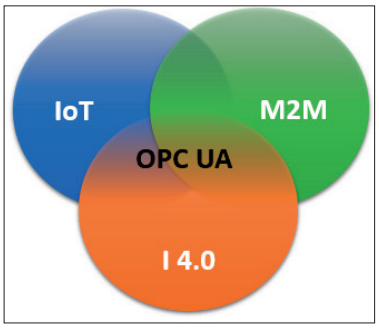
OPC UA IN DER PRAXIS

Die Lücke schließen

Die Open Platform Communications Unified Architecture ermöglicht eine sichere und systemübergreifende Kommunikation zwischen Maschinen – über die Cloud.

Vernetzung und Digitalisierung sind mittlerweile zentrale Begriffe in allen Branchen, ja selbst schon im Alltagsleben. In der Industrie vollzieht sich die Entwicklung inzwischen unter der Überschrift „Industrie 4.0“. Auch Unternehmen des Maschinen- und Anlagenbaus werden immer stärker mit den Konzepten von Industrie 4.0 konfrontiert; Daten produzieren diese Maschinen genügend. Doch was genau soll mit diesen Daten passieren, und wie kommen sie von Maschine zu Maschine oder sogar in die Cloud? Genau hier setzt das Konzept von OPC UA an, der Open Platform Communications Unified Architecture [1]. Sie hat sich als ein De-facto-Standard für die hersteller- und plattformunabhängige industrielle Kommunikation entwickelt.



Open Platform Communications Unified Architecture ist der Standard für die Schnittmenge von IoT, M2M und I4.0 (Bild 1)

OPC UA in der Theorie

Wer sich mit dem Thema OPC UA beschäftigt, stößt unweigerlich auf Begriffe wie M2M, IoT oder Industrie 4.0. OPC UA ist dabei eine Schnittmenge dieser Bereiche (Bild 1). Das Konzept soll einen sicheren, zuverlässigen und herstellerunabhängigen Transport von Daten und Informationen von Sensoren zu Maschinen, zwischen Maschinen oder Geräten bis hin zum Leitsystem oder sogar in die Cloud gewährleisten.

OPC UA definiert, wie dabei die Kommunikation zwischen den Parteien aussehen soll. Es handelt sich also um eine Art Framework für industrielle Interoperabilität ohne Abhängigkeiten zu proprietären Technologien. Dadurch ist auch ein Austausch von Daten zwischen Maschinen unterschiedlicher Hersteller einfach möglich.

OPC UA erweitert hierbei den bisher verbreiteten Industriestandard des „klassischen“ OPC um so wichtige Eigenschaften wie Plattformunabhängigkeit, Skalierbarkeit, hochgradige Verfügbarkeit und Internetfähigkeit und basiert nicht mehr länger auf Microsofts DCOM-Technologie. Somit ist man auch nicht mehr an die Microsoft-Plattform gebunden. Unterstützt wird diese Plattformunabhängigkeit durch das Bereitstellen von Implementierungen in .NET, Java und ANSIC/C++.

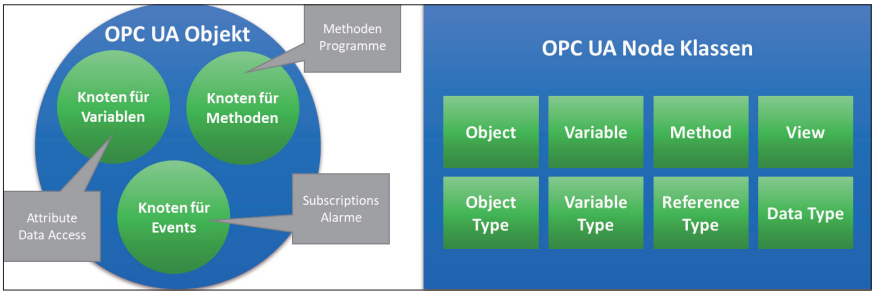
OPC UA setzt auf eine dienstorientierte Architektur und verwendet bewährte Sicherheitskonzepte, um unerlaubte Zugriffe oder Modifikationen von Daten zu verhindern. Gerade das Thema Sicherheit haben viele Firmen sträflich vernachlässigt, da die Maschinen bisher in der Regel nicht mit dem Internet verbunden waren. Aber spätestens seit Industrie 4.0 und der zunehmenden Vernetzung ist Sicherheit ein hochsensibles Thema.

Hier bietet OPC UA bereits einige Lösungen wie zum Beispiel Authentifizierung, das Signieren von Nachrichten oder das Verschlüsseln von Daten. Wie wichtig das Thema ist, zeigt sich daran, dass es schon in Abschnitt 2 (von 13) der OPC-UA-Spezifikation behandelt wird. Dieses Kapitel beschäftigt sich mit Sicherheitsstandards für Kommunikation wie SSL, TLS oder AES. Diese Sicherheitsmechanismen sind verpflichtend für die Hersteller, können aber frei kombiniert werden, angepasst an die Anwendungsfälle [1].

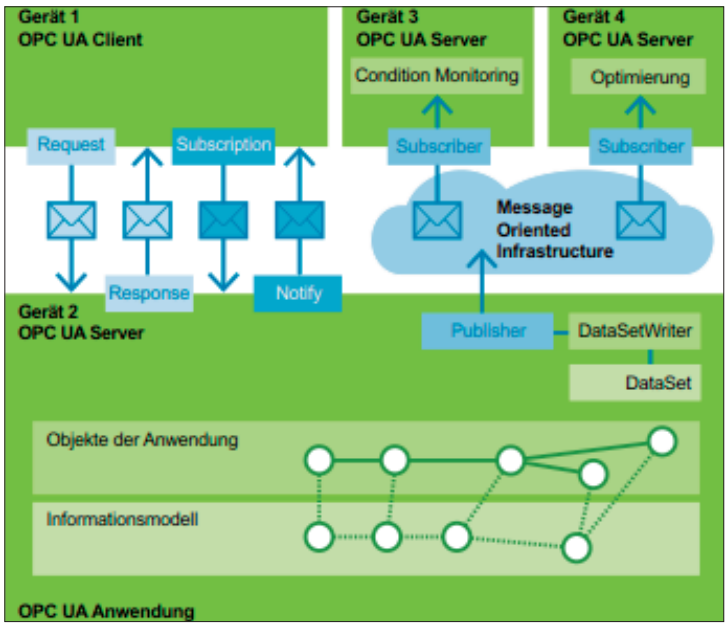
Wie gut hierbei die Sicherheit durchdacht wurde, zeigt das Bundesamt für Sicherheit in der Informationstechnik (BSI), das sich bereits mit dem Konzept OPC UA beschäftigt, es auf Sicherheitslücken überprüft und keine systematischen Fehler gefunden hat [2].

Neben den Kommunikationsstandards spielt das Informationsmodell eine wichtige Rolle. Denn was bringt eine sichere, standardisierte Kommunikation, wenn die Maschinen mit den Daten nichts anfangen können, weil sie diese nicht verstehen?

Um die Daten systematisch aufzubereiten und bereitzustellen, werden diese in einem Informationsmodell dargestellt. Dieses besteht aus Knoten und Beziehungen zwischen den



OPC-UA-Knoten und -Objekte (Bild 2)



Die Kommunikationsarten bei OPC Unified Architecture (Bild 3)

Knoten, abgebildet als beliebige Hierarchien, durch die navigiert werden kann. Die Knoten sind Variablen, Methoden und Ereignisse, die sich zu beliebig komplexen Objekten zusammenbauen lassen. Hierfür gibt es spezielle OPC-UA-Node-Klassen.

Aus diesen Typen können die Objekte gebildet werden (siehe Bild 2). Ein Beispiel für ein solches Objekt könnte ein Temperiergerät einer Spritzgussmaschine sein; es enthält Werte für Temperaturen, Events wie Alarme bei Temperaturüberschreitungen und semantische Informationen wie etwa Alarmgrenzen.

Die Typen der Knoten können standardisiert werden, wie etwa die Knoten einer Spritzgussmaschine. Diese allgemeine Definition nennt man „companion specification". Die Euromap 77 ist eine solche Spezifikation für Kunststoff- und Gummimaschinen und definiert alle allgemeingültigen Variablen, Methoden und Ereignisse. Dadurch können zwei verschiedene Hersteller unterschiedliche Maschinen produzieren und – da beide ihre OPC-UA-Schnittstellen gemäß E77 bereitstellen – trotzdem die gleiche Sprache sprechen. Somit kann ein OPC-UA-Client durch die Instanzen dieser E77-Objekte navigieren und diese Werte nutzen oder auch manipulieren. Der OPC-UA-Server vereinheitlicht die Adressräume verschiedener Maschinen, sodass die Clients nur eine Schnittstelle benötigen, um auf die verschiedenen Daten zuzugreifen.

Für die Kommunikation zwischen Server und Clients sieht OPC UA zwei Arten vor (siehe Bild 3): direkte Client-Server-Kommunikation oder ein Publish/Subscribe-Verfahren, wobei auch beide Arten parallel genutzt werden können. Ein Server kann sowohl Sender als auch Empfänger von Nachrichten sein. Bei der direkten Kommunikation müssen Server und Client sich kennen und tauschen direkt Nachrichten miteinander aus, wobei dem Sender der Empfang der Nachricht vom Empfänger bestätigt wird. Bei Publish/Subscribe kennt der Sender die Empfänger der Nachrichten nicht und weiß auch nicht, wie viele Empfänger sich für bestimmte Nachrichten angemeldet haben.

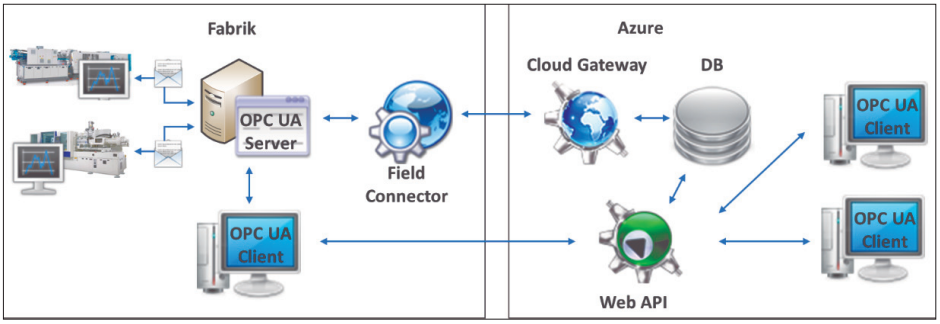
Um beim Beispiel der Spritzgussmaschine zu bleiben, könnten die Daten, die der Server bereitstellt, der Materialdurchsatz oder Energiedaten sein. Die Abonnenten könnten verschiedene Dienste oder Anwendungen sein, die diese Daten nutzen und auswerten.

Die Praxis

In Bild 4 ist die vereinfachte Architektur der Umsetzung eines Kundenszenarios mittels OPC UA zu sehen. Innerhalb eines Firmennetzwerks können Clients direkt über SOAP auf den OPC-UA-Server zugreifen und Daten abfragen sowie Daten zurückschicken. Soll auch von außerhalb des Firmennetzwerks der Zugriff auf die Daten möglich sein, um zum Beispiel freie Kapazitäten von Maschinen auf einem Marktplatz anzubieten, werden diese Daten über ein Field Gateway in die Cloud geschickt. Im Field Connector lassen sich die Daten filtern, bündeln oder bereits verarbeiten, bevor sie in die Cloud geschickt werden.

Eine Alternative, die Microsoft selbst mit einer eigenen Referenzimplementierung zu fördern versucht, ist der Weg direkt vom OPC-UA-Server in die Cloud. Innerhalb der Cloud werden die Daten gespeichert und über REST-Schnittstellen den verschiedenen Anwendungen zur Verfügung gestellt.

Zu den vorhandenen Referenzimplementierungen gehört auch eine von Microsoft [3]. Sie steht unter der MIT-Lizenz und kann daher auch zu gewerblichen Zwecken genutzt werden, solange Änderungen zurück ins offizielle Git-Repository fließen. Die Referenzimplementierung von Microsoft ist eine Open-Source-Lösung; es gibt aber auch kommerzielle Angebote, wie etwa von Unified Automation [4], die dann auch entsprechenden Support beinhalten.



Vereinfachte Darstellung einer möglichen OPC-UA-Architektur (Bild 4)

Zentrale Datenknoten

Die verschiedenen Maschinen selbst stellen bereits viele Daten zur Verfügung. Bisher ist es aber notwendig, diese Daten direkt ▶

an der Maschine am Bedienpult auszulesen. Das ist sehr unpraktisch und zeitintensiv – ganz abgesehen davon, dass es auf diese Weise nicht möglich ist, sich einen schnellen Überblick über den Zustand aller Maschinen zu verschaffen. Mit OPC UA lässt sich dieses Problem sehr gut lösen.

Generell sind zwei Situationen denkbar. Erstens: Der OPC-UA-Server dient als zentraler Datenknoten.

Szenario zwei (siehe Bild 5) sieht pro Maschine einen OPC-UA-Server vor. Dies wird bei vielen Maschinenherstellern der Fall sein, da diese oft noch keinen zentralen Server haben, der mit allen Maschinen verbunden ist und auf dem der OPC-UA-Server installiert werden könnte; diese Infrastruktur müsste zunächst eingerichtet werden. Was aber hingegen vorhanden ist, sind die Rechner an den Maschinen selbst. Auf ihnen laufen bereits Programme, welche die Daten der Maschine abfragen und sie dem Nutzer auf Bedien-Panels zur Verfügung stellen. Hier ist es relativ einfach, die Daten ebenfalls zu nutzen und über einen OPC-UA-Server für andere Nutzern, die nicht vor Ort sind, verfügbar zu machen.

Kommunikation zwischen Maschine und Server

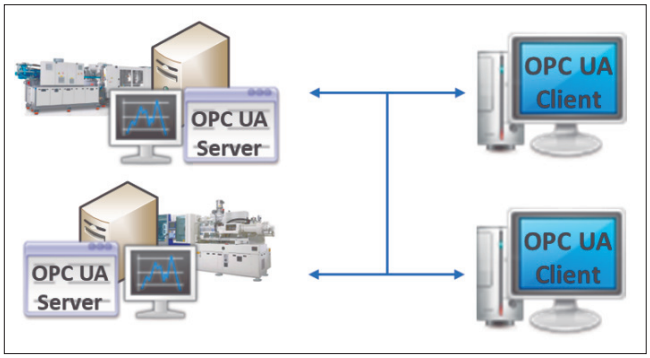
Nun stellt sich die Frage, wie der Server an die Daten kommt. Auch wenn dieser Weg von OPC UA nicht explizit vorgeschrieben wird und sich der OPC-UA-Server auf dem Leitrechner der Maschine befindet, sollte dieser typischen Standards entsprechen.

Listing 1: Variablen abfragen mit ZeroMQ-Nachricht

```
using ProtoBuf;
using ZeroMq.Middleware.Common.MappingModel;
using ZeroMq.Middleware.Common.MessageDefinitions.
RequestMessages.Interfaces;

namespace ZeroMq.Middleware.Common.
    MessageDefinitions.RequestMessages
{
    [ProtoContract(Name = nameof(
        GetVariableRequest))]
    public class GetVariableRequest :
        IVariableRequest
    {
        [ProtoMember(1, Name = nameof(
            RequestedVariableSetName),
            IsRequired = true)]
        public string RequestedVariableSetName {
            get; set; }

        [ProtoMember(2, Name = nameof(Property),
            IsRequired = false,
            OverwriteList = true)]
        public PropertyNode Property { get; set; }
    }
}
[<cTypeface:Roman>]
[<cFont:Candida Std>]
[<cTypeface:>] [<cFont:>]
```



Szenario zwei mit OPC-UA-Server pro Maschine (Bild 5)

Eine Möglichkeit ist Messaging; hierfür gibt es verschiedene Bibliotheken. ZeroMQ [5] etwa arbeitet asynchron, ist von der Programmiersprache unabhängig und kommt ohne Message Broker aus, was in diesen Einsatzszenarien von Vorteil ist, da die Endpunkte der Kommunikation bekannt sind und somit den Overhead eines Brokers einsparen.

Außerdem ist gewährleistet, dass sich der OPC-UA-Server später Daten verschiedener Maschinen, auch verschiedener Hersteller bedienen kann, wenn er als zentraler Datenknoten fungieren soll. Wichtig ist hierbei, dass die Maschinen ihre Daten per ZeroMQ übermitteln, egal ob es dann eine .NET-, eine Java- oder eine Implementierung in einer anderen Programmiersprache ist.

In Listing 1 ist eine Nachricht im ZeroMQ-Format zum Abfragen einer Variablen, etwa die Temperatur einer Spritzgussmaschine, zu sehen. Diese wird vom Server an die Maschine gesendet. Die Gegenstelle, die die Nachrichten empfängt und bearbeitet, wurde in die Software zur Bedienung der Maschine eingebaut, da diese bereits solche Werte von den Sensoren der Maschine abfragen kann.

Die Nachricht wurde generisch gehalten, um mit einer solchen Nachricht möglichst alle verschiedenen Variablen in dem Adressraum ermitteln zu können. Hier muss nur in der Software der Maschine sichergestellt werden, dass die in der Nachricht angefragte Variable auch auf die richtige Variable in der Maschine gemappt wird.

Adressraummodell

In diesem Praxisbeispiel wurde zunächst die E77-Spezifikation (Companion Spec) für Spritzgussmaschinen in den Server eingebunden. Diese steht bei Euromap, der Dachorganisation der europäischen Plastik- und Rubber-Machinery-Branche [6] als Modelldesign-Datei zur Verfügung und kann mittels eines sogenannten Modell-Compilers zum Beispiel in C#-Code umgewandelt werden.

In Listing 2 ist ein Auszug der Klasse für eine Spritzeinheit zu sehen. Insgesamt umfasst der generierte Code mehr als 60000 Zeilen. Für das Erzeugen der Klassen hat der Autor den Modell-Compiler von Microsoft für die Referenzimplementierung verwendet [7].

Listing 2: Auszug aus einer mit dem Modell-Compiler generierten Klasse

```
[System.CodeDom.Compiler.GeneratedCodeAttribute(
    "Opc.Ua.ModelCompiler", "1.0.0.0")]
    public partial class InjectionUnitsState :
    BaseObjectState
{
    #region Constructors
    /// <summary>
    /// Initializes the type with its default
    /// attribute values.
    /// </summary>
    public InjectionUnitsState(NodeState parent) :
        base(parent)
    {
    }

    /// <summary>
    /// Populates a list with the children that
    /// belong to the node.
    /// </summary>
    /// <param name="context">The context for the
    /// system being accessed.</param>
    /// <param name="children">The list of children to
    //// populate.</param>
    public override void GetChildren(
        ISystemContext context,
        IList<BaseInstanceState> children)
    {
        if (m_nodeVersion != null)
        {
            children.Add(m_nodeVersion);
        }

        base.GetChildren(context, children);
    }
}
```

Nachdem die Klassen für den Adressraum zur Verfügung standen, mussten als Nächstes die realen Daten über eine Middleware angebunden werden, im vorliegenden Fall über Messaging. Dafür war ein Mapping zwischen den generierten Klassen und den geschriebenen Messages notwendig. Damit kann der Adressraum mit validen Daten gefüttert werden. Zur Laufzeit wird via *NodeManager* der Adressraum erzeugt und die Daten können über die Nachrichten von der Maschine abgefragt werden

Anbindung an die Cloud

Bleibt noch die Anbindung an die Cloud. Gebraucht wird ein Field Gateway, das die Daten des OPC-UA-Servers aufbereitet und in die Cloud bringt. Dieser Bereich ist nicht mehr Teil der OPC-UA-Spezifikation und wird daher auch nur kurz betrachtet.

Das Field Gateway ist im Grunde nichts anderes als ein OPC-UA-Client, der die Daten des OPC-UA-Servers nutzt. Einerseits könnten einfach bestimmte Daten genau wie bei lokalen Clients nur abgefragt und an die Cloud eins zu eins weitergereicht werden, andererseits ist aber auch denkbar, dass in dem Gateway auch bereits Logik integriert wird, welche die Daten bündelt oder Durchschnittswerte liefert.

Wie der Fall sich dann innerhalb der Cloud darstellt, ist sehr vielseitig. Von Überwachungsprogrammen der Maschinen bis hin zu Marktplätzen, auf denen freie Kapazitäten angeboten werden, ist vieles denkbar. Im vorliegenden Fall werden Ereignisse an einen Event-Hub in Microsoft Azure weitergereicht, um zum Beispiel Alarme an andere Arten von Clients weiterzugeben. Dafür müssen die vom OPC-UA-Server über REST-Schnittstellen abgefragten Daten (JSON) in das Nachrichtenformat des Event-Hub überführt werden (Klasse *EventData*).

Fazit

OPC UA ist der Industrie-4.0-Standard für die sichere und plattformunabhängige Kommunikation zwischen Maschinen und Anwendungen. Er liefert sichere Konzepte und ermöglicht eine M2M-Kommunikation oder ein Bereitstellen von Maschinendaten für andere Anwendungen, ohne das Rad neu erfinden zu müssen. Maschinenhersteller, die das digitale Zeitalter nicht verpassen wollen, werden nicht umhinkommen, sich mit diesem Standard zu beschäftigen. ■

[1] OPC Unified Architecture, Unified Architecture, www.dotnetpro.de/SL1805OPCUA1

[2] BSI, Sicherheitsanalyse Open Platform Communications Unified Architecture (OPC UA), www.dotnetpro.de/SL1805OPCUA2

[3] OPC UA .Net Standard Stack and Samples, www.dotnetpro.de/SL1805OPCUA3

[4] Unified Automation, www.unified-automation.com

[5] ZeroMQ, http://zeromq.org

[6] Euromap 77, www.dotnetpro.de/SL1805OPCUA4

[7] OPC Foundation Model Compiler, www.dotnetpro.de/SL1805OPCUA5

Martin Kleine
arbeitet als Senior Consultant und Projektleiter bei AIT. Er ist seit 2003 als Softwareentwickler und Architekt im industriellen Umfeld tätig. Sein Schwerpunkt liegt auf ALM-Prozessen und dem Entwickelt von Windows- und mobilen Anwendungen mit Microsoft-Technologien.

dnpCode A1805OPCUA