OPC UA als Basistechnologie zur Orchestrierung von Produktionssystemen

Orchestrierung von Diensten der Komponenten in Produktionssystemen mit Hilfe von OPC UA

Dr.-Ing. **M. Schleipen**, **J. Pfrommer**, Fraunhofer IOSB, Karlsruhe

Kurzfassung

Die steigende Anzahl von Produktvarianten zwingt Anlagenbetreiber dazu, Produktionsressourcen nicht nur für einzelne Anwendungen zu nutzen, sondern in unterschiedlichen Szenarien für verschiedene Produktvarianten angepasst einzusetzen. Die Orchestrierung bezeichnet das flexible Verbinden von einzelnen Diensten für einen definierten Zweck [1]. OPC UA [2] als Basistechnologie bietet die Möglichkeit, dass entsprechende Anbieter von Komponenten, die einen OPC UA Server enthalten, abgegrenzte Funktionalität in Form von Methoden kapseln. OPC UA kann dabei als generische, dienste-basierte Schnittstelle zur Orchestrierung eines entsprechenden Produktionssystems dienen, das aus mehreren Teilnehmern oder Komponenten besteht.

Abstract

The increasing number of product variants leads to multi-purpose production resources which are used for more than one scenario for different product variants. Orchestration describes the flexible combination of singe services for a defined purpose [1]. OPC UA as technology allows components which provide an OPC UA server to encapsulate delimited functionality in methods. OPC UA acts as generic, service-based interface for the orchestration of a production system which consists of multiple parties or components.

1. Einleitung

Im Kontext von Industrie 4.0 [3] sollen zusätzliche Funktionalitäten als Dienst für eine Komponente, z.B. eine Maschine oder ein Roboter, angeboten und nachgerüstet werden. Dies kann sich in neuen Geschäftsmodellen und Wertschöpfungssystemen widerspiegeln, muss aber durch die Technik entsprechend unterstützt und abgesichert sein. Um eine sichere Inbetriebnahme solcher zusätzlicher Dienste in der Produktionsumgebung zu gewährleisten, kann ein kombinierter OPC UA Server und Client mit einem passenden Informationsmodell genutzt werden, um beispielsweise die Aktivierung von Zusatzfunktionalität mittels Codeanalyse abzusichern. Der zusätzliche Dienst kann dabei auch auf entfernten Geräten ausgeführt

werden. OPC UA ermöglicht hierbei die effiziente Kollaboration verschiedener Produktionsressourcen.

Im vorliegenden Beitrag wird das Konzept der flexiblen Orchestrierung der Komponenten in Produktionssystemen mittels OPC UA aufgegriffen. Ein zentraler Orchestrierungsserver koordiniert das Verhalten innerhalb des Produktionssystems und fungiert als Bindeglied zwischen den Einzelkomponenten. Jede der Einzelkomponenten kapselt ihre Funktionalität [4] in einem Objekt im OPC UA Informationsmodell, welches auch Methoden enthält.

Der Ansatz wurde auf Basis von verfügbaren Hilfsmitteln umgesetzt. So dient die offene Implementierung open62541 der OPC UA Basisfunktionalität als plattform-unabhängige Grundlage für Orchestrierungsserver und Komponenten. Beschreibende XML-basierte Modelle des Produktionssystems werden als Informationsquelle für den entsprechenden Orchestrierungsserver eingesetzt. Die darin beschriebenen Eigenschaften der Komponenten können statischer Natur (vordefinierte Werte im Modell) oder dynamischer Natur (Platzhalter für spätere Variablen, die aus dem Prozess gespeist werden) sein. Um flexibel auf Änderungen reagieren zu können, wird davon ausgegangen, dass entsprechende Beschreibungen des Produktionssystems auch während des Betriebs des Servers verändert werden können. Hierfür wird open62541 [5] mit der aus dem Gaming-Bereich bekannten Scripting-Sprache Lua [6] kombiniert.

2. OPC UA Script

Als industrielles Kommunikationsprotokoll basierend auf einer plattform-unabhängigen Service-orientierten Architektur, wurde OPC UA im vorliegenden Anwendungsfall sowohl als generische Schnittstelle als auch als Framework genutzt, um verschiedene Produktionskomponenten (Hardware oder Software) und ihre Fähigkeiten und zugehörigen Methoden zu verbinden. Die Entwicklung komplexer technischer Lösungen, beispielsweise für OPC UA) erfordert heute viele Iterationen während der Entwicklung und bei Anpassungen nach der ersten Inbetriebnahme. Um die Iterationszyklen zu beschleunigen hat das Fraunhofer IOSB das in C99 verfasste Open Source SDK [7] in die Lua Programmiersprache [6] integriert. In Anlehnung an VBScript in der MS Office-Welt wurde das Ergebnis „UAScript“ getauft.

Haupteigenschaften von Lua sind die Kompaktheit und die einfache Erlernbarkeit. Es wird hauptsächlich für C++-basierte Gaming-Engines mit weichen Echtzeitkriterien verwendet. Darüber hinaus gibt es immer mehr Anwendungen aus dem Internet-of-Things Kontext, die Lua wegen der Geschwindigkeit und dem geringen Speicherbedarf wählen. Die Integration eines OPC UA SDKs in Lua bringt verschiedene Vorteile mit sich. Beispielsweise bietet sich dadurch ein einfacher Zugriff auf OPC UA Basis-Datentypen. Aber auch als Kommandozei-

leninterpreter für die schnelle Diagnose und Entwicklung zur Laufzeit lässt sich UAScript nutzen. So können technische Lösungen schnell exploriert werden und die komplexen Konzepte von OPC UA einfach zugänglich gemacht werden. Ein Beispiel zur Erzeugung eines OPC UA Servers in UAScript mit einem Methoden-Callback in die Scripting-Umgebung, sowie der Aufrufs der Methode mit einem Client findet sich in Bild 1 und 2.

UAScript reduziert als hochflexible Integrationsplattform die kognitive Belastung des Entwicklers durch folgende Mechanismen:

```
-- define a callback method
function hello_world(object_id, arg1)
    return "Hello " .. tostring(arg1.value)
end

-- initialize the server with a port number
server = ua.Server(48480)

-- add a method node
string_arg = ua.types.Argument()
string_arg.dataType = ua.types.String.typeId
string_arg.valueRank = -1
input_args = { string_arg }
output_args = { string_arg }
attr = ua.types.MethodAttributes()
attr.displayName = ua.types.LocalizedText("en_US", "Hello World")
attr.executable = true
browsename = ua.types.QualifiedName(1, "hello_world")
nodeid = ua.types.NodeId(1, "helloworld")
parent = ua.nodeIds.Objects
parent_reference = ua.nodeIds.HasComponent
server:addMethodNode(nodeid, parent, parent_reference, browsename,
                     attr, hello_world, input_args, output_args)

server:run()
```

Bild 1: Listing für einen Server mit einer Lua-Methode als Callback

```
client = ua.Client()
client:connect("opc.tcp://127.0.0.1:4840")

cmr = ua.types.CallMethodRequest()
cmr.objectId = ua.nodeIds.Objects
cmr.methodId = ua.types.NodeId(1, "helloworld")
cmr.inputArguments = { ua.types.Variant("Peter") }
res = client:call({cmr})
print(res)

client:disconnect()
```

Bild 2: Listing zum Aufruf der Methode mit einem Client

- Automatische Speicherverwaltung mittels Garbage Collection
- Transparente Umwandlung der nativen Lua Datentypen von und nach (der in C implementierten) OPC UA Datentypen
- Schnelle Entwicklung auf dem Zielsystem (kein Cross-Compile + Deploy Zyklus)
- Einfache Ausgabe (print) aller Lua und OPC UA Datentypen, sowie die Möglichkeit zur Introspektion verfügbarer Datentypen und Methoden

So wird die dynamische Erstellung eines Informationsmodells und von Kommunikationsverbindungen der Systemkomponenten zur Laufzeit erheblich vereinfacht. Dennoch bleibt die ursprüngliche Geschwindigkeit von Client und Server der in C implementierten Bibliothek erhalten. Lediglich Aufrufe von als Lua Funktion implementierten Methoden im Informationsmodell machen zur Laufzeit des Servers Gebrauch von der Scripting-Umgebung. UAScript kann darüber hinaus in weitere Anwendungen eingebettet werden. So hat das Fraunhofer IOSB den UAScript Interpreter in die V-Rep - virtual robot experimentation platform [8] eingebettet. So können flexibel UA Server und Clients zu entsprechenden Simulationen hinzugefügt werden, die die Kommunikation mit externen Komponenten übernehmen, z.B. einen steuernden Zugriff auf das Simulationsmodell von außen erlauben.

Der Orchestrierungsserver, welcher auch OPC UA Clients zur Einbindung entfernter Komponenten enthält, wurde basierend auf UAScript entwickelt, um Informationen plattform- und systemunabhängig zur Verfügung stellen und flexibel agieren zu können. Er kann auf einem Windows-basierten Desktop-PC ebenso laufen, wie auf kleineren Systemen, z.B. Raspberry Pi oder Beaglebone Black (mit Linux als Betriebssystem).

3. OPC UA Informationsmodell

Beschreibende XML-basierte Modelle des Produktionssystems werden als Informationsquelle für den entsprechenden Orchestrierungsserver eingesetzt.

Diese enthalten neben der möglichen Funktionalität der Komponenten auch Informationen über die Eigenschaften der Komponenten. Diese können statischer Natur (vordefinierte Werte im Modell) oder dynamischer Natur (Platzhalter für spätere Variablen, die aus dem Prozess gespeist werden) sein.

Sowohl Hardware-Komponenten (Systemteilnehmer) als auch die verfügbare Software-Funktionalität werden als Objekte im Informationsmodell des zentralen Orchestrierungsservers repräsentiert.

Zusätzlich zu den Objekten werden deren Eigenschaften und Fähigkeiten über Variablen und Methoden im Informationsmodell repräsentiert. Die Fähigkeiten der Produktionsressourcen

müssen in einem solchen flexiblen Szenario zwingend beschrieben sein, um mit den Software- und Hardwarekomponenten als Blackbox interagieren zu können (siehe [9]).
Dabei können beispielsweise so genannte Companion Spezifikationen der OPC UA zum Einsatz kommen. Hier bieten sich verschiedene an, „AutomationML für OPC UA“ [10] kombiniert aber ein recht generisches Modell der Produktionsressourcen, produzierten Produkte und Produktionsprozesse auf hoher Abstraktionsebene mit OPC UA und eignet sich daher sehr gut. Um ein entsprechendes Informationsmodell oder Teile davon (auch zur Laufzeit) integrieren zu können, muss entweder eine entsprechende Modelltransformation (siehe [11]) in das XML-Format der OPC UA erfolgen oder es ist ein spezifischer Importer für den Modelltyp nötig, z.B. mit RobotML (siehe [12]).

4. (Modell-)Verifikation

Bei einer solch flexiblen Orchestrierung stellen sich allerdings auch Fragen nach der Sicherheit der verteilten zusätzlichen Funktionalität. Um diese zu gewährleisten, sollten weitere Mechanismen vor der Freischaltung einer Funktionalität erfolgen. Sei es, dass Vertrauensketten die Sicherheit der Funktionalität belegen und diese über z.B. die Identität der Software-Komponente verifiziert werden können (siehe [13]). Oder aber, dass zusätzliche Codeanalyse-Methoden (z.B. statische Codeanalyse), Modellverifikation oder weitere Absicherungsmechanismen in den Orchestrierungsserver als eigenständige Softwarekomponenten integriert werden können (siehe [12]). Im Rahmen der beschriebenen Anwendung wurde Frama-C [14] zur Verifikation von dynamisch hinzuladbaren Software-Komponenten verwendet, welche in C verfasste Codeanteile umfassen. Zusätzlich zu den Verifikationsmechanismen muss die Freischaltung neuer Funktionalitäten rechtlich bedingt von einem Anwender bestätigt werden. Dies erfolgt über gängige HMI-Anwendungen, die über eine OPC UA-Schnittstelle verfügen.

5. Anwenderschnittstelle

Ein Anwender kann in Kombination mit dem Orchestrierungsserver die Eigenschaften von OPC UA nutzen, um einen Gesamtüberblick über das Produktionssystem, dessen Status und Ergebnisse bspw. der Analysemethoden zu erhalten. Gleichzeitig aber können auch mit verschiedenen OPC UA Clients auf unterschiedlichen Endsystemen (Mobil, Desktop) die Informationen zu den Einzelkomponenten für den Nutzer sichtbar gemacht werden. Im vorliegenden Fall basiert die einfache Visualisierung zum Bedienen und Beobachten (dargestellt in Bild 3) auf dem Fraunhofer IOSB-eigenen System ProVis.Visu® [15], das auf Windows

Betriebssystemen läuft und über einen OPC UA Client basierend auf dem Unified Automation C++ OPC UA Toolkit [16] verfügt.
Andere Standard-Clients und Systeme können ebenso genutzt werden. Beispielsweise um mobile Endgeräte nutzen zu können. Genau dies ist einer der großen Vorteile von OPC UA. Kombiniert man nun OPC UA mit AutomationML können die dort im Modell enthaltenen Informationen ebenso als Quelle für die automatisierte Erzeugung der Visualisierung genutzt werden (siehe [17] und [18]).

6. Informationsverteilung

Der zentrale Orchestrierungsserver koordiniert das Verhalten innerhalb des Produktionssystems und fungiert als Bindeglied zwischen den Einzelkomponenten. Jede der Einzelkomponenten „versteckt“ ihre Funktionalität hinter OPC UA Methoden, die sie anbietet.
Der Orchestrierungsserver versorgt die einzelnen Teilnehmer mit den sie betreffenden Teilmodellen bzw. ist dafür zuständig, erhaltene Teilmodelle in ein Gesamtmodell zu integrieren. Er muss in jedem Fall über vollständige Informationen zu den Teilmodellen des Produktionssystems verfügen.

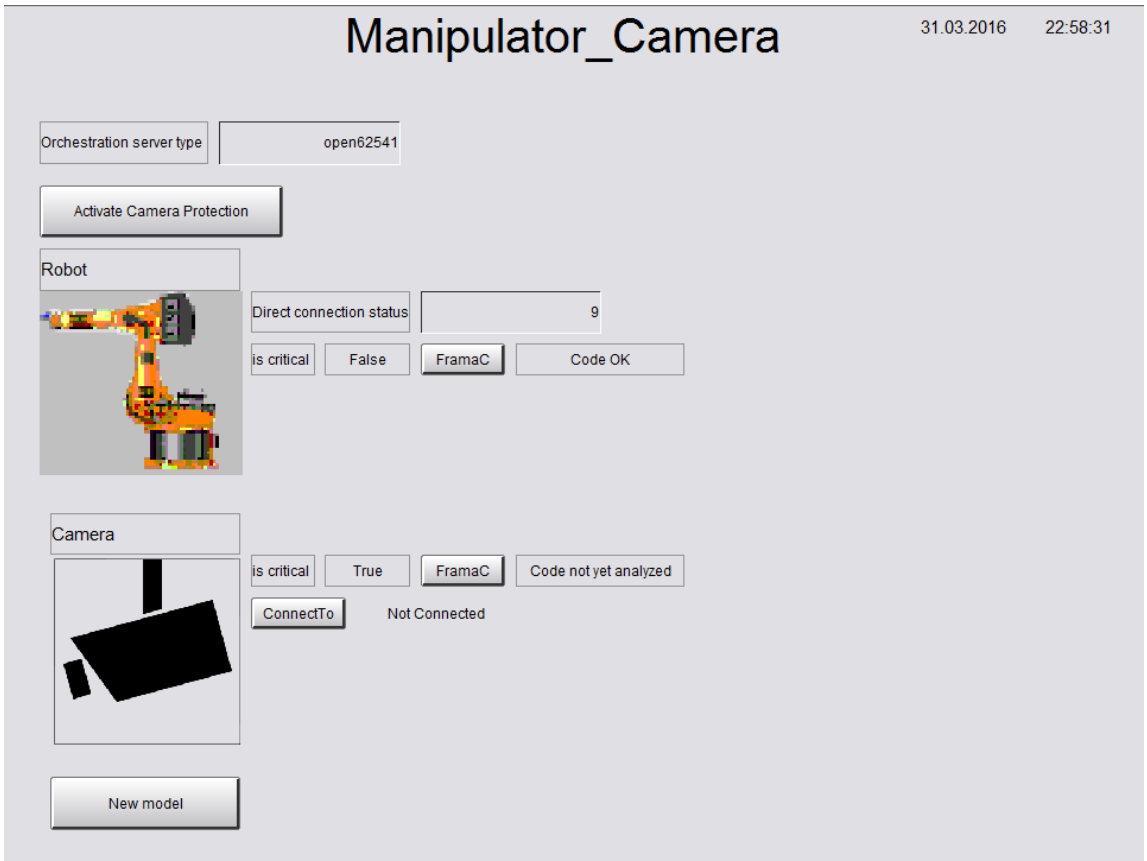

Bild 3: Visualisierung für Produktionszelle bestehend aus Roboter und Kamera

Um auf die Informationen (Variablen mit entsprechenden Werten aus dem Prozess) und die Funktionalität der Teilnehmer (Methoden, die mit entsprechenden Parametern ausgeführt werden können) zugreifen zu können, implementiert der Orchestrierungsserver neben der Server- auch eine entsprechende OPC UA Client-Funktionalität. Neben der Verteilung der Informationen an die Teilnehmer, verfügt der Orchestrierungsserver über eigene Methoden beispielsweise zur Analyse, kann aber auch Teilnehmer des Systems auffordern, bestimmte Funktionalitäten auszuführen.
Neben diesen eher fachlichen Diensten, werden im GMA Fachausschuss 7.21 „Industrie 4.0“ aktuell auch Basis-Dienste erarbeitet, die eine Industrie4.0-Komponente zur Verfügung stellen sollte. Auch diese können der Außenwelt über OPC UA „angeboten“ werden.

7. Anwendungsfall

Der Orchestrierungsserver wurde aktuell in 2 Anwendungsfällen genutzt und weiterentwickelt. In einem Anwendungsfall (siehe [12]) wurde ein kollaborativer Roboter zur Laufzeit mit einer zusätzlichen externen Kamerakomponente ausgestatten, die zur Kollisionserkennung mit entfernten Objekten, z.B. in das Produktionssystem eindringende Personen, genutzt werden kann und deren Integration in das Produktionssystem mittels statischer Codeanalyse abgesichert wird. Hier wurde als Informationsmodell RobotML integriert.
Im zweiten Anwendungsfall wurden zusätzliche Software-Auswertefunktionalitäten integriert (siehe [19]), die basierend auf dem Informationsmodell – in diesem Fall AutomationML – die relevanten Informationen über die Struktur der Anlage extrahieren und die Ergebnisse wieder ins Informationsmodell zurück schreiben.

8. Zusammenfassung und Ausblick

Das erarbeitete Konzept konkret umgesetzt und bietet die Grundlage für die Umsetzung vieler weiterer Anwendungsfälle, gerade im Kontext Industrie 4.0. Das Konzept ist nicht mit konkreten Modellierungssprachen oder Programmierhilfen gekoppelt, sondern lässt sich flexibel auf andere Beschreibungen (z.B. AutomationML) oder andere Programmierhilfen (z.B. C++ SDK von Unified Automation) übertragen. Es werden hierdurch die Möglichkeiten von OPC UA in der flexiblen Orchestrierung von Diensten mittels kombinierten OPC UA Server und Clients aufgezeigt. Die Kapselung von Funktionalität und deren Beschreibung ist Teil des Konzepts der I4.0 Komponenten (siehe [3]). Durch die Kopplung des OPC UA Toolkits open62541 mit der Scripting-Sprache Lua wird ein schnelles Prototyping und Deployment ermöglicht, das in vielen Anwendungen dringend benötigt wird. Als weitere Schritte können nun die zur Laufzeit hinzugefügten Funktionalitäten als Fähigkeiten beschrieben und mittels

Planungsverfahren dynamisch zur Erreichung von Zielen in der Produktion kombiniert und orchestriert werden [9].

Literatur

[1] I40-Glossar, http://i40.iosb.fraunhofer.de/FA7.21%20Begriffe, Stand 31.3.2016.

[2] The OPC Unified Architecture (UA), OPC Foundation, https://opcfoundation.org/about/opc-technologies/opc-ua/, Stand 31.3.16.

[3] Umsetzungsstrategie Industrie 4.0, https://www.bitkom.org/Bitkom/Publikationen/Umsetzungsstrategie-Industrie-40.html, 15.4.2015.

[4] Julius Pfrommer, Denis Stogl, Kiril Aleksandrov, Viktor Schubert, Björn Hein. Modelling and Orchestration of Service-Based Manufacturing Systems via Skills. In: Emerging Technologies & Factory Automation (ETFA), 2014 IEEE 19th Conference on, September 2014.

[5] Florian Palm, Sten Grüner, Julius Pfrommer, Markus Graube, Leon Urbas. open62541 - der offene OPC UA-Stack. KommA 2014 : 5. Jahreskolloquium Kommunikation in der Automation ; 18.11.2014, Lemgo, 2014.

[6] Lua, http://www.lua.org, Stand 31.3.16.

[7] open62541: An open source and free c (c99) implementation of opc ua stack., TU Dresden PLT, Fraunhofer IOSB, RWTH-PLT, http://open62541.org/, Stand 31.3.16.

[8] V-Rep, http://www.coppeliarobotics.com/, Stand 31.3.16.

[9] Pfrommer, J., Stogl, D., Aleksandrov, K., Escaida Navarro, S., Hein, B., & Beyerer, J. (2015). Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. at-Automatisierungstechnik, 63(10), S. 790-800.

[10] Companion Specification „AutomationML for OPC UA“, https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-automationml/, Stand 31.3.16.

[11] Robert Henßen, Miriam Schleipen: Online-Kommunikation mittels OPC-UA vs. Engineering-Daten (offline) in AutomationML - Eine Möglichkeit der Integration und Kombination. Automation, VDI-Verlag, 2014 (VDI-Berichte 2231), ISBN: 978-3-18-092231-7, pp.59-74, 2014.

[12] Selma Azaiez, Michael Boc, Loic Cudennec, Max Da Silva Simoes, Jens Haupert, Selma Kchir, Xenia Klinge, Wael Labidi, Karima Nahhal, Julius Pfrommer, Miriam Schleipen, Christian Schulz, Thibaud Tortech: Towards Flexibility in Future Industrial Manufacturing: A Global Framework for Self-Organization of Production Cells. The Second International Workshop on Recent Advances on Machine-to-Machine Communications, RAMCOM 2016 (accepted), May 23-26, 2016, Madrid, Spain.

[13] SecurePLUGandWORK, www.secureplugandwork.de, Stand 31.3.16.

[14] Cuoq, Pascal, et al. "Frama-C - A Software Analysis Perspective." Software Engineering and Formal Methods. Springer Berlin Heidelberg, 2012. 233-247.

[15] Fraunhofer IOSB: ProVis.Visu, www.iosb.fraunhofer.de/?ProVis.Visu®, Stand 31.3.16.

[16] Unified Automation, https://www.unified-automation.com/, Stand 31.3.16.

[17] Miriam Schleipen: OPC UA supporting the automated engineering of production monitoring and control systems. Proceedings of 13th IEEE International Conference on Emerging Technologies and Factory Automation ETFA, 15.-18.9.2008, Hamburg, Germany, S.640-647, 2008

[18] Miriam Schleipen, Michael Okon, Robert Henßen, Thomas Hövelmeyer, Andreas Wagner, Gerhard Wolff, Halit Demir, Matthias Jentsch, Marco Gebhardt, Thomas Stoll, Dennis Asi: Monitoring and control of flexible transport equipment - Towards Industrie 4.0 based on standards, at - Automatisierungstechnik. Band 63, Heft 12, Seiten 977–991, ISSN (Online) 2196-677X, ISSN (Print) 0178-2312, DOI: 10.1515/auto-2015-0013, December 2015

[19] Miriam Schleipen, Christian Kühnert, Michael Okon, Robert Henßen, Tino Bischoff: MoMo - Mobile Monitoring und smarte Datenanalyse basierend auf offenen Standards, Automation 2015.