die dritte Variante ist das *Fixed Position Layout*. In dieser Variante ist die Position des Produkts fixiert und Material, Werker und Bearbeitungsmaschinen werden zum Produkt transportiert. Da in diesem Szenario die Produkte nicht bewegt werden und es nach (Smriti 2014) mit einem hohen Personalaufwand verbunden ist und nur einen geringen Automatisierungsgrad aufweist, wird dieses Layout im Beitrag nicht weiter betrachtet.

Die vierte Variante ist eine *Kombination der drei vorher beschriebenen Layouts*. Häufig wird eine Linienproduktion mehrfach aufgebaut, was zu einer Matrix führt, die gleiche, parallele Strukturen enthält. Die Linienproduktion bietet aufgrund der Kombinatorik von Bearbeitungsservices bereits ein gewisses Flexibilitätsspektrum. Durch die redundante Linie wird neben einer Skalierung der Produktion auch Redundanz und Flexibilität verbessert.

Abb. 8 zeigt die Mischform von Functional Layout und Linie. Die Produktionssequenz der Linie (blau, rot, grün) wird dreimal umgesetzt. Querverbindungen ermöglichen den Wechsel von einer Linie zur anderen.

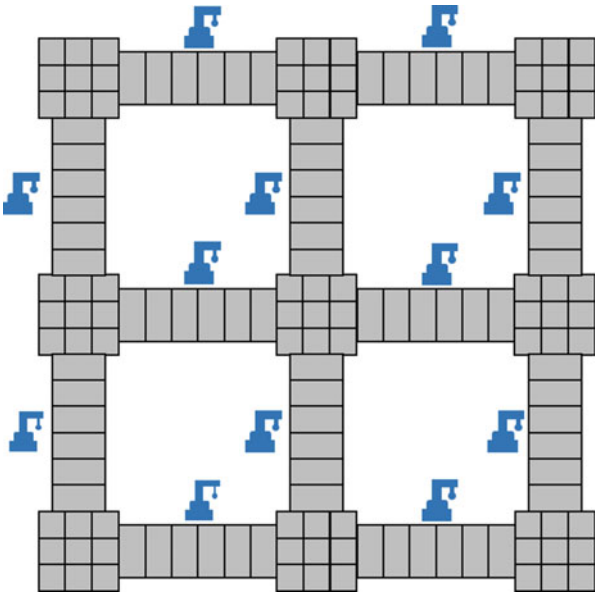
3.3.2 Wegfindung in Linie und Matrix

Die einfache Variante der Wegfindung findet im Szenario der Linienproduktion statt. Der oben beschriebene Mechanismus zum Koordinationsablauf wird ausgeführt. Das Intelligente Produkt findet über den Discovery-Server das nächste Bearbeitungsmodul und ruft dessen Service auf (1) (Abb. 9).

Das aufgerufene Modul ruft den Transportservice eines Logistikmoduls auf, das ihm Werkstücke zuliefern kann (2). Wenn dieses das aufrufende Werkstück nicht hat, ruft es den Service des Moduls auf, das ihm zuliefern kann (3). So werden die Services weiter aufgerufen, bis ein Logistikmodul das Werkstück hat (4). Die Services werden entsprechend gegenläufig zur Aufruf-Hierarchie abgearbeitet. In

Abb. 6 Linienproduktion

Abb. 7 Functional Layout



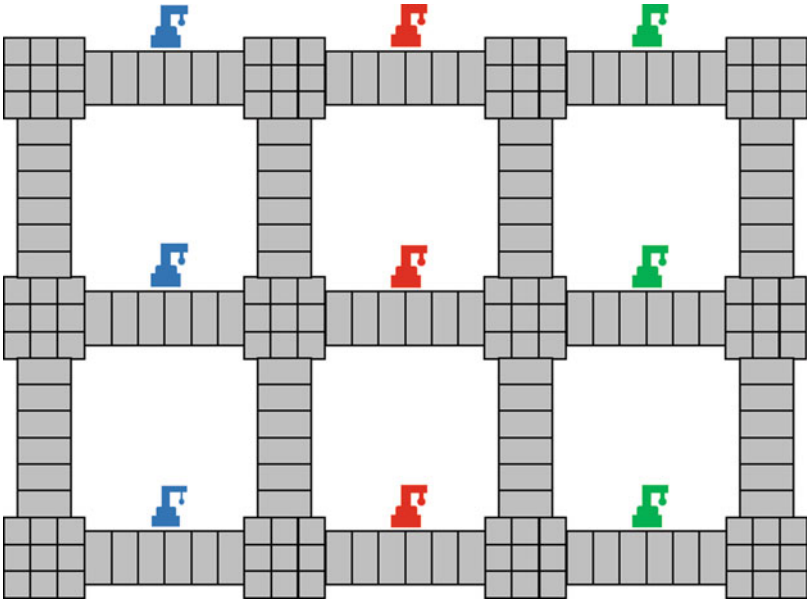


Abb. 8 Mischform, Redundante Linie

der Linienproduktion sind die Abhängigkeiten im Gegensatz zum Functional Layout einfach zu hinterlegen. Jedes Werkstück kann also gemäß seinem Produktionsplan

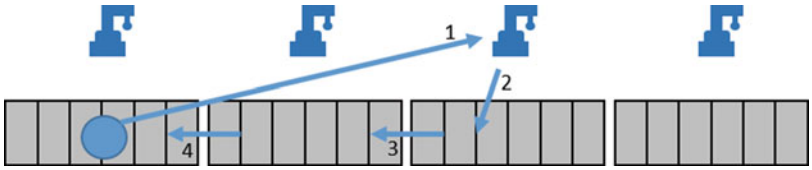


Abb. 9 Wegfindung in Linienproduktion

transportiert und bearbeitet werden. Aufgrund des Linien-Layouts gibt es keine Mehrfach-Aufrufe von Services.

Die gleichen Mechanismen können auch im Functional Layout eingesetzt werden. Um das Szenario zu beschreiben werden zunächst nur ein Produkt und ein Bearbeitungsmodul in einem allerdings vergleichsweise komplexen Layout betrachtet (Abb. 10).

Bis zum Aufruf eines Logistikservice läuft die gleiche Sequenz ab. Das Produkt ruft das Bearbeitungsmodul auf (1) und das Bearbeitungsmodul ruft das Logistikmodul auf (2). Anschließend folgt die Aufruf-Hierarchie grundsätzlich den gleichen Regeln, wie in der Linienproduktion. Die Komplexität steigt jedoch, da es nicht mehr nur ein Modul gibt, das zuliefern kann, sondern in der dargestellten Struktur bis zu drei. In diesem Fall gibt es drei Varianten die Service-Calls bis zum Produkt fortzusetzen:

1. Alle Logistikmodule, die einem Logistikmodul zuliefern können werden aufgerufen (3). Ein Netz aus möglichen Fahrwegen spannt sich vom Transportziel bis zur Position des Produkts auf. Wenn das Logistikmodul mit dem Werkstück beginnt den Service abzuarbeiten und den Service-Call diesbezüglich bestätigt, werden für den Transport unnötige, aber bereits getätigte Service-Anfragen im Netzwerk gelöscht. Da nur verfügbare Logistikmodule weiter aufrufen und Services bestätigen, werden so automatisch Wege gefunden, auch, wenn der direkte Weg gerade belegt oder blockiert ist, oder ein Modulausfall auftritt. Nachteil dieses Algorithmus ist ein relativ großer Kommunikations-Overhead. Der Overhead steigt überproportional mit einer höher skalierten Anlagengröße.
2. Der rudimentäre Algorithmus aus Variante 1 kann reduziert werden, indem die Logistikmodule mit zusätzlicher Intelligenz ausgestattet werden. Wenn jedes Modul, auch die Service-Aufrufe für andere Module mithört und speichert, können Mehrfachaufrufe von Modulen (4) vermieden werden.
3. Die sicherlich eleganteste Variante ist jedoch ein Wegfindungsservice, der im Logistik Layer angesiedelt wird und die Anlagenstruktur, sowie die aktuelle Belegung der Module kennt. Logistikmodule, die für die Erfüllung ihres Transportauftrages auf andere Module angewiesen sind, können diesen Service anfragen und dann gezielt, wie bei der Linie jeweils einen Service aufrufen. Dieser Wegfindungsservice kann beliebig im Netzwerk verortet werden und muss mit Modellen über die Anlagenstruktur und Wissen über den aktuellen Anlagenzustand angereichert werden. Die Anlagenstruktur muss beim Anlagenbau einmalig modelliert und nach jeder Rekonfiguration aktualisiert werden. Die aktuelle

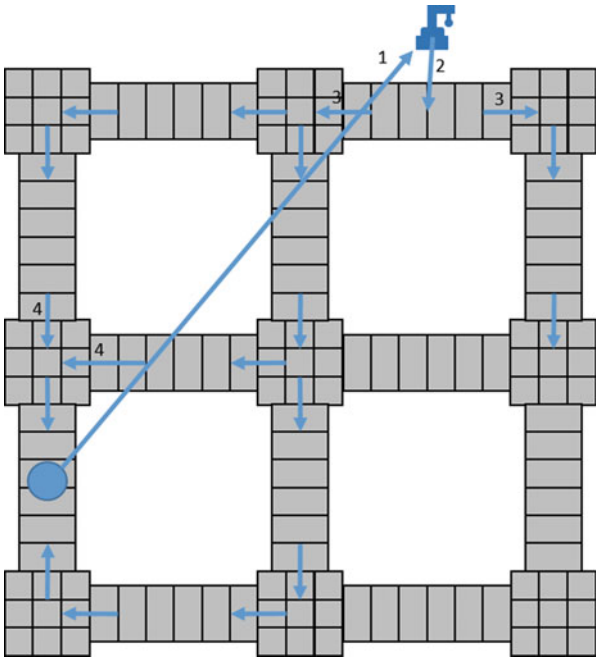


Abb. 10 Wegfindung im Functional Layout

Belegung der Module kann aus der Kommunikation zwischen den Modulen ermittelt werden, da Zustandsänderungen an der Anlage durch Abarbeitung von Services erfolgen und die erfolgreiche Abarbeitung im Netzwerk kommuniziert werden muss. Diese Variante kann ebenfalls für jeden Transportauftrag einen individuellen Weg finden und vermeidet den Kommunikationsoverhead. Nachteil ist der zusätzliche Entwicklungs- und Modellierungsaufwand für den Wegfindungsservice.

Alle drei Varianten führen zu einer dezentralen Intralogistik, die hochgradig automatisiert in dem Sinne agiert, dass weitere Eingriffe des Menschen oder von anderen technischen Systemen nicht notwendig sind.

3.4 Echtzeitfähiges SOA-Protokoll

Um die Kommunikation innerhalb eines Produktionssystems umzusetzen muss das resultierende Kommunikationsverhalten der service-orientierten Organisationsstruktur, des eingesetzten Kommunikationsprotokolls und der eingesetzten Kommunikationsmedien die bestehenden Echtzeitanforderungen erfüllen.

Für eine SOA müssen Syntax und Semantik spezifiziert sein. Diese werden nachfolgend beschrieben.

3.4.1 Spezifikation der Syntax

Für die Koordinierung wurde ein echtzeitfähiges Protokoll entwickelt. Es ist in ISO/OSI-Layer 7 angesiedelt und kann auf unterschiedliche Bus-Systeme aufgesetzt werden. Das Protokoll selbst ist sehr leichtgewichtig mit geringem Kommunikations-Overhead. Je nach Anforderungen an Antwortzeiten und Datenraten muss das SOA-Protokoll auf ein angemessenes Bus-System und somit Bus-Protokoll aufgesetzt werden.

Zunächst wurden notwendige Nachrichtentypen definiert mit den entsprechenden Daten.

Die Messages enthalten Nutzdaten, die in den Datenfeldern übertragen werden. Folgende Tabelle zeigt, welche Nutzdaten den Messages zugeordnet werden (Tab. 2):

Die weiteren Datenfelder im Service-Call können zur Parametrierung der Services genutzt werden.

Im folgenden Diagramm wird die Kommunikation im Falle eines Service-Aufrufs dargestellt. Die Registrierung und Fehlerfälle werden dabei nicht betrachtet (Abb. 11).

Zunächst wird das Bearbeitungsmodul mit der Bearbeitung des Produkts beauftragt. Wenn dieses den Auftrag akzeptiert, ruft es das Logistik-Modul auf, um den Transport des Werkstücks in die Wege zu leiten. Da ein Förderband unter Umständen mehrere Werkstücke transportiert, ist die Reaktion nicht nur ein *Answer_Service*, sondern der Status des Förderbands. Nach erfolgreichem Transport wird das Werkstück bearbeitet. Anschließend wird die Abarbeitung des Service dem intelligenten Produkt gemeldet.

3.4.2 Spezifikation der Semantik

Für eine SOA muss auch die Semantik spezifiziert werden. Das bedeutet, dass den Nachrichtentypen und Nutzdaten je eine Bedeutung gegeben werden muss. Nur kann zwischen den verschiedenen Geräten eine Interoperabilität dahingehend hergestellt werden, dass Geräte die Netzwerkbotschaften verstehen (Schiekofer et al. 2018). Diese müssen im SOA-Netzwerk der Anlage einheitlich verwendet werden. Daher ist es sinnvoll diese Daten im Discovery-Server zu verwalten, der sie den Modulen auf Abfrage bereitstellt.

3.4.3 Umsetzung des Protokolls am modularen Produktionssystem

Das beschriebene Protokoll wurde am MPS umgesetzt. Die 12 Mikrocontroller, die die Bearbeitungs- und Logistikmodule steuern unterstützen bereits den Controller Area Network (CAN)-Bus. Dieser deckt die ISO/OSI-Layer 1 und 2 ab. Die Layer 3 bis 6 sind in diesem Fall nicht notwendig. Durch Priorisierung der Messages kann der CAN-Bus echtzeitfähig ausgelegt werden (Abel et al. 2006).

Den Nachrichtentypen wurden Message-IDs zugeordnet. Nachrichten, die für die Koordinierung benötigt werden (z. B. Service-Call), bekommen dabei höhere Prioritäten als Nachrichten für Verwaltungszwecke (z. B. Registrierung).

Tab. 2 Protokoll

Nachrichten-Typ	Datenfeld 1	Datenfeld 2	Datenfeld 3	Datenfeld 4
Call_Service	Service-ID	Product-ID	Recall-Bit	
Answer_Service	Status-IP	Product-ID	Service-ID	
Get_Info_DS	Service-Type	Consumer-ID		
Info_from_DS	Service-ID	LM-ID	Consumer-ID	
Call_LM-Service	Service-ID	Product-ID	LM-ID	Consumer-ID
LM_Status	LM-Status	Service-ID	Product-ID	Consumer-ID
Register_Service	Service-ID	Module-ID	Service-Type	Position-ID
Unregister_Service	Service-ID	Module-ID		

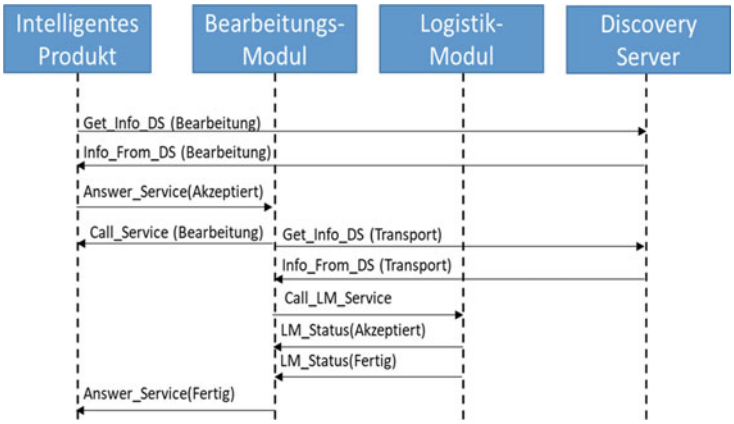


Abb. 11 Ablauf der Service-Calls

3.5 Automatisierungsgrad des Intralogistiksystems

Durch die dezentral organisierte, modulare Intralogistik wird im Vergleich zu monolithischen Logistiksystemen, neben der Individualproduktion, auch eine hohe Rekonfigurierbarkeit auf Modul- und Systemebene erreicht.

Ein hoher Automatisierungsgrad nach (Parasuraman et al. 2000) wird dadurch erreicht, dass auch nach einer Rekonfiguration keinerlei menschliche Eingriffe in die Steuerung der Intralogistik notwendig sind, da die Abhängigkeiten zwischen den Modulen durch die SOA aufgelöst werden.

So wird ein Intralogistiksystem orchestriert, das den Bearbeitungsmodulen, die zur Individualproduktion notwendigen Transportservices anbietet.

4 Umsetzung der service-orientierten Architektur mit einem Modellierungs-Framework

Die SOA stellt eine Architektur dar, die den Anforderungen an zukünftige Automatisierungssysteme gerecht wird. Daher lohnt sich eine Betrachtung der Entwicklungsprozesse, die notwendig sind, um eine SOA zu entwickeln.

Es lohnt sich in diesem Kontext das Paradigma der modellgetriebenen Entwicklung zu betrachten, das die Minimierung des Entwicklungsaufwandes durch maximale Automatisierung der Softwareentwicklung zum Ziel hat. Sie basiert auf einer vollständigen funktionalen Modellierung der Software, sodass Folgemodelle oder Code automatisiert generiert werden können. Gerade in Bezug auf die Rekonfigurierbarkeit, die den Aufwand im Rekonfigurationsfall in den Fokus stellt, bringt der optimierte Entwicklungsprozess neben der SOA weitere Vorteile.

Angelehnt an dieses Paradigma wird im Folgenden ein Schalenmodell vorgestellt, das die Modellierung der Software erleichtert, Codegenerierung für die verteilten Steuerungen ermöglicht und so den Entwicklungsprozess vereinfacht.

4.1 Das Modellierungsframework für die Services

Im disruptiven Umfeld zukünftiger Produktionssysteme werden Rekonfigurationen häufig vorkommen. In Bezug auf die SOA bedeutet das, dass ständig neue Services in die Anlage integriert oder bestehende Services angepasst werden müssen. Dies bezieht sich sowohl auf die Intralogistik, als auch auf die wertschöpfenden Bearbeitungsmodule.

Die Anforderungen, die in Abschn. 2 hergeleitet wurden, wurden in dem Schalenmodell berücksichtigt, sodass Automatisierungsfunktionen einfach modelliert und in einem Service gekapselt in das SOA Netzwerk integriert werden können (Abb. 12).

Im Folgenden werden die Schalen kurz erläutert.

Basissoftware-Shell:

Um die Heterogenität zu adressieren, werden alle hardware-abhängigen Teile des Modells in dieser der Basissoftware-Shell (BSW) modelliert. Dies ermöglicht eine einfache Wiederverwendung der weiteren Modelle und einen einfachen Umzug auf eine andere Hardware. Support-Packages gewährleisten einen einfachen Hardware-Zugriff aus den Modellen heraus und eine einfache Modellierung der Signalflüsse, bzw. eine Konfiguration entsprechender Bausteine im Modell.

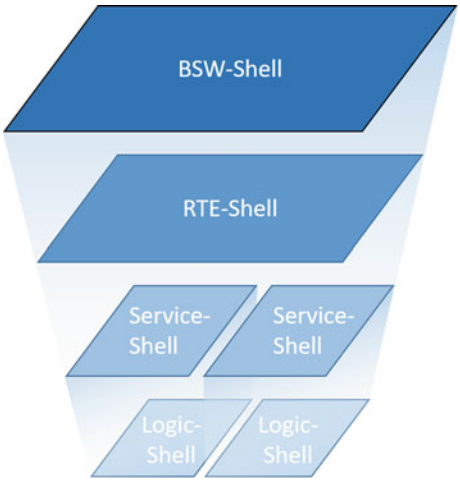
RTE-Shell:

Die Laufzeitumgebung (RTE) ist notwendig, da ein Modul unter Umständen mehrere Services anbieten kann. Die notwendigen Signalflüsse werden in der RTE zugeordnet. Im RTE-Modell werden die Services gekapselt. So können Services, die von mehreren Modulen verwendet werden wiederverwendet werden.

Service-Shell:

Die Service-Shell kapselt die Logic-Shell, die die eigentliche Applikationslogik enthält. In der Service-Shell werden die SOA-Mechanismen und das SOA-Protokoll

Abb. 12 Schalenmodell



umgesetzt. Service-Aufrufe werden hier verarbeitet und beantwortet. Die Service-Shell kann mit wenigen Parametern konfiguriert und, wenn sie einmal modelliert wurde, für alle Module wiederverwendet werden.

Logic-Shell:

Die Logic-Shell enthält die eigentliche Applikationslogik, die der Automatisierungsingenieur modelliert. Hier findet die Modellierung der Automatisierungsfunktion statt. Die Kapselung in der Logic-Shell ermöglicht eine hohe Wiederverwendung der Applikationen, sowie eine Fokussierung des Entwicklungsingenieurs auf die Applikation.

4.2 Verwendung des Frameworks

Das so entwickelte Schalenmodell kann als Framework für die Entwicklung verteilter SOA-Steuerungen verwendet werden. Für die BSW-Shell, muss ein Supportpackage für die Zielhardware mit der zugehörigen Toolchain einmalig in Betrieb genommen werden, um eine Konfiguration der Hardware im Modell zu ermöglichen. RTE- und Service-Shell müssen für jedes Modul konfiguriert werden. Lediglich die Applikationen müssen individuell modelliert werden.

Das Ergebnis sind dann vollständige und funktionale Modelle, aus denen der Code für die gesamte Anlagensteuerung generiert werden kann.

5 Evaluation der service-orientierten Architektur

5.1 Realisierung der SOA im Labor-Maßstab

Um Industrie 4.0-Konzepte, welche die oben genannten Anforderungen erfüllen sollen, evaluieren zu können wird am Institut für Automatisierungstechnik und Softwaresysteme (IAS) ein MPS im Labormaßstab eingesetzt.

Das MPS besteht aus den in Abb. 13 dargestellten 8 Bearbeitungsmodulen und 4 Logistikmodulen. Die Steuerung der Anlage wurde zunächst klassisch mit einer speicherprogrammierbaren Steuerung (SPS) realisiert, die über Feldbusse und Bus-Koppler auf die Sensoren und Aktoren der Module zugreift. Anschließend wurde eine dezentrale, mikrocontrollerbasierte Steuerung realisiert. Jeder Mikrocontroller steuert dabei je ein Logistik- oder Bearbeitungsmodul. Über ein Bussystem können alle Controller kommunizieren und koordiniert werden.

Die Mikrocontroller-Steuerung setzt eine SOA um. Dabei wird jedes Modul dezentral von einem Mikrocontroller gesteuert. Jeder Mikrocontroller bietet im Netzwerk den entsprechenden Service an. Intelligente Produkte können über das Netzwerk die Services abrufen. So wird eine sehr flexible Produktion mit einer hohen Rekonfigurierbarkeit erreicht. Abhängigkeiten zwischen Modulen werden echtzeitfähig aufgelöst.

Die intelligenten Produkte wurden über generische Zustandsautomaten auf einem Desktop-PC realisiert. Dabei repräsentiert jeder Zustand einen Bearbeitungsschritt.

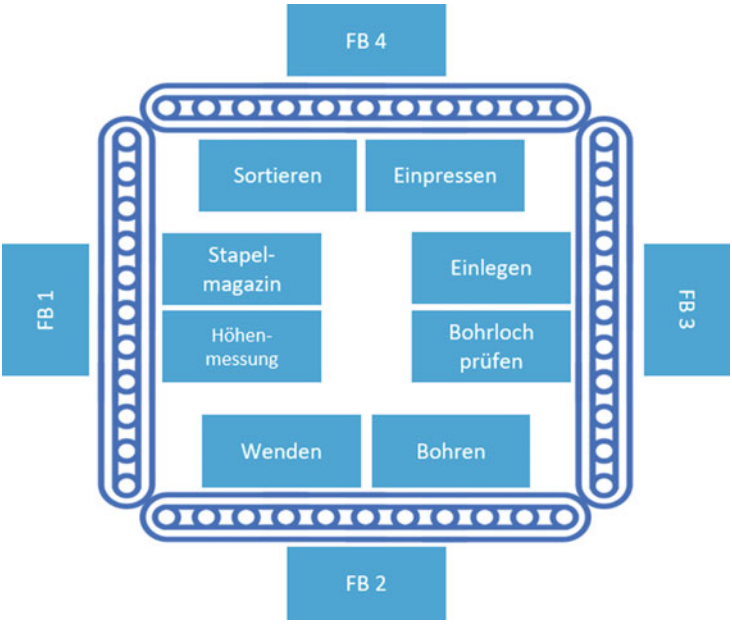


Abb. 13 Modulares Produktionssystem

Diese können über eine GUI individuell konfiguriert werden und über ein CAN-Interface mit dem Anlagenbus kommunizieren und so die Services abrufen. Durch die Trennung von Bearbeitungs- und Logistikmodulen ist es möglich Produkte ohne Kenntnis über die räumliche Verteilung der zur Herstellung notwendigen Bearbeitungsmodule zu konfigurieren.

5.2 Bewertung der service-orientierten Architektur

An der Laboranlage kann die SOA der zentralen SPS-Vairante gegenübergestellt werden. Beide Architekturen können anhand der Anforderungen aus Abschn. 2 bewertet werden. In Abschn. 2.2.5 wurde beschrieben, dass Software-Agenten für die Realisierung einer SOA genutzt werden können. Grundsätzlich kann das vorgestellte Konzept also auch mit einem Agentennetzwerk realisiert werden. Daher wird im Folgenden nur die SOA mit der konventionellen Realisierung verglichen.

Flexibilität: Durch die SOA wurde eine Individualproduktion realisiert. Abhängig vom Bauplan des individuellen Produkts werden unterschiedliche Module angesteuert und die entsprechenden Services abgerufen. Bereits bei dieser kleinen Labor-Anlage ist die Produktion von 64 Produktvarianten möglich. Die Variantenvielfalt der Produkte und damit die Produktflexibilität steigen exponentiell mit einer größeren Anzahl verfügbarer Module und Services. An der SPS wurde klassisch ein Ablauf umgesetzt. Neue Produktabläufe erfordern eine Änderung des SPS-Codes. Damit ist die SOA zunächst flexibler. Grundsätzlich ist es allerdings möglich alle Varianten im SPS-Code zu berücksichtigen. Dann wären beide Steuerungsparadigmen gleich flexibel. Die hohe Flexibilität würde sich die SPS-Variante jedoch mit enormem Entwicklungsaufwand erkaufen.

Heterogenität: Das Framework ermöglicht eine einfache Portierung auf andere Hardware oder Bus-Systeme, da hardwarespezifische Modell-Teile in die BSW ausgelagert sind. Das SOA-Protokoll wird in der Applikationsschicht implementiert und kann auf beliebige Kommunikations-Stacks aufgesetzt werden. Eine Portierung des SPS-Codes auf eine andere Hardware ist nicht sinnvoll. Im Kontext der Heterogenität stellt sich im SPS-Kontext die Frage, ob neue Sensoren, Aktoren oder Bus-Koppler mit der SPS kompatibel sind.

Rekonfigurierbarkeit: Die SOA am MPS wurde rekonfigurierbar aufgebaut. Um diese nachzuweisen wurden folgende Rekonfigurations-Szenarien durchgeführt und protokolliert:

- Neubau
- Erweiterung
- Rückbau
- Austausch von Modulen (Änderung der Position im Anlagenlayout)
- Änderung in einer Modulsteuerung

Bei der Evaluation der Rekonfigurierbarkeit haben sich folgende Vorteile der SOA gegenüber der SPS herauskristallisiert:

Die Entwicklung und Änderung von Steuerungslogik werden erleichtert, da sich die Services nur auf ein Modul beziehen und nicht der gesamte SPS-Code nachvollzogen werden muss. Außerdem wird der Entwicklungsprozess durch die modellgetriebene Entwicklung unterstützt.

Änderungen im Layout (Erweiterung, Rückbau, Austausch) werden durch die wohldefinierten Schnittstellen der Services vereinfacht. Diese ermöglichen es die Services durch Konfiguration schnell an den neuen Ort im Layout anzupassen. Änderungen am Produkt oder den anderen Modulen sind nicht notwendig. Abhängigkeiten zwischen den Modulen werden durch die SOA-Mechanismen aufgelöst. Bei der SPS müssen Abhängigkeiten manuell aufgelöst werden.

Echtzeitfähigkeit: Der CAN-Bus mit einer hohen Baud-Rate und einer geringen Bus-Auslastung gewährleistet garantierte Antwortzeiten. Bei höheren Echtzeitanforderungen ist die Portierung auf einen anderen Bus einfach realisierbar, da die Heterogenität im Konzept berücksichtigt wurde. Die SPS arbeitet zyklisch und gewährleistet so ebenfalls garantierte Antwortzeiten (Abb. 14).

Über den Trace der Kommunikation auf dem Bus lässt sich die Reaktionszeit auf eine Service-Anfrage messen. Der rote Rahmen im Trace zeigt die Abarbeitung eines Produktionsschrittes vom Service-Aufruf über den Transport bis zur Abarbeitung des Service. Über die Baudrate und die Konfiguration der Zykluszeiten, in denen die Modelle gerechnet werden, können die Reaktionszeiten beeinflusst werden.

Die Laufzeiten genügen den Echtzeitanforderungen des MPS. Theoretisch sind deutlich schnellere Reaktionszeiten realisierbar. Die Reaktion auf ein Event dauert zwei Berechnungen auf dem Controller und eine Call-Service-Message auf dem Bus. Die Zykluszeiten für die Berechnung der Modelle können in wenigen Mikrosekunden berechnet werden. Für das Versenden einer 8 Byte CAN-Message mit CAN-Frame (insgesamt 108 bit) wird bei einer Baud-Rate von 500kbaud eine Laufzeit von 216µs benötigt. Reaktionszeiten von unter einer Millisekunde sind in einer SOA also durchaus erreichbar.

5.3 Rahmenbedingungen

Bei der Realisierung und Evaluation des Konzepts wurden einige Annahmen getroffen. Diese sollen hier aufbereitet werden.

Zunächst wurden nur stationäre Logistikmodule für die Realisierung verwendet. Unter gewissen Rahmenbedingungen kann das Konzept auch für mobile Logistikmodule, wie fahrerlose Transportsysteme eingesetzt werden. Die Voraussetzung hierfür ist, dass sich Transportaufträge als Services kapseln lassen. Bei definierten Zielorten für Transportaufträge, z. B. Lagerorten oder Bearbeitungsmodulen, können relativ einfach Services im Netzwerk angeboten werden, um die Produktion mit Hilfe von mobilen Logistikmodulen zu realisieren. Bei der Realisierung einer entsprechenden Mensch-Maschine-Schnittstelle und Ausgabe von klaren Anweisungen wäre sogar ein manueller Transport durch Menschen denkbar, um einen Service abzuarbeiten und so Bedienpersonal in die Produktion einzubinden.

Receive / Transmit | Trace | PCAN-USB

Recording... | 1028,9635 s | 0,20 % | Ring Buffer | Rx: 204 | Tx: 0 | Errors: 0

Time	Type	ID	DLC	Data
1016,3093	Data	002	8	21 01 01 00 00 00 00 00
1016,3525	Data	001	8	02 01 00 00 00 00 00 00
1016,3840	Data	002	8	08 01 02 00 00 00 00 00
1016,4140	Data	003	8	07 02 00 00 00 00 00 00
1016,4329	Data	004	8	07 07 02 00 00 00 00 00
1016,4644	Data	005	8	07 01 07 02 00 00 00 00
1016,4928	Data	006	8	21 07 01 02 00 00 00 00
1019,3946	Data	003	8	08 02 00 00 00 00 00 00
1019,4128	Data	004	8	08 07 02 00 00 00 00 00
1019,4346	Data	005	8	08 01 07 02 00 00 00 00
1019,4624	Data	006	8	21 08 01 02 00 00 00 00
1019,5042	Data	003	8	09 02 00 00 00 00 00 00
1019,5224	Data	004	8	09 07 02 00 00 00 00 00
1019,5339	Data	005	8	09 01 07 02 00 00 00 00
1019,5632	Data	006	8	21 09 01 02 00 00 00 00
1019,6145	Data	002	8	21 01 02 00 00 00 00 00
1019,6629	Data	001	8	04 01 00 00 00 00 00 00
1019,6855	Data	002	8	08 01 04 00 00 00 00 00
1019,7155	Data	003	8	07 04 00 00 00 00 00 00
1019,7330	Data	004	8	07 08 04 00 00 00 00 00
1019,7556	Data	005	8	07 01 08 04 00 00 00 00
1019,7907	Data	006	8	21 07 01 04 00 00 00 00
1019,8310	Data	003	8	07 08 00 00 00 00 00 00
1019,8426	Data	004	8	07 07 08 00 00 00 00 00
1019,8704	Data	005	8	07 01 07 08 00 00 00 00
1019,8929	Data	006	8	21 07 01 08 00 00 00 00
1021,5630	Data	008	8	37 08 00 00 00 00 00 00
1023,4512	Data	008	8	42 07 00 00 00 00 00 00
1023,5129	Data	001	8	01 02 00 00 00 00 00 00
1023,9099	Data	002	8	08 02 01 00 00 00 00 00
1027,1860	Data	003	8	08 04 00 00 00 00 00 00
1027,2035	Data	004	8	08 08 04 00 00 00 00 00
1027,2158	Data	005	8	08 01 08 04 00 00 00 00
1027,2509	Data	006	8	21 08 01 04 00 00 00 00
1028,5465	Data	003	8	09 04 00 00 00 00 00 00
1028,5632	Data	004	8	09 08 04 00 00 00 00 00
1028,5762	Data	005	8	09 01 08 04 00 00 00 00
1028,6114	Data	006	8	21 09 01 04 00 00 00 00
1028,6561	Data	002	8	21 01 04 00 00 00 00 00
1028,7031	Data	001	8	02 02 00 00 00 00 00 00
1028,7249	Data	002	8	08 02 02 00 00 00 00 00
1028,7534	Data	001	8	03 02 00 00 00 00 00 00
1028,7554	Data	005	8	07 02 07 02 00 00 00 00
1028,7832	Data	002	8	08 02 03 00 00 00 00 00
1028,7844	Data	006	8	21 07 02 02 00 00 00 00
1028,8129	Data	003	8	07 03 00 00 00 00 00 00
1028,8333	Data	004	8	07 08 03 00 00 00 00 00
1028,8633	Data	005	8	07 02 08 03 00 00 00 00
1028,9013	Data	006	8	21 07 02 03 00 00 00 00
1028,9416	Data	005	8	07 02 07 08 00 00 00 00
1028,9635	Data	006	8	21 07 02 08 00 00 00 00

Connected to PCAN-USB (100 kBit/s) | Overruns: 0 | QXmtFull: 0

Abb. 14 CAN-Trace

Die zweite Rahmenbedingung ist die Verwendung von Mikrocontrollern zur Realisierung. Eine dezentrale und durch eine SOA koordinierte Steuerung kann auch mit SPSen realisiert werden. Hier muss zunächst die Open Platform Communication (OPC) Foundation mit ihrer OPC Unified Architecture (OPC-UA) genannt werden. OPC-UA als Kommunikationsstandard bietet die Möglichkeit eine SOA zu realisieren. OPC-UA bietet hierfür ein Kommunikationsprotokoll, darüber hinaus arbeiten Standardisierungsgremien an Standards für domänenspezifische Informationsmodelle, um auch die Semantik zu standardisieren. Dieser Quasi-Industriestandard wird an aktuellen SPS-Modellen als Kommunikationsprotokoll bereits angeboten. Die Spezifikation von Nachrichteninhalten, die für die Realisierung des Konzepts im Prototyp notwendig war, wird zukünftig vereinfacht oder im Idealfall entfallen, wenn die Standardisierungen entsprechend fortgeschritten sind. Die Implementierungen werden ebenfalls vereinfacht, wenn Geräte genutzt werden können, die entsprechende Protokolle unterstützen.

In der vorgestellten Realisierung sind Deadlocks aufgrund der Linienstruktur der Intralogistik nicht möglich. Daher wurden Konzepte zur Deadlockverhinderung oder -vermeidung nicht implementiert. Wenn IPs ihren Weg zur Laufzeit dynamisch umplanen, um auf Fehler in einzelnen Modulen zu reagieren oder um aufgrund neuer Modulbelegungen durch andere Produkte den Weg zu optimieren, sind theoretisch auch Lifelocks denkbar.

Für den Einsatz in einer Matrix-Struktur wären folgende Konzepte denkbar:

1. Deadlock Prevention durch die Verhinderung von Hold and Wait: Wenn ein Produkt einen Weg durch die Anlage benötigt, werden alle dafür notwendigen Services gleichzeitig zugeteilt. So kann eine Verklemmung aufgrund mehrerer Produkte, die den gleichen Service benötigen nicht auftreten.
2. Ein Life- oder Deadlock kann aufgelöst werden, indem eine intelligente Überwachung der Prozesse (Transportaufträge durch IPs) implementiert wird und den Produkten zugewiesene Services entzogen werden (Preemtion). Eine solche Überwachung könnte zentral im Discovery-Server implementiert werden.

6 Zusammenfassung

Um den Anforderungen der zukünftigen Produktion nach erhöhter Flexibilität und Rekonfigurierbarkeit gerecht zu werden, werden Lösungen für die Intralogistik von Produktionssystemen benötigt, welche die Flexibilität des Produktionssystems unterstützen und auch nach Rekonfiguration des Produktionssystems noch greifen.

- Das Paradigma der SOA wurde aufgrund der Erfüllung der angeführten Anforderungen an die Koordination der Intralogistik von Produktionssystemen ausgewählt.
- Die vorgestellte Service-Hierarchie der Organisationsstruktur für die SOA deckt sowohl die Bearbeitungsprozesse als auch die Intralogistikprozesse von Produktionssystemen ab und basiert auf einer Entkopplung von Produkt, Prozess und

Intralogistik durch die Einführung entsprechender Layer. Dies ermöglicht die Organisation der Intralogistik innerhalb des Logistik Layers in Form eines hochautomatisierten Intralogistiksystems.
- Zur Realisierung einer echtzeitfähigen Kommunikation wurde ein leichtgewichtiges, busunabhängiges, SOA-Protokoll entwickelt, welches zur Umsetzung des Koordinationskonzepts eingesetzt wird.
- Darüber hinaus wurde ein Schalenmodell präsentiert, das als Modellierungsframework eine modellgetriebene Entwicklung ermöglicht und so einen optimierten Entwicklungsprozess für die Anlagen- und Intralogistiksteuerung bereitstellt.

Das vorgestellte Protokoll wird nicht allen Anforderungen der Automatisierungstechnik genügen. Als SOA-Protokoll wird OPC-UA als Quasi-Industriestandard eine große Rolle spielen. Gegenwärtig wird von den Organisationen IEC SC65C/MT9 und IEEE 802 versucht die Kommunikation im Feld als Profil für OPC-UA zu standardisieren. Ein solcher Standard für die Feldebene könnte dann genutzt werden, um die beschriebene Service-Hierarchie und die Aufruf-Mechanismen tatsächlich herstellerunabhängig umzusetzen.

Für einen industriellen Einsatz sind weitere Studien bezüglich der Skalierbarkeit in Kombination mit Echtzeitfähigkeit notwendig. Insbesondere der Kommunikationsaufwand wird mit dem hochskalieren der Anlage deutlich steigen, sodass möglicherweise eine Partitionierung des SOA-Netzwerks oder der Einsatz von Feldbussen mit höherer Bandbreite notwendig werden können. Beide Ansätze können simulationsbasiert analysiert werden.

Literatur

Abel H-B, Blume H-J, Skabrond K, Beikirch H, Boller S, Frey G, Kraft D, Löhr W, Meyer H, Predelli O et al (2006) Handbuch der Mess-und Automatisierungstechnik im Automobil: Fahrzeugelektronik, Fahrzeugmechatronik. Springer-Verlag Berlin Heidelberg

Bloch H, Fay A, Knohl T, Hoernicke M, Bernshausen J, Hensel S, Hahn A, Urbas L (2017) A microservice-based architecture approach for the automation of modular process plants 2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA), S 1–8

Caridi M, Cavalieri S (2004) Multi-agent systems in production planning and control: an overview. Prod Plan Control 15:106–118

Cucinotta T, Mancina A, Anastasi GF, Lipari G, Mangeruca L, Checcozzo R, Rusinà F (2009) A real-time service-oriented architecture for industrial automation. IEEE Trans Ind Inf 5:267–277

Faul A, Beyer T, Klein M, Vögeli D, Körner R, Weyrich M, Vogel-Heuser B (2018) Eine agentenbasierte Produktionsanlage am Beispiel eines Montageprozesses. In: Software-Agenten in der Industrie 4.0. S 89–108. https://doi.org/10.1515/9783110527056-005

Fay A, Wassermann E (2018) Sicherstellung von Interoperabilität. atp magazin 60:34–45

Foundation for Intelligent Physical Agents (2004) FIPA Agent Management Specification, Document Number SC00023K. http://www.fipa.org, Genf, Schweiz

Geisberger E, Broy M (2012) CPS-Themenfelder. Springer, Berlin/Heidelberg

Hansson MN, Järvenpää E, Siltala N, Madsen O (2017) Modelling capabilities for functional configuration of part feeding equipment. Procedia Manuf 11:2051–2060

Hees AF (2017) System zur Produktionsplanung für rekonfigurierbare Produktionssysteme. Herbert Utz Verlag GmbH, München

Hennecke A, Ruskowski M (2018) Design of a flexible robot cell demonstrator based on CPPS concepts and technologies 2018 IEEE Industrial Cyber-Physical Systems (ICPS). Saint-Petersburg, Russia, S 534–539

Hompel Mten (2006) Zellulare Fördertechnik. Logistics Journal: nicht-referierte Veröffentlichungen. https://www.logistics-journal.de/not-reviewed/2006/8/599. zugegriffen im August 2006

Järvenpää E, Siltala N, Lanz M (2016) Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems 2016 IEEE international symposium on assembly and manufacturing (ISAM). Fort Worth, TX, USA, S 120–125

Kagermann H, Anderl R, Gausemeier J, Schuh G, Wahlster W (2016) Industrie 4.0 im globalen Kontext: Strategien der Zusammenarbeit mit internationalen Partnern. Herbert Utz Verlag GmbH, München

MacKenzie CM, Laskey K, McCabe F, Brown PF, Metz R, Hamilton BA (2006) Reference model for service oriented architecture 1.0. OASIS standard 12

McFarlane D, Sarma S, Chirn JL, Wong CY, Ashton K (2002) The intelligent product in manufacturing control and management. IFAC Proc Vol 35:49–54

Msadek N, Kiefhaber R, Ungerer T (2015) A trustworthy, fault-tolerant and scalable self-configuration algorithm for organic computing systems. J Syst Archit 61:511–519

Müller-Schloer C, Schmeck H, Ungerer T (2012) Organic Computing. Informatik Spektrum 35:71–73. https://doi.org/10.1007/s00287-012-0599-2

Pantförder D, Mayer F, Diedrich C, Göhner P, Weyrich M, Vogel-Heuser B (2017) Agentenbasierte dynamische Rekonfiguration von vernetzten intelligenten Produktionsanlagen Handbuch Industrie 4.0 Bd. 2. Springer Vieweg, Berlin, Heidelberg, S 31–44

Parasuraman R, Sheridan TB, Wickens CD (2000) A model for types and levels of human interaction with automation. IEEE Trans Syst Man Cybern-Part A: Syst Humans 30:286–297

Regulin D, Vogel-Heuser B (2017) Agentenorientierte Verknüpfung existierender heterogener automatisierter Produktionsanlagen durch mobile Roboter zu einem Industrie-4.0-System. In Handbuch Industrie 4.0 Bd. 2, S. 93–118. Springer Vieweg, Berlin, Heidelberg

Ribeiro L, Barata J, Mendes P (2008) MAS and SOA: complementary automation paradigms international conference on information technology for balanced automation systems. Springer, Boston, MA, S 259–268

Schiekofer R, Scholz A, Weyrich M (2018) REST based OPC UA for the IIoT 2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA). Turin, Italy, S 274–281

Schmidt J-P, Müller T, Weyrich M (2018) Methodology for the model driven development of service oriented plant controls. Procedia CIRP 67:173–178

Smriti C (2014) Four main types of plant layout. http://www.yourarticlelibrary.com/industries/plant-layout/four-main-typesof-plant-layout/34604/. Zugegriffen am 10.12.2019

Stehle T, Heisel U (2017) Konfiguration und Rekonfiguration von Produktionssystemen. In Neue Entwicklungen in der Unternehmensorganisation, S. 333–367 Springer Vieweg, Berlin, Heidelberg

Vogel-Heuser B, Fay A, Seitz M, Gehlhoff F (2019) Agenten zur Realisierung von Industrie 4.0. VDI/VDE-Gesellschaft Mess-und Automatisierungstechnik

Wolff E (2016) Microservices; Grundlagen flexibler Softwarearchitekturen. dpunkt, Heidelberg

Zaeh MF, Moeller N, Muessig B, Rimpau C (2006) Life cycle oriented valuation of manufacturing flexibility proceedings of the 13th CIRP international conference on life cycle engineering, LICE. Leuven, Belgien, S 699–704