http://dx.doi.org/10.25673/116045

„Ulrich Jumar, Christian Diedrich (Hrsg.):
EKA 2024 - Entwurf komplexer Automatisierungssysteme, 18. Fachtagung“

Konzepte zur Realisierung proaktiver Verwaltungsschalen und deren Kommunikation

Untersuchung der Einsatzmöglichkeiten von Events und Operationen für den Nachrichtenaustausch

Melanie Stolze[1], Gustavo Cainelli[2], Christian Kosel[3], Alexander Belyaev[4], Christian Diedrich[5]

Abstract: Sowohl die passive als auch die reaktive Verwaltungsschale (VWS) halten immer mehr Einzug in die Praxis. Ebenso wird der Einsatz und die Entwicklung proaktiver VWS, deren Umsetzung zum aktuellen Zeitpunkt noch nicht spezifiziert ist, in aktuellen Forschungsprojekten evaluiert. Das Ziel ist, einen Überblick über die derzeit bestehenden Möglichkeiten für die Implementierung einer proaktiven VWS zu geben und somit einen Teil zu deren Spezifizierung beizutragen. Die drei Schwerpunkte liegen dabei auf der Realisierung der Architektur und Schnittstelle, sowie auf der flexiblen Integration verschiedener Interaktionsmuster für proaktive VWS.

Keywords: Verwaltungsschale, Interaktion, I4.0-Sprache, Digitaler Zwilling

1 Motivation und Problemstellung

Mittlerweile hält die Verwaltungsschale (VWS) immer mehr Einzug in die Praxis. Die VWS kann dabei in 3 Arten (passiv, reaktiv und proaktiv) realisiert werden, wie in Abbildung 1 dargestellt ist. Komponentenhersteller bieten ihren Kunden bereits sowohl passive als auch reaktive VWS über ihre Weboberflächen an [WI24], [WA24]. Ebenso wird der Einsatz und die Weiterentwicklung der VWS in Forschungsprojekten der Konjunktur- und Zukunftspakete KoPa35c [BA24], Gaia-X [BM24] und Manufacturing-X [PI24] des Bundesministeriums für Wirtschaft und Klimaschutz vorangetrieben. Forschungsprojekte aus KoPa35c, wie „Verwaltungsschale für den Leitungssatz“ VWS4LS [AR24], fokussieren die Nutzung passiver, reaktiver als auch proaktiver VWS, während sich die anderen genannten Zukunftspakete mit Datenökosystemen beschäftigen. Arbeiten wie [Ju23] zeigen bereits sehr gut, wie eine Brücke zwischen den in den geförderten Zukunftspaketen entwickelten Ergebnissen geschlagen werden kann. Dafür

[1] Institut für Automation und Kommunikation e.V., IKT & Automation, Werner-Heisenberg-Str. 1, 39106 Magdeburg, melanie.stolze@ifak.eu
[2] Institut für Automation und Kommunikation e.V., IKT & Automation, Werner-Heisenberg-Str. 1, 39106 Magdeburg, gustavo.cainelli@ifak.eu
[3] Arena 2036 e.V., Pfaffenwaldring 19, 70569 Stuttgart, christian.kosel@arena2036.de
[4] Otto von Guericke Universität Magdeburg, 39106 Magdeburg, alexander.belyaev@ovgu.de
[5] Otto von Guericke Universität Magdeburg, 39106 Magdeburg, christian.diedrich@ovgu.de

verbindet [Ju23] die Eigenschaften proaktiver VWS und ihrer I4.0-Sprache mit denen in einem Datenökosystem vorhandenen Konnektoren zum unternehmensübergreifenden Datenaustausch.

Diese Arbeit legt den Fokus jedoch auf die Implementierungsmöglichkeiten der proaktiven VWS und der I4.0-Sprache und präsentiert unter anderem Forschungsergebnisse aus dem Projekt VWS4LS [AR24], in dem die Leitungssatzentwicklung und -fertigung mit Hilfe von VWS effektiver gestaltet werden soll. In der Leitungssatzfertigung steigt durch heterogene Maschinenparks und Produkte der Losgröße 1 der Zeitaufwand für die Aufgaben in der Fertigungsplanung wesentlich an. Unregelmäßigkeiten im Produktionsablauf, wie z. B. ungeplante Instandsetzungen, Materialflussschwierigkeiten oder kurzfristige Bedarfsschwankungen führen zu neuen Ablaufplänen und erschweren die Planung. Zur Lösung des Problems wird in [AR24] der Einsatz proaktiver VWS evaluiert, die die Fertigungsplaner in ihren Tätigkeiten unterstützen sollen, teilautomatisiert geeignete Ressourcen für ein zu fertigendes Produkt zu finden. Bei dieser Art von VWS erfolgt eine eigenständige Kommunikation zwischen VWS mittels der in der VDI/VDE 2193-1 spezifizierten I4.0-Sprache [VD20a]. Allein mit [VD20a] ist die proaktive VWS jedoch noch nicht spezifiziert und somit existieren verschiedene Lösungen für deren reale Umsetzung, die in diesem Bericht zusammengefasst werden. Beginnend führt Kapitel 0 in Architekturvarianten ein, die sowohl aus der Literatur sowie eigenen im Projekt VWS4LS entwickelten Konzepten bekannt sind. Kapitel 3 beleuchtet die Einsatzmöglichkeiten der im VWS-Metamodell [ID23a] spezifizierten Elemente „Operation“ und „Event“ und deren Vor- und Nachteile, um die proaktive VWS mit Kommunikationsfähigkeiten auszustatten. Dabei wird auch bezüglich der spezifizierten Schnittstelle der VWS [ID23b] untersucht, inwieweit sie die Übergabe von I4.0-Nachrichten gemäß der VDI/VDE 2193-1 ermöglicht. Zum Ende wird in Kapitel 0 die Integration verschiedener Interaktionsmuster in den vorgeschlagenen Architekturen diskutiert.

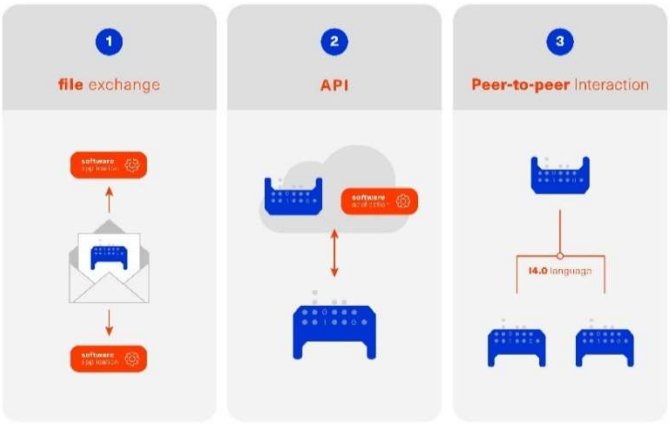

Abbildung 1: Die drei Arten von Verwaltungsschalen: passiv (1), reaktiv (2) und proaktiv (3) [ID23a]

2 Architekturen für die proaktive Verwaltungsschale

In [BD19] wird ein abstraktes Konzept für die Architektur einer proaktiven VWS vorgestellt. Dort setzt sich die proaktive VWS aus einem aktiven und passiven Part zusammen. Während der passive Part die Daten des Assets über die VWS-Teilmodelle in Form eines Datenmodells bereitstellt, übernimmt der aktive Part die Ausführung von Algorithmen, deren Orchestrierung, sowie die Aussendung und das Empfangen von I4.0-Nachrichten. [BD19]

Zu sehen ist das abstrakte Konzept in Abbildung 2. Die Komponenten des aktiven Parts werden in [BD19] wie folgt beschrieben werden:

- Ein Interaktionsmanager, der die Zustandsmaschinen für ausgewählte Interaktionsmuster implementiert,
- mehrere Algorithmen, die Berechnungen in einzelnen Zuständen der Zustandsmaschine mit Hilfe der Daten aus dem passiven Part übernehmen,
- ein Komponentenmanager, der die Kommunikation zwischen dem Interaktionsmanager und den Algorithmen orchestriert, sowie
- ein Messenger als Kommunikationsschnittstelle, der zum einen die Nachrichten von dem Interaktionsmanager entgegennimmt, diese gemäß der I4.0-Sprache aufbereitet und nach außen versendet und ebenso I4.0-Nachrichten von außen entgegennimmt und wiederum an den Interaktionsmanager weitergibt.

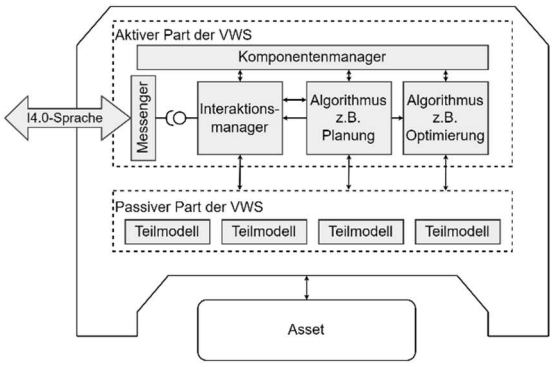

Abbildung 2: Abstrakte Architektur für eine Typ 3 VWS (in Anlehnung an [BD19])

Für die folgenden Abschnitte wird folgender Grundsatz für die Implementierung einer proaktiven VWS angenommen. Der passive Part wird durch einen VWS-Server realisiert, der zum einen eine Schnittstelle für die Assetanbindung und zum anderen eine Schnittstelle zu anderen Applikationen bereitstellt, der VWS-API (Application Programming Interface) [ID23b]. Die Realisierung des Servers kann mit Hilfe des Eclipse BaSyx™ Frameworks [EB24] geschehen. Die bis zu diesem Punkt implementierten Funktionalitäten repräsentieren eine reaktive VWS. Dabei kann ein Server nur eine

(genannt VWS-Server) oder aber mehrere VWS (genannt VWS-Repository) verwalten. Damit aus einer reaktiven VWS eine proaktive VWS wird, muss diese in die Lage versetzt werden, I4.0-Nachrichten zu verschicken, wie es in Abbildung 1 unter Nummer 3 und in [PI20] definiert ist. Dazu werden VWS-Elemente in der proaktiven VWS modelliert, die den späteren Austausch von I4.0-Nachrichten ermöglichen. Diese Elemente werden für die folgenden Abschnitte in jeder proaktiven VWS als gegeben betrachtet und in Kapitel 3 näher erläutert. Auf dem beschriebenen Grundsatz basierend werden in den nächsten Abschnitten drei Arten von konzeptionellen Architekturen gezeigt, mit denen eine proaktive VWS implementiert werden kann.

2.1 Monolithischer Ansatz

Bei dem monolithischen Ansatz ist laut [Ge15] im Allgemeinen eine einzige zentrale Applikation zur Koordinierung und Steuerung aller Funktionen und Prozesse zuständig, wobei die Applikation konzeptionell als eine zusammenhängende und unteilbare Einheit behandelt wird. Implementierungsmäßig können die zentral gesteuerten Funktionen jedoch auf mehrere Prozesse und Hardwarekomponenten verteilt sein. Für die Implementierung einer proaktiven VWS bedeutet dies, dass der passive Part mit Hilfe des VWS-Servers/-Repositories bereitgestellt wird. Zudem übernimmt in diesem Fall der VWS-Server/-Repository auch die Realisierung der Algorithmen. Die Algorithmen werden in Form von VWS-Services oder Teilmodell-Services programmiert, die beim Start des Servers eingelesen und mit einem eingehenden Trigger durch den Server ausgeführt [Gr22] werden. Aufgrund der laufenden Entwicklungsarbeiten am Eclipse BaSyx™ Framework [EB24] wurden in [Gr22] die Services noch als „Lambda Funktionen“ bezeichnet. Abbildung 3 zeigt anhand eines VWS-Servers/-Repositories die beiden Möglichkeiten zur Umsetzung einer proaktiven VWS. Die Anteile, aus denen sich eine proaktive VWS zusammensetzt sind in allen folgenden Architekturabbildungen rot markiert. Während der VWS-Server als eine Einheit eine proaktive VWS bildet, setzt sich diese in einem VWS-Repository aus Einem von mehreren aktiven und passiven Parts zusammen sowie dem aktiven Part mit Algorithmen, der auf alle VWS angewendet werden kann. Es ist ersichtlich, dass der Programmcode bei Nutzung mehrerer VWS-Server dupliziert wird. Dies erschwert mit zunehmender Komplexität den Wartungsaufwand der Applikationen. Um dies zu umgehen, können mit Hilfe von VWS-Repositories und durch die spezifizierten VWS-Teilmodelle, Algorithmen generisch programmiert und zur Laufzeit für jede VWS individuell ausgeführt werden. Jedoch duplizieren sich auch hier die Algorithmen, wenn man mehr als ein VWS-Repository instanziiert. In [Gr22] wurde beschrieben, dass ein solcher Ansatz bei kleinen Anwendungen noch einfach zu handhaben ist. Für komplexere proaktive VWS, deren Funktionalitäten sich über die Lebensdauer der VWS erweitern, ist dieser Ansatz laut [Gr22] jedoch nicht zu empfehlen. Ein Grund dafür kann sein, dass neu entwickelte Algorithmen unter der Beachtung von Abhängigkeiten in bereits bestehende Funktionen

und Komponenten integriert werden müssen. Mit einem wachsenden Gesamtsystem, steigt auch die Anzahl der Abhängigkeiten.

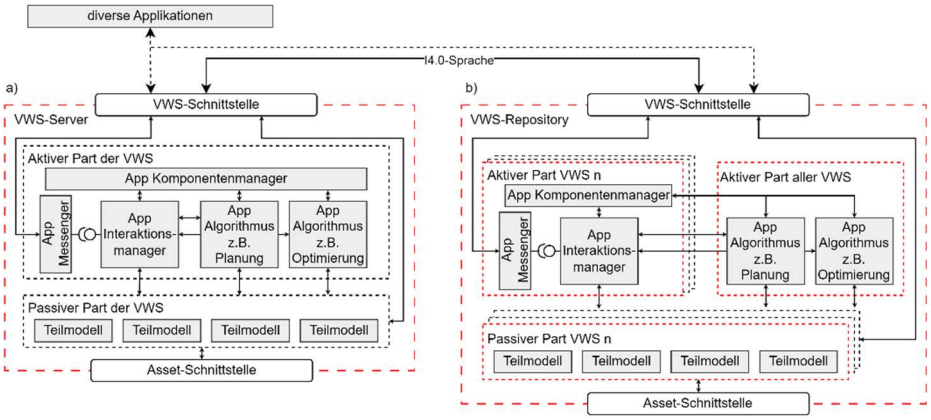

Abbildung 3: Proaktive VWS, realisiert mit dem monolithischen Ansatz. a) Verwendung von VWS-Server, b) Verwendung von VWS-Repository

Die in Abbildung 3 dargestellte VWS-Schnittstelle des VWS-Servers/-Repositories, realisiert in dieser Arbeit mehrere Aufgaben. Zum einen stellt sie die von der IDTA spezifizierte VWS-API [ID23b] bereit, um Daten aus der VWS über http/REST (Hypertext Transfer Protocol / Representational State Transfer) zu lesen und zu schreiben. Diese Kommunikation ist durch gestrichelte Pfeile gekennzeichnet. Zum anderen ermöglicht die Schnittstelle mit Hilfe des App Messengers das aktive Versenden und Empfangen von I4.0-Nachrichten (durchgezogene Pfeile).

2.2 Agentenbasierter Ansatz

Der agentenbasierte Ansatz ist durch die Präsenz von autonom agierenden Agenten charakterisiert. Jeder Agent beinhaltet die Fähigkeiten, Schlussfolgerungen zu ziehen, Entscheidungen zu treffen und Informationen auszusenden, um z. B. in Interaktion mit anderen Agenten gemeinsame Ziele zu erreichen. Entgegen dem monolithischen Ansatz bietet dieser mehr Flexibilität, Dezentralisierung und Ausfallsicherheit für ein System. Jedoch ist das Management der miteinander interagierenden Agenten komplex, vor allem in verteilten Systemen. Zudem steigen mit erhöhtem Kommunikationsbedarf auch die Latenzzeiten im System an. [SLP22], [CF21] Angewandt auf die proaktive VWS kann eine agentenbasierte Architektur wie in Abbildung 4 realisiert werden. Dabei sind de Komponenten, die von Agenten repräsentiert werden durch ein entsprechendes Agentensymbol markiert. Zudem gibt es keinen Komponentenmanager mehr, der die Agenten orchestrieren würde.

In der Literatur gibt es durchgehend agentenbasierte Ansätze zur Realisierung einer proaktiven VWS. Dabei gibt es Arbeiten wie [Is23], [SLP22], [SPL23], [Ca23] und [Gr22], die deutlich von der Implementierung einer proaktiven VWS sprechen. Andere Arbeiten wie [YH20] und [Ju23] sprechen von reaktiven VWS. Die zuletzt genannten Arbeiten realisieren jedoch durch die Zusammenschaltung der reaktiven VWS mit anderen Applikationen indirekt Teile einer proaktiven VWS. Nur wenige der genannten Arbeiten beziehen jedoch die I4.0-Sprache in ihr Konzept mit ein, die als ein Hauptmerkmal der proaktiven VWS in [PI20] definiert ist. Durch die Verlagerung des Interaktionsmanagers in einen Agenten, können VWS flexibel an Interaktionsmuster angebunden bzw. wieder entfernt werden, wenn sie die für die Interaktion notwendigen Daten in den VWS-Teilmodellen abbilden.

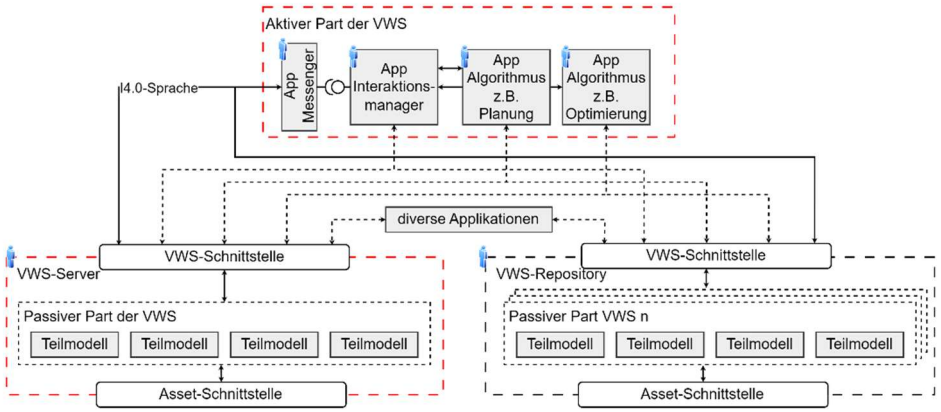

Abbildung 4: Proaktive VWS, realisiert mit dem agentenbasierten Ansatz

In [SLP22] und [SPL23] wird eine Methodologie vorgeschlagen, VWS mit Mutli-Agenten-System (MAS) zu verbinden. Darin wird eine Richtlinie beschrieben, die die Entwicklung von proaktiven VWS mit Hilfe der Vorteile von MAS forciert. In [Gr22] ist der aktive Part mittels externer Applikationen realisiert, die die für eine Interaktion notwendigen Informationen aus dem VWS-Server/-Repository weiterverarbeiten. Wiederum eine andere Variante wird in [Is23] beschrieben, wo ganze VWS oder einzelne Teilmodelle mit Agenten ausgestattet werden. Zusätzlich wird in [Ca23] und [YH20] der VWS-Server nicht mit Hilfe des Eclipse BaSyx™ Frameworks [EB24] implementiert, sondern über einen OPC UA Server, der das VWS Modell verwaltet. Der aktive Part wird in diesem Fall durch einen OPC UA Client übernommen, der die Businesslogik implementiert.

Der Vorteil der agentenbasierten Architektur ist, dass einzelne Agenten für mehrere VWS verwendet werden können und somit nicht mehrfach entwickelt werden müssen. Jedoch entsteht durch die Anwendung eines Agenten auf viele VWS ein zentralistischer Ansatz, der bei Ausfall eines Agenten eine Beeinträchtigung vieler VWS nach sich zieht. Zudem benötigen die separierten Agenten ein integriertes Management.

2.3 Orchestrierender Ansatz

In einem orchestrierenden Ansatz wird eine zentrale Entität verwendet, auch Orchestrator genannt, der die Kombination von Komponenten in einem Interaktionsmuster übernimmt. Da bisher noch keine Publikationen bzgl. eines solchen Ansatzes zur Implementierung einer proaktiven VWS existieren, hat das Projekt VWS4LS [AR24] ein Konzept für diesen Ansatz entwickelt. Dabei wird der Orchestrator durch ein Workflow Management System (WMS) repräsentiert, das sowohl den Messenger als auch den Komponenten- und Interaktionsmanager in einem abbildet. Das WMS implementiert hierbei die Interaktionsmuster und ermöglicht wie der agentenbasierte Ansatz die flexible Einbindung

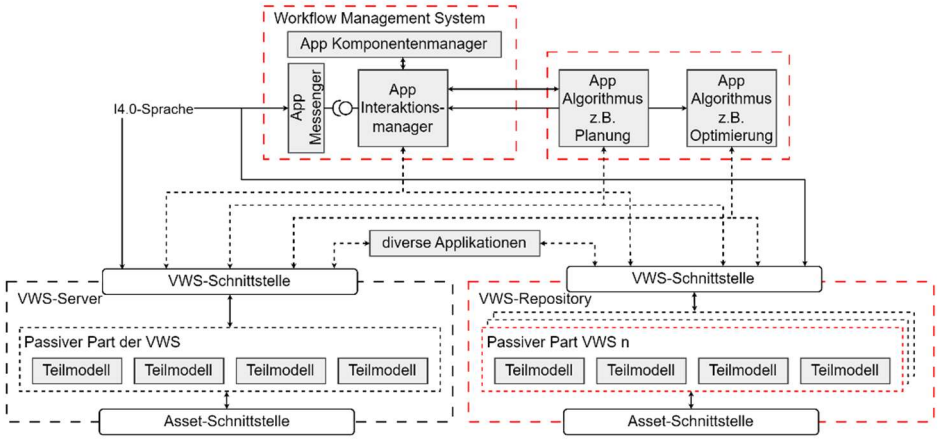

Abbildung 5: Proaktive VWS, realisiert mit dem orchestrierenden Ansatz

von neuen VWS in die Interaktionsmuster. Die Algorithmen werden auf andere Applikationen verteilt, wie z. B. einem Produktionsleitsystem. Der passive Part ist wie bei allen anderen Ansätzen durch einen VWS-Server/-Repository implementiert. Dieser Ansatz liefert einige Vorteile, z. B. die vom WMS unterstützte horizontale Skalierbarkeit, die es ermöglicht, VWS nach Bedarf hinzuzufügen oder zu entfernen. Zudem gibt es etablierte Softwaretools für die Erstellung von Workflows, die hier zum Einsatz kommen können. Unter anderem implementieren bereits erste Softwareunternehmen die VWS-API in ihre WMS-Lösungen. Andererseits besteht hier ähnlich dem agentenbasierten Ansatz die Gefahr, dass bei Ausfall des WMS alle an das WMS gekoppelten VWS ihre Funktion als proaktive VWS verlieren.

2.4 Fazit

Jede der vorgestellten Architekturen besitzt ihre Vor- und Nachteile und es kann nicht eine Architektur als die beste Lösung forciert werden. Vielmehr muss abgestimmt auf den Anwendungsfall die bestmöglichste Architektur ausgewählt werden. Beginnend mit dem

monolithischen Ansatz, bietet er für leichtgewichtige Funktionen einen guten Lösungsansatz. Durch die Verwendung eines VWS-Repositories ist es möglich, mehrfach anwendbare Algorithmen nur einmal zu integrieren. Totzdem können bei der Verwendung mehrerer VWS-Repositories, in denen die gleichen Algorithmen benötigt werden, eine Redundanz von Programmcode entstehen, die einen erhöhten Wartungsaufwand fordert. Bei Nutzung mehrerer dezentral gehaltener VWS-Repositories behalten hingegen bei Ausfall eines VWS-Repositories die proaktiven VWS in allen anderen VWS-Repositories ihre Funktionalitäten. Dieser Vorteil ist bei dem agentenbasierten und orchestrierenden Ansatz nicht per se gegeben. Wird für einen Algorithmus bzw. eine Funktionalität, z. B. nur ein Agent integriert, verlieren alle proaktiven VWS diese Funktionalität, sobald der Agent ausfällt. Dies ähnelt eher einem zentralisierten Ansatz. Vielmehr müsste die Integration eines Agenten-Typen forciert werden, von denen Agenten-Instanzen abgeleitet werden, die jeweils nur für eine Gruppe oder eine einzelne proaktive VWS verantwortlich sind. Der Vorteil wiederum ist, dass der Programmcode für die Agenten nur einmalig entwickelt werden muss und somit der Wartungsaufwand geringer als bei dem monolithischen Ansatz ist. Jedoch steigt mit mehreren Agenten auch die Kommunikationslast in einem Netz.

Das Management aller Komponenten wird bei dem monolithischen Ansatz von einer zentralen Applikation übernommen. Das hat zum Nachteil, dass bei Ausfall dieser Applikation alle Prozesse zum Erliegen kommen. Der agentenbasierte Ansatz ist demgegenüber vorteilhafter, da die Agenten sich jeweils selbst managen. Jedoch ist die übergeordnete Organisation aller Agenten nicht leicht. Die Vor- und Nachteile beider Ansätze werden durch den orchestrierenden Ansatz verbunden. Zum einen gibt es das WMS als zentrale Instanz, dass die verteilten Komponenten organisiert. Fällt das WMS jedoch aus, verlieren die proaktiven VWS ihre Funktionalität. Um diesen Nachteil zu mindern, müssten mehrere verteilte und voneinander unabhängige WMS genutzt werden.

3 Schnittstellen zum Austausch von I4.0-Nachrichten

Im vorhergehenden Kapitel wurden Architekturen zur Implementierung proaktiver VWS beschrieben. Nun folgt ein Einblick in die Implementierungsdetails des aktiven und passiven Parts, um die autonome Interaktion zwischen proaktiven VWS mit Hilfe der I4.0-Sprache mit den zurzeit spezifizierten Mitteln zu ermöglichen.

3.1 Wahl des Kommunikationsprotokolls

Für die Realisierung des Nachrichtenaustauschs werden die Kommunikationsprotokolle MQTT (Message Queuing Telemetry Transport) und http/REST betrachtet.

MQTT ermöglicht den Austausch von Topics mit flexiblem Datenrumpf. Für die Übertragung der I4.0-Nachricht sind in [BD19] bereits Beispiele für die Implementierung

des Nachrichtenrumpfs in JSON-Formaten beschrieben, die in [Gr22] und [Is23] wiederverwendet wurden. Problematisch ist bisher jedoch, dass es keine VWS-API Spezifikation für MQTT gibt. Somit besteht die Gefahr, proprietäre Schnittstellen zu entwickeln, die dann nicht miteinander kompatibel sind. Die REST-Schnittstelle der VWS hingegen ist in [ID23b] spezifiziert und die Struktur der Datenrümpfe für die Schnittstellenaufrufe sind fest vorgeschrieben. Jedoch stellt die REST-API keine Operationen bereit, die speziell für die Übergabe von I4.0-Nachrichten gedacht sind. Deshalb werden im Folgenden andere Lösungen betrachtet, die unter anderem die Interoperabilität proaktiver VWS gewährleisten sollen.

3.2 Nutzung von Events und Operationen für die Übergabe der I4.0-Nachricht

In einer proaktiven VWS muss es Funktionen geben, die die Übertragung von I4.0-Nachrichten in einer Interaktion zwischen mehr als zwei proaktiven VWS ermöglichen. Diese Funktionen werden von dem Messenger als aktive Komponente übernommen. Zur Realisierung der Funktionen gibt es die VWS-Elemente „Event“ und „Operation“. Das Event wird in der Spezifikation des VWS-Metamodells unter dem Reifegrad „experimentelle Phase“ [ID23a] geführt. Zudem ist weder in [ID23a] ein Eventtyp beschrieben, mit dem die Übertragung einer I4.0-Nachricht möglich wäre, noch eine API-Operation zum Aufruf von Events. Das führt dazu, dass der Nachrichtenaustausch über Events für die Praxis bisher noch ungeeignet ist. Die Operation hingegen besitzt eine eigene API-Operation, über die die Operation aufgerufen werden kann. Damit die Operation möglichst interoperabel für den Nachrichtenaustausch genutzt werden kann wird die Einführung eines in dem Projekt VWS4LS entwickelten Teilmodells namens „Message Participant“ vorgeschlagen (siehe Abbildung 6). Dieses Teilmodell bildet die Beschreibung für den Messenger aus dem Konzept von [BD19] und implementiert zusätzlich Operationen für das Anstoßen zusätzlicher, in der Interaktion benötigter, Algorithmen.

Unter „userOperation“ werden benutzerdefinierte Operationen modelliert, die die für ein Interaktionsmuster zusätzlich benötigten Algorithmen anstoßen können. Diese Operationen können mit den implementierten Interaktionsmustern variieren. Die Operation „newMessage“ ist für die Übertragung der I4.0-Nachricht zuständig und bleibt in ihrer Struktur konstant. Die einzigen variablen Größen bezüglich des übermittelten Inhalts bilden die „interactionElements“. Zusätzlich erhält das Teilmodell eine Auflistung der Protokolle und Rollen, an denen die proaktive VWS teilnehmen kann.

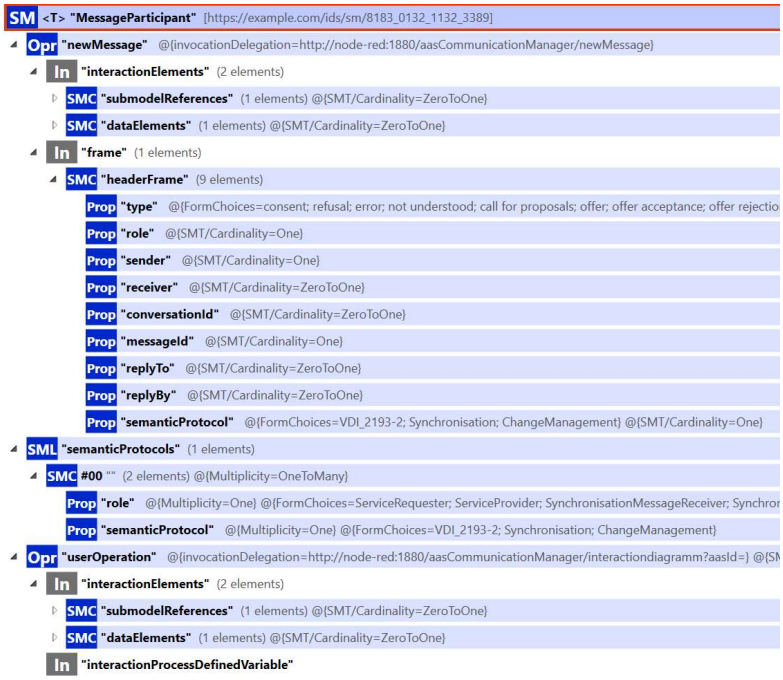

Abbildung 6: Teilmodell "Message Participant" für die Umsetzung proaktiver VWS

3.3 Fazit

Für die Interaktion mit VWS und ihren Inhalten ist bisher nur die VWS-API [ID23b] mit dem http/REST-Protokoll spezifiziert. Die Nutzung anderer Kommunikationsprotokolle ist somit vorerst proprietär. Betrachtet man die VWS-Elemente, zur Ausführung von Algorithmen, steht derzeit nur die „Operation“ zur Verfügung, da sich das „Event“ noch in der experimentellen Phase befindet. Damit jedoch I4.0-Nachrichten über die bestehenden VWS-Technologien gesendet und proprietäre Formate umgangen werden können, wurde das im Projekt VWS4LS entwickelte Teilmodel „Message Participant“ als erster Vorschlag für die interoperable Umsetzung proaktiver VWS vorgestellt. Mit Hilfe

dieses Teilmodells sind sowohl die Protokolle, mit den von der VWS eingenommenen Rollen, als auch die Struktur zur Übergabe von I4.0-Nachrichten spezifiziert.

4 Interoperabilität trotz vielfältiger Interaktionsmuster ermöglichen

Bisher wurde in der VDI/VDE 2193-2 [VD20b] das Ausschreibungsverfahren zwischen einem Auftraggeber und einem Auftragnehmer als Interaktionsmuster spezifiziert. Diese Interaktion ermöglicht sowohl die unternehmensinterne als auch unternehmensübergreifende Verhandlung für eine automatisierte Auftragsvergabe zwischen einem zu fertigenden Produkt (mit geforderten Fähigkeiten) und der Ressource (mit angebotenen Fähigkeiten). Neben dem genannten Interaktionsmuster gibt es weitere Interaktionsmuster, die aus anderen Anwendungsfällen für proaktive VWS entstehen. Darunter fallen z. B. die automatisierte Anmeldung von Wartungs- und Instandhaltungsmaßnahmen [Is23] oder das automatisierte Änderungsmanagement in der Entwicklung [AR24]. Es kann erwartet werden, dass die Anzahl der Interaktionsmuster in der Zukunft stetig steigen wird. Aufgrund dessen müssen die in den vorherigen Abschnitten vorgestellten Konzepte die Integration der individuellen Interaktionsmuster über den Interaktionsmanager als aktive Komponente ermöglichen. Ein leichtgewichtiger Lösungsansatz hierfür ist die Nutzung von Workflow Management Systemen, wie es in dem orchestrierenden Ansatz bereits erwähnt wurde. Dabei werden zunächst die zu einem Interaktionsmuster zugehörigen Zustandsmaschinen der einzelnen Interaktionsteilnehmer abgeleitet und darauf basierend Workflows modelliert. Die einzelnen Workflowobjekte werden mit Businesslogik ausgestattet, die das Versenden von I4.0-Nachrichten über die in Abschnitt 3.2 genannte Operation „newMessage“, deren Empfang, sowie das Anstoßen der benutzerdefinierten Operationen aus dem Teilmodell „Message Participant“ übernehmen. Die einzelnen Zustandsmaschinen können jedoch genauso in dem monolithischen Ansatz in einem VWS-Server/-Repository oder in dem agentenbasierten Ansatz in einem Agenten-Typen programmiert werden. Vorteil der WMS ist jedoch, dass sie für den Entwickler der Zustandsmaschinen einfacher verständlich sind, da sie eine grafische Repräsentation der Zustandsmaschinen ermöglichen. Zudem stellen viele WMS-Lösungen bereits Schnittstellen zu Applikationen der Produktions-, Betriebs- und Unternehmensleitebene her, sodass übergreifende Interaktionsmuster einfacher zu implementieren sind.

5 Ausblick

Die vorgestellten Konzepte haben verschiedene Implementierungsmöglichkeiten für proaktive VWS in Hinsicht auf deren Architektur betrachtet. Die meisten Forschungsarbeiten bedienen sich dabei dem agentenbasierten Ansatz. Der monolithische und orchestrierende Ansatz sind bisher wenig untersucht. Zudem wurde die Integration

der I4.0-Sprache aus der VDI/VDE 2193-1 mit Hilfe der bestehenden VWS Spezifikationen betrachtet. Da nur wenige der genannten Arbeiten die Integration der I4.0-Sprache verfolgten wurde ein in [AR24] entwickeltes Konzept zur Übertragung der I4.0-Nachricht in dieser Arbeit vorgestellt. Die Lösung bietet mit dem VWS-Element „Operation“ einen leichtgewichtigen Ansatz für die Übermittlung von I4.0-Nachrichten über die bereits spezifizierte VWS-API und sollte in weiteren Arbeiten auf deren Allgemeingültigkeit hin untersucht werden. Auf den Einsatz des VWS-Elements „Event“ zur Nachrichtenübertragung sollte solange verzichtet werden, bis dieses ausreichend spezifiziert ist. Ansonsten werden die erarbeiteten Lösungen in der Praxis nur schwer Einzug halten.

Zukünftig sollten praktische Evaluierungen der hier vorgestellten Implementierungskonzepte einer proaktiven VWS erfolgen. Dabei sollten unter anderem die Anforderungen an Flexibilität, Performance, Integration von Zugriffskontrollen und Stabilität jedes vorgestellten Ansatzes untersucht werden. Zudem wird eine Spezifikation der proaktiven VWS benötigt, die einen Konsens über die Anforderungen an eine proaktive VWS schafft, um auch die in der Industrie 4.0 angestrebte Interoperabilität zu fördern und nicht zu behindern. Dies muss zudem zeitnah geschehen, damit vor allem die Implementierung der proaktiven VWS in der industriellen Umgebung ein Erfolg wird.

Literaturverzeichnis

[AR24] Verwaltungsschale für den Leitungssatz. Zugegriffen: 15. April 2024. [Online]. Verfügbar unter: https://arena2036.de/de/verwaltungsschale-fuer-den-leitungssatz

[BA24] BAFA - Modernisierung Fahrzeughersteller / Zulieferindustrie. Zugegriffen: 22. Januar 2024. [Online]. Verfügbar unter: https://www.bafa.de/DE/Wirtschaft/Handwerk_Industrie/Modernisierung_Fahrzeughersteller_Zulieferindustrie/modernisierung_fahrzeughersteller_zulieferindustrie_node.html

[BD19] A. Belyaev und C. Diedrich: Specification „Demonstrator I4.0-Language“ v3.0. 2019.

[BM24] B.-B. für W. und Klimaschutz: Das Gaia-X Ökosystem - Souveräne Dateninfrastruktur für Europa. Zugegriffen: 22. Januar 2024. [Online]. Verfügbar unter: https://www.bmwk.de/Redaktion/DE/Dossier/gaia-x.html

[CF21] Cardoso, R.C.; Ferrando, A.: A Review of Agent-Based Programming for Multi-Agent Systems. Computers 2021, 10, 16. https://doi.org/10.3390/computers10020016

[EB24] Eclipse BaSyx™. Zugegriffen: 15.April 2024. [Online]. Verfügbar unter: https://github.com/eclipse-basyx

[Ge15] M. Geirhos: Entwurfsmuster - Das umfassende Handbuch, 1. Aufl. Bonn: Rheinwerk Verlag, 2015.

[Ca23] G. P. Cainelli, P. Yazdani, L. Underberg, U. Jumar, and C. E. Pereira: Managing the 5G system based on production requirements using an Asset Administration Shell approach. IFAC-PapersOnLine, vol. 56, no. 2, pp. 2108–2114, 2023, doi: 10.1016/j.ifacol.2023.10.1113.

[Gr22] S. Grunau, M. Redeker, D. Göllner, and L. Wisniewski: The Implementation of Proactive Asset Administration Shells: Evaluation of Possibilities and Realization in an Order Driven Production. pp. 131–144, 2022, doi: 10.1007/978-3-662-64283-2_10.

[ID23a] Industrial Digital Twin Association, Hrsg.: Asset Administration Shell Specification - Part 1: Metamodel Schema. März 2023.

[ID23b] Industrial Digital Twin Association, Hrsg.: Asset Administration Shell Specification - Part 2: Application Programming Interfaces. Juni 2023.

[Is23] R. Islam u. a.: Erfahrungsbericht bei der Umsetzung der VWS Type 3 : Interaktionen in einer Maintenance-Anwendung. 2023, doi: 10.25673/111634.

[Ju23] S. Jungbluth, A. Witton, J. Hermann and M. Ruskowski: Architecture for Shared Production Leveraging Asset Administration Shell and Gaia-X. 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), Lemgo, Germany, 2023, pp. 1-8, doi: 10.1109/INDIN51400.2023.10218150.

[PI24] Manufacturing-X Initiative. Zugegriffen: 23. Januar 2024. [Online]. Verfügbar unter: https://www.plattform-i40.de/IP/Navigation/DE/Manufacturing-X/Initiative/initiative-manufacturing-x.html

[PI20] Plattform Industrie 4.0: Verwaltungsschale in der Praxis – Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen (Version 1.0). Berlin: BMWi, 2020.

[SLP22] L. Sakurada, P. Leitao and F. De La Prieta: Engineering a Multi-agent Systems Approach for Realizing Collaborative Asset Administration Shells. 2022 IEEE International Conference on Industrial Technology (ICIT), Shanghai, China, 2022, pp. 1-6, doi: 10.1109/ICIT48603.2022.10002770.

[SPL23] L. Sakurada, F. D. L. Prieta and P. Leitao: A Methodology for Integrating Asset Administration Shells and Multi-agent Systems. 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), Helsinki, Finland, 2023, pp. 1-6, doi: 10.1109/ISIE51358.2023.10227964.

[VD20a] VDI, Hrsg.: VDI/VDE 2193 Blatt 1- Sprache für I4.0-Komponenten - Struktur von Nachrichten. 2020.

[VD20b] VDI, Hrsg.: VDI/VDE 2193 Blatt 2 - Sprache für I4.0-Komponenten - Interaktionsprotokoll für Ausschreibungsverfahren. 2020.

[WA24] Digitales Engineering, WAGO Deutschland. Zugegriffen: 22. Januar 2024. [Online]. Verfügbar unter: https://www.wago.com/de/digitalisierung/digitales-engineering

[WI24] Digitaler Zwilling - WITTENSTEIN SE. Zugegriffen: 22. Januar 2024. [Online]. Verfügbar unter: https://www.wittenstein.de/de-de/unternehmen/digitalization-center/digitaler-zwilling/

[YH20] X. Ye and S. H. Hong: Toward the Plug-and-Produce Capability for Industry 4.0: An Asset Administration Shell Approach. no. December, 2020.