Ansatz zur modellgetriebenen Entwicklung flexibler Automatisierungssysteme durch Serviceorientierung

Jan-Philipp Schmidt, Andreas Zeller, Michael Weyrich
IAS – Institut für Automatisierungstechnik und Softwaresysteme
Universität Stuttgart
Pfaffenwaldring 47
70550 Stuttgart
Tel.: 0711 685 67301
Fax: 0711 685 67302
E-Mail: ias@ias.uni-stuttgart.de

Abstract:

Im Rahmen dieser Veröffentlichung wird die Herausforderung der Komplexität verteilter Systeme thematisiert und ein Ansatz zur Komplexitätsbeherrschung vorgestellt. Auf Basis einer mikrocontrollerbasierten Softwarearchitektur wird eine modellgetriebene Entwicklung umgesetzt. Die gekapselten Modelle bilden dabei die Grundlage einer serviceorientierten Architektur.

Stichworte: Modellgetriebene Entwicklung, SOA, Verteilte Systeme, Koordination, Kommunikation, Zustandsautomaten, Kapselung

1 Motivation

Individuelle Kundenwünsche führen zu einem disruptiven Wandel der Produktionssysteme. Dezentral gesteuerte und flexibel vernetzbare Automatisierungssysteme ermöglichen eine kostengünstige Bedienung dieses volatilen Marktes. Diese Dezentralisierung resultiert in einer höheren Komplexität in der Kompositionsstruktur und der Kommunikationsarchitektur. Um den Aufwand für das Engineering solcher Systeme zu reduzieren, muss die Komplexität beherrschbar gemacht werden. Im Rahmen dieser Veröffentlichung wird ein Ansatz präsentiert, der beide Arten der Komplexität berücksichtigt, mit dem Ziel, diese zu verringern.

2 Stand der Technik

2.1 Verteilte Systeme

Ein verteiltes System ist „ein Zusammenschluss unabhängiger Computer, die sich für den Benutzer als ein einziges System präsentieren.“ [TaSt2007] Klassische Anwendungsbereiche sind Multicore-Controller, Cloud und Grid-Computing.

Nach dieser Definition führt eine Auslagerung der Steuerung ins Feld zu einem verteilten System und somit zu einem neuen Anwendungsfeld im Bereich der Anlagenautomatisierung. Die modulare Steuerung der Anlage unterscheidet sich für den Anwender nach außen hin nicht von einem zentralen Automatisierungssystem. Trotzdem sind die Steuerungen unabhängig in dem Sinne, dass sie keine gemeinsamen Speicher besitzen und Informationen daher über Kommunikationskanäle austauschen müssen.

Erhöhte Komplexität aufgrund der neuen Kompositionsstruktur

Die verteilte Steuerung führt durch Auslagerung der Steuerung in die Feldebene zu einer flacheren Hierarchie. Dabei wird jeder Teilprozess durch ein eigenes Steuerungsmodul automatisiert. Die Komplexität dieser neuen Strukturen resultiert aus der Heterogenität der Prozesse und der Steuerungshardware. Es müssen also viele Steuerungen für verschiedene Teilprozesse entwickelt werden.

Erhöhte Komplexität aufgrund der neuen Kommunikationsarchitektur

Neben der Herausforderung, eine Vielzahl an heterogenen Steuerungen zu realisieren, ergibt sich, verglichen mit zentralen Steuerungen, eine höhere Komplexität der Kommunikation und Koordinierung zwischen den verteilten Steuerungen. Für die Kommunikation steht je nach Anforderungen an Datenvolumen, Reaktionszeit und Echtzeitverhalten eine Vielzahl an Bussystemen zur Verfügung. Eine größere Herausforderung stellt die Koordination des Systems dar. Hier darf keinesfalls eine statische Definition von Botschaften erfolgen, da sonst die Flexibilität und die Unabhängigkeit der Module verloren gingen. Demzufolge muss das Ziel eine dynamische Vernetzung und Koordinierung der Aufgaben zwischen den verteilten Steuerungen sein.

2.2 Modellgetriebene Entwicklung

In der Literatur finden sich Ansätze der modellbasierten Entwicklung für Automatisierungssysteme [FaVo2015]. Diese beziehen sich, im Gegensatz zu diesem Ansatz, auf einheitliche Entwurfsmuster, um eine durchgängige Modellierung in den Entwicklungsphasen zu erhalten, und nicht auf die Generierung von lauffähigem Code aus den Modellen.

Die Automobilindustrie stand Anfang dieses Jahrtausends vor ähnlichen Problemen. Softwaretechnische Innovationen im Automobil wurden zunächst in neuen Steuergeräten umgesetzt. Betrachtet man das Fahrzeug als automatisiertes System, kann das Steuergerätenetzwerk als verteiltes Automatisierungssystem interpretiert werden. Die Komplexität führte dazu, dass OEMs und Zulieferer sich auf die Referenzarchitektur AUTOSAR einigten. Diese inzwischen etablierte Architektur dient der Komplexitätsbeherrschung, indem modellgetriebene Entwicklung und Virtualisierung durch Standardisierung von Basissoftware-Modulen und Schnittstellen ermöglicht werden. [AUTO2015]

Die Modelle, die zur Codegenerierung genutzt werden, bilden die Basis, um das Testen in frühen Phasen zu ermöglichen. Um bereits den Entwurf abzusichern, können die Modelle direkt in einer Simulationsumgebung des Modellierungstools gegen die Anforderung getestet werden.

Die Rahmenbedingungen für verteilte Systeme der Fahrzeugindustrie und der industriellen Automatisierungstechnik sind aber nur bedingt vergleichbar. Steuergerätenetzwerke in Fahrzeugen sind sehr statisch, weshalb Änderungen nur mit großem Entwicklungsaufwand durchführbar sind. Im Gegensatz dazu ist bei der Feldgerätekommunikation der industriellen Automatisierungstechnik ein höchst dynamisches Netzwerk gefordert, um Eigenschaften wie Flexibilität und Wandelbarkeit gewährleisten zu können.

2.3 Serviceorientierte Architektur

Zur Gewährleistung dieser Dynamik zur Laufzeit ist die Berücksichtigung weiterer Technologien notwendig. Im Bereich der Internettechnologien werden dynamisch verteilte IT-Systeme schon über 15 Jahre über das Paradigma der serviceorientierten Architektur (SOA) strukturiert [Gartner96], was zunehmend in der industriellen Automatisierungstechnik als Kommunikationsarchitektur der Zukunft diskutiert wird. So beschreibt der VDI/VDE-GMA-Statusreport der Referenzarchitektur Industrie-4.0 (RAMI4.0) die Industrie-4.0-konforme Kommunikation einer Industrie-4.0-Komponente auf Basis einer SOA – z.B. OPCUA [RAMI40]. Die Dynamik der Kommunikationsarchitektur ergibt sich aus der Kapselung der Funktionalität in einem Service, standardisierten Kommunikationsschnittstellen und einer losen Kopplung der Services. So können heterogene Systeme zur Laufzeit dynamisch gebunden und zu höheren Services orchestriert werden.

Es gibt bereits serviceorientierte Protokolle, wie OPC-UA, die in der Automatisierungstechnik eingesetzt werden. Da diese Protokolle aus dem Bereich der Internettechnologien kommen, setzen sie auf dem Ethernet-Protokoll auf [IEC 62541-7]. Im Bereich der industriellen Automatisierungstechnik gibt es aufgrund unterschiedlichster Anforderungen in Bezug auf Datendurchsatz und Echtzeit verschiedenste Bussysteme. Daher wurde ein Ansatz entwickelt, der eine SOA busunabhängig umsetzen kann.

Aufgrund hoher Heterogenität der Steuerungshardware ist die Entwicklung einer verteilten Steuerung gemäß der SOA-Struktur aber sehr komplex. Zur Simplifizierung des Entwicklungsprozesses fehlen Methoden, um Anwendungen hardwareunabhängig und einfach testbar entwickeln zu können. Ein Konzept zur hardwareunabhängigen Erstellung der Applikation für eine SOA-Struktur wird im Folgenden vorgestellt.

3 Konzept

Die Komplexität in Kompositionsstruktur und Kommunikation bei der Entwicklung verteilter Steuerungen wird in diesem Konzept adressiert. Dafür werden die bewährten Ansätze der modellgetriebenen Entwicklung und SOA, wie in Abbildung 1 dargestellt, in ein ganzheitliches Konzept integriert. Schnittstellen sind dabei die gekapselten Funktionen, die modellgetrieben entwickelt werden und die Basis für die Services darstellen. Im Folgenden werden die Teilaspekte des Konzeptes erläutert.

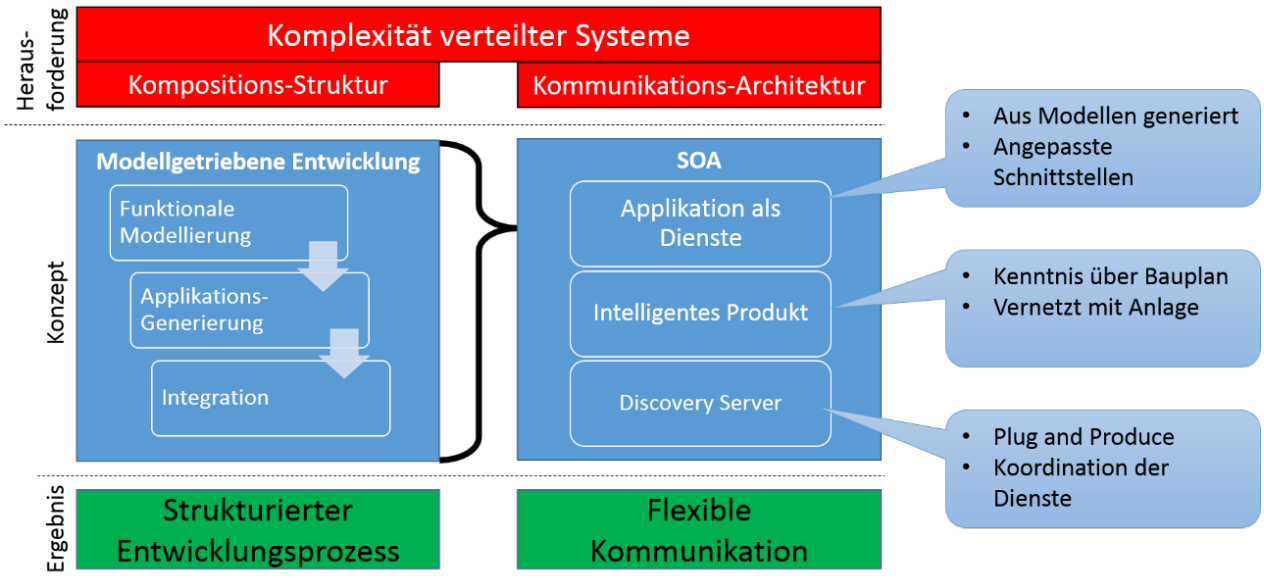

Abbildung 1: Konzept zur Komplexitätsbeherrschung

3.1 Modellgetriebene Entwicklung

Die Komplexität der Kompositionsstruktur kann durch Umsetzung einer an AUTOSAR angelehnten Softwarearchitektur bewältigt werden. Die Steuerungsaufgaben sind durchaus vergleichbar. So benötigt man als elementare Funktionen Input und Output sowie die Kommunikationsfähigkeit. Dabei müssen Echtzeitanforderungen eingehalten werden. Diese Rahmenbedingungen werden durch AUTOSAR abgedeckt. Ein Echtzeitbetriebssystem, der IO-Stack sowie der Kommunikationsstack werden in jeder Steuerung benötigt und können somit standardisiert werden. Daher wird wie bei AUTOSAR eine modulare, wiederverwendbare Basissoftware entworfen. Das Wiederverwendungskonzept ist der erste Schritt zur Komplexitätsbeherrschung.

Die Laufzeitumgebung (Runtime-Environment RTE) stellt eine Schnittstellenabstraktion dar und ermöglicht somit eine modellgetriebene Entwicklung. Aus funktionalen Modellen können für die standardisierten Schnittstellen Softwarekomponenten generiert werden. So können die Ergebnisse der Entwurfsphase weiter verwendet und eine fehleranfällige Implementierung vermieden werden.

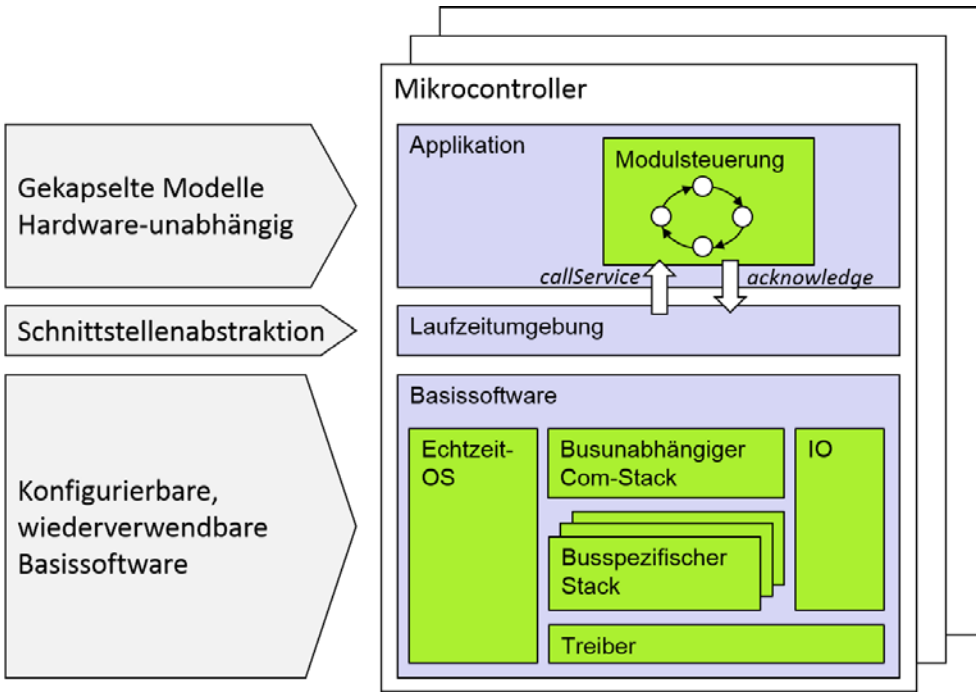

Abbildung 2: Embedded Software-Architektur

Mit der in Abbildung 2 dargestellten, am IAS entworfenen Architektur [ScWe2015] lässt sich der Aufwand für die Implementierung der verteilten Steuerungen auf die grafische Modellierung der Applikation und die Konfiguration der Basissoftware reduzieren.

Die Generierung aus der grafischen Modellierung vereinfacht nicht nur den Entwicklungsprozess, sondern stellt auch die Grundlage für die Koordinierung dar. Durch die Codegenerierung entstehen unabhängige gekapselte Funktionen, die jeweils einen Teil des zu automatisierenden Prozesses steuern.

3.2 Erweiterung der Komponentenarchitektur um intelligente Produkte und Discovery-Server

Dienste entsprechen gekapselten Funktionen mit wohldefinierten Schnittstellen nach außen. So reicht die Kenntnis über die Schnittstellen eines Dienstes aus, um mit diesem zu interagieren, ohne dessen Implementierung berücksichtigen zu müssen. Diese implementierungsunabhängige Kommunikation ermöglicht eine flexible Kommunikationsstruktur zwischen den Automatisierungskomponenten des verteilten Systems. Die verteilten Steuerungen des Produktionssystems fungieren dabei als Server, welche ihre Fähigkeiten in Form von Diensten anbieten.

Zur Realisierung einer SOA bedarf es neben den Servern eines Clients, welcher Dienste anfordert und den dazu geeigneten Server aufruft. Das intelligente Produkt, welches seinen Bauplan kennt, stellt einen solchen Client dar. Es ruft, dem Bauplan entsprechend, die Dienste auf, welche sich für den folgenden Prozessschritt eignen.

Da bei einer flexiblen Struktur die Server, welche den gewünschten Dienst anbieten, meist nicht von vornherein bekannt sind, muss, neben dem klassischen Server und Client, die Architektur um eine weitere Komponente erweitert werden – einem Discovery Server. Dieser besitzt eine

fest definierte ID, die systemweit bekannt ist. Zur Gewährleistung von Plug & Produce registrieren sich die Server beim Discovery-Server. Dieser verwaltet in einer Liste die verfügbaren Server und die Dienste, welche angeboten werden. Dies gewährleistet eine einfache Erweiterbarkeit und Rekonfigurierbarkeit des Automatisierungssystems. Die Schnittstellen der Komponenten sind in Abbildung 3 dargestellt. Da die Grundlage die generierten gekapselten Funktionen sind, kann die SOA modellgetrieben entwickelt werden. Neben den gekapselten und generierten Funktionen, die in Form von Services angesteuert werden, können auch der Client und Discovery-Server modellgetrieben entwickelt werden. Die Applikation, die durch weiße Rechtecke in Abbildung 3 dargestellt ist, kann dabei durch die Hardwareabstraktion der modellgetriebenen Entwicklung hardwareunabhängig entworfen werden.

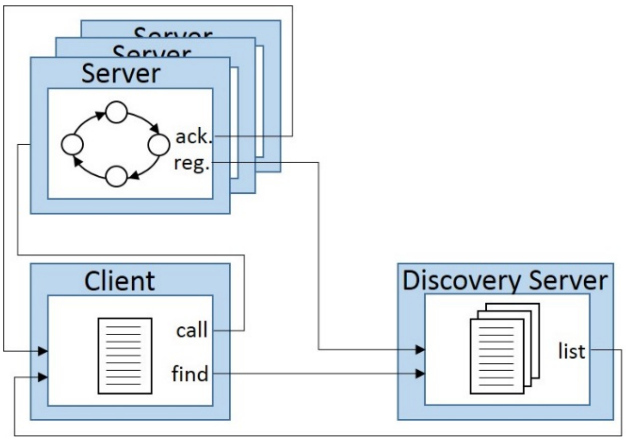

Abbildung 3: Kommunikationsstruktur der Komponenten eines verteilten Systems

Die Aufrufmodalitäten eines Services durch einen Client in einem dynamischen Netzwerk ist in Abbildung 4 dargestellt. Um eine Liste der Server zu erhalten, welche den gewünschten Service anbieten, wendet sich der Client an den Discovery-Sever. Anhand der Liste wählt der Client entsprechend prädefinierter Kriterien den am besten geeigneten Service und fragt diesen an. Dies ermöglicht das Finden und dynamische Binden eines Clients zu einem Server zur Laufzeit. Da die Grundlage die generierten gekapselten Funktionen sind, kann die SOA modellgetrieben entwickelt werden. Neben den gekapselten und generierten Funktionen, die in Form von Services angesteuert werden, können auch der Client und Discovery-Server modellgetrieben entwickelt werden.

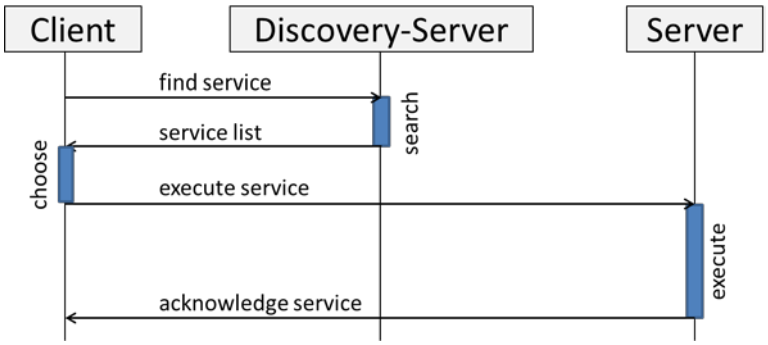

Abbildung 4: Kommunikationsstruktur der Aufrufmodalität eines Dienstes

4 Validierung

4.1 Szenario eines verteilten Produktionssystems

Das Konzept lässt sich an modularen Anlagen umsetzen. Am IAS wird zur Evaluation ein modulares Produktionssystem der Firma Festo Didactic eingesetzt. Es besteht aus drei Transferlinien und sechs Modulen zur Bearbeitung. Jede Transferlinie und jedes Modul wird jeweils durch eine Steuerung geführt. Auf allen Steuerungen läuft die gleiche Basissoftware, die für die jeweiligen Anforderungen konfiguriert wird. Der Aufbau des Demonstrators ist in Abbildung 5 dargestellt.

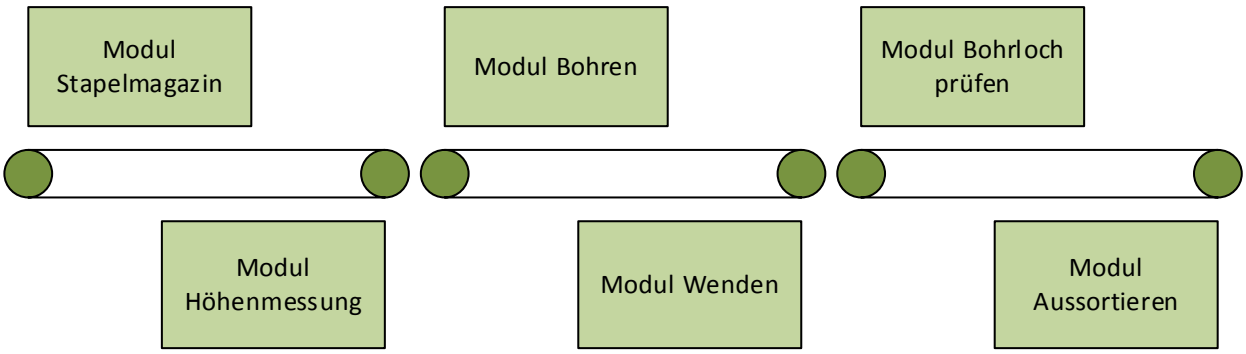

Abbildung 5: Demonstrator

Für die Applikation wird von jedem Modul ein funktionales Modell in Matlab Stateflow erstellt. Der Embedded Codegenerator ermöglicht eine Code-Generierung. Durch Schnittstellenanpassung kann jede so generierte Softwarekomponente als Dienst abgerufen werden. Bei einem Produktionsauftrag wird ein Intelligentes Produkt mit Kenntnis über seinen Bauplan erzeugt, das die Dienste dann abrufen kann. Dabei können alle Module, welche die Domänen Bearbeitung Logistik und Qualitätssicherung abdecken, einheitlich über Zustandsautomaten modelliert werden.

5 Zusammenfassung und Ausblick

Im Gegensatz zu zentralen Systemen weisen verteilte Systeme eine höhere Komplexität in der Kompositionsstruktur und der Kommunikationsarchitektur auf. Zur Reduktion der Komplexität, verursacht durch die Kompositionsstruktur, bietet sich die modellgetriebene Entwicklung an. Dieses Vorgehen ist in der Fahrzeugindustrie schon etabliert. Im Gegensatz zu verteilten Systemen im Fahrzeug, werden im Bereich der industriellen Automatisierungstechnik andere Anforderungen, wie beispielsweise Flexibilität, gestellt. Durch Anpassung der Schnittstellen der gekapselten Funktionen wird das SOA-Paradigma ermöglicht und es werden so bewährte Konzepte der Automobilindustrie auf die Automatisierungstechnik übertragen.

Im Gegensatz zum Einsatz von OPC-UA bleibt der hier aufgeführte Ansatz aufgrund der Basissoftware-Architektur busunabhängig. Somit bleibt die Echtzeitfähigkeit erhalten, die beim Einsatz von Ethernet verloren ginge.

Das bisherige Konzept betrachtet die modellgetriebene Modellierung von Diensten und weist ein Minimalbeispiel einer SOA auf. Dabei ruft das intelligente Produkt direkt einen Server auf, welcher den Prozessschritt ausführt. In einem nächsten Schritt soll die Steuerungsstruktur innerhalb eines Steuergeräts modellgetrieben gemäß eines SOA-Paradigmas entworfen werden. Dies erlaubt die Skalierbarkeit der SOA-Architektur bis auf die Sensor- / Aktorebene, was eine weitere Flexibilisierung innerhalb der Steuerung bewirkt. Dabei kann ein durchgängiges Modellierungskonzept auf verschiedenen Abstraktionsebenen angewendet werden.

6 Literatur

[TaSt2007] Tanenbaum, A.; van Stehen, M.: Verteilte Systeme: Prinzipien und Paradigmen, 2. Auflage, Pearson Studium – IT, 2007

[FaVo2015] Fay, A; Vogel-Heuser B. et al.: Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns, The Journal of Systems and Software 101, page 221-235, 2015

[Auto2015] http://www.autosar.org/, abgerufen am 24.11.2015

[Gartner96] Gartner (1996a): Research Note SPA-401-068 „Service Oriented" Architectures, Part 1&2, 12 April 1996

[RAMI40] VDI / VDE GMA, ZVEI: Statusreport – Referenzarchitekturmodell Industrie 4.0 (RAMI4.0), April 2015

[IEC 62541-7] IEC 62541-7:2015: OPC Unified Architecture – Part 7: Profiles

[ScWe2015] Schmidt, J.; Weyrich, M.: Entwurf einer Softwarearchitektur für dezentrale mikrocontrollerbasierte Steuerungssysteme, Automation Kongress, Baden-Baden, 12.06.2015