OPC UA

Information Modeling

Alexander Heinisch

0627820, 535

alexander.heinisch@gmx.at

Inhaltsverzeichnis

Allgemeines über OPC UA

Die grundlegenden Teile der OPC Unified Architecture (UA) sind Übertragungsmechanismen und Datenmodellierung. In der ursprünglichen Form der UA wurde spezieller Wert auf die performante Übertragung von Binärdaten via TCP, sowie die Übertragung über etablierte Internet Standards wie XML, HTTP oder WebServices gelegt, da diese eine Firewall freundliche Übertragung ermöglichen.

Außerdem wurde eine abstrakt gehaltene Service-Schicht in UA eingeführt welche den Servern und Clients als Interface zum Information Model dienen, um den Zugriff auf einzelne Daten ohne das Wissen über das gesamte Model voraussetzen zu müssen. Wobei der Server als Daten publizierender und der Client als Daten konsumierender Teil zu verstehen sind. Eine Applikation muss jedoch nicht strikt ein Server oder ein Client sein, es können hier auch hybride Varianten implementiert werden.

OPC UA Spezifikationen

OPC UA ist auch unter dem Standard IEC 62541 bekannt. Diese Standardisierung setzt voraus, dass die Architektur in verschiedene Spezifikationsteile unterteilt ist.

Die in Abbildung 1 dargestellten Parts sind unterteilt in die Core Specification Parts und die Access Type Specification Parts wobei erstere dazu dienen OPC UA und letztere das Information Model zu

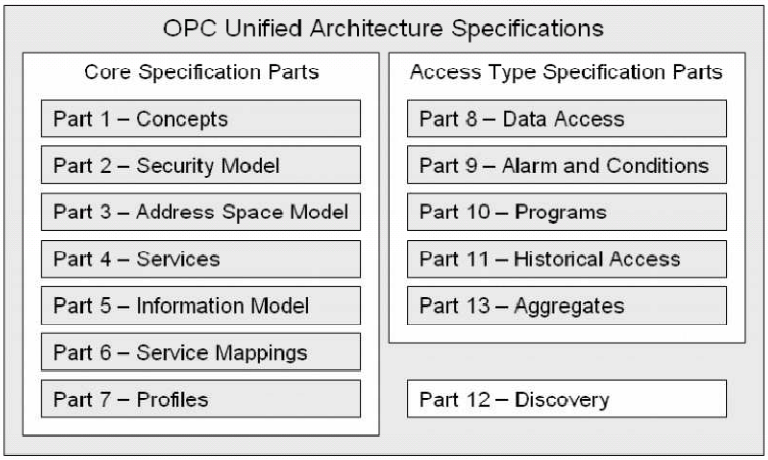

Abbildung 1

spezifizieren.

Als kleine Anmerkung zu dem in Allgemeines über OPC UA erwähnten Services sei noch gesagt das die in Services (Part4) spezifizieren, abstrakt die Daten die vom Server angeboten werden damit diese vom Client gefunden werden können, es beschreibt jedoch nicht wie die Daten übertragen werden und wie die Daten in der API die von der Applikation verwendet wird gehalten werden. Die Übertragungsart und das Mapping der Daten wird in Service Mappings (Part 6) spezifiziert.

- Information Model (Part 5) definiert die Basis für alle Information Models welche OPC UA verwenden. Es definiert unter anderem:
- Den Einstiegspunkt in den Address Space der von den Clients benutzt wird um durch die Instanzen und Typen von OPC UA Servern zu navigieren.
- Die Basistypen welche die Grundlage für verschiedene Typhierarchien darstellen.
- Die vorhandenen und erweiterbaren Objekt- und Datentypen.
- Das Server Objekt welches die Fähigkeiten dieses und Diagnoseinformationen über diesen deklariert.

Der Access Type Specification Part Data Access definiert wie Automatisierungsdaten dargestellt werden, und welche Eigenschaften einzelne Knoten haben. In Alarm and Conditions wird festgelegt wie gewisse Bedingungen und Alarmzustände überwacht werden und welche Events und Zustandsabfolgen auftreten können. In Programs wird festgelegt welche Zustände im Programmablauf auftreten können und wie in diesen eingegriffen, sowie überwacht werden kann. History Information spezifiziert wie Informationen über die Konfiguration von Daten und die Event History aufbereitet wird. Aggregates dient dazu zu definieren wie Verknüpfte Informationen weiter verarbeitet werden und wie aktuelle Werte überwacht werden können. Discovery beschreibt wie Server im Netzwerk gefunden werden können und wie ein Client eine Verbindung zu diesem herstellen kann.

Warum OPC UA

Die bisherige OPC Spezifikation beruht Großteils auf den von Microsoft entwickelten und geförderten OLE und COM – basierten Modellen, welche zu stark plattformabhängig (Microsoft Windows) und daher für Bereiche aus dem Embedded Systems- und Cross Platform-Bereich ungeeignet sind.

Weiters werden einheitliche Schnittstellen für .Net, Java und andere moderne Programmiersprachen bereitgestellt, welche die Entwicklung solcher Anwendungen erleichtert.

Information Modeling

Wozu benötigt man Information Modeling

Im Classic OPC konnten nur „rohe“ Daten zur Verfügung gestellt werden, zum Beispiel den Messwert eines Sensors. Dieser wurde über dessen Namen und dessen Gerätekennung angefordert.

In Information Model werden generische Gerätetypen zur Verfügung gestellt, welche von den jeweiligen Herstellern erweitert werden können. Der Client jedoch kann auf jedes Gerät welches vom selben Basismodell abgeleitet wurde über das Basismodel auf die Werte zugreift oder jenes des

Herstellers verwendet, da dieses möglicherweise mehr Informationen und Kontrollmöglichkeiten anbietet. Dies bietet mehr Spielraum wenn man auf Faktoren wie Composability denkt da es möglich ist Geräte von verschiedensten Herstellern über einheitliche Schnittstellen anzusprechen.

Die grundlegenden Prinzipien des Information Modeling in OPC UA sind so aufgebaut das dem Systemdesigner quasi keine Grenzen gesetzt sind. Es wurden alle Typen objektorientiert aufgezogen, was den Mechanismus der Vererbung und somit der Typhierarchie mit sich bringt. Dadurch ist es wie bereits kurz beschrieben möglich, das ein Client mit den Basistypen arbeitet, und herstellerspezifische Typen nicht beachtet. Es können außerdem verschiedenste Referenzen zwischen Daten und zwischen Daten und Funktionen definiert werden, wodurch es möglich ist die gleichen Daten je nach Anwendungsfall verschieden aufzubereiten.
Dadurch das Information Modeling immer am Server gemacht wird, benötigt der Client keine Informationen zu diesem, was das Design dieses vereinfacht und den Einsatzbereich vergrößert.

Jeder Client hat dadurch die Möglichkeit nach seinen Bedürfnissen über einheitliche Schnittstellen auf die Daten die er benötigt zuzugreifen und über ebenfalls diese Schnittstellen das Gerät zu konfigurieren. Es obliegt dabei nur dem Client, bzw. dessen Anwendungsbereich ob er spezialisierte oder allgemeine und vor allem welche Typen er für den Zugriff verwendet.

Information Modeling

Nodes und Referenzen

Die Hauptelemente beim Information Model sind Nodes und Referenzen zwischen diesen. Nodes können von verschiedenen NodeClasses sein und stellen Interfaces oder Typen dar. Je nach

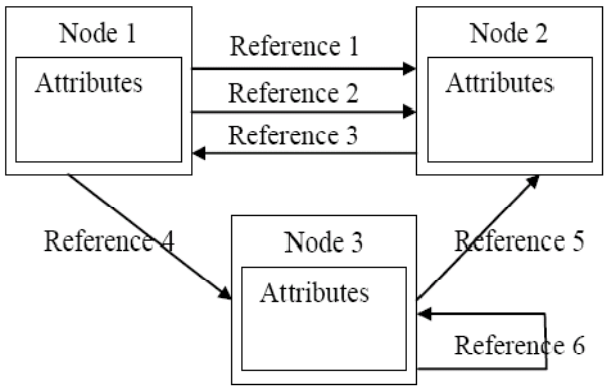

Abbildung 2

NodeClass haben die Nodes verschiedene Attribute.

Verschiedene Attribute sind zum Beispiel: NodeId (Identifiziert den Node eindeutig im Server), NodeClass (gibt an ob es sich um ein Objekt oder eine Methode handelt), BrowseName (der Name mit dem der Knoten gesucht werden kann), DisplayName (der Name des Knotens den man an einem Frontend ausgeben kann), Description, WriteMask (gibt an welche Attribute nur gelesen und welche auch verändert werden dürfen), UserWriteMask (gibt an welche Benutzer die zur Zeit mit dem Server verbunden sind Modifikationen durchführen dürfen).

Ebenso wie bei den Nodes gibt es auch für die Referenzen verschiedene Typen, die Reference Types, welche wie Nodes im Address Space sichtbar sind, um es den Clients zu ermöglichen Informationen über die Referenzen zu sammeln.

Die Reference Types beinhalten neben, den bereits von den Nodes bekannten Attributen, die Attribute: IsAbstract (gibt an ob der Referenztyp nur zu organisatorischen Zwecken dient oder ob er auch tatsächlich als Referenz benutzt werden kann), Symmetric (gibt an ob es sich um eine symmetrische Referenz handelt) und InverseName.

Object, Variables and Methods

NodeClass Variablen stellen einen Wert dar der von den Clients gelesen werden kann und auf dessen wechsel Clients hören können, wobei der Typ des Wertes von der Variable abhängt.

NodeClass Methoden stellen Methoden aufrufe dar, die vom Client über CallServices gerufen werden können und spezifizierte Eingabewerte verlangen und anhand der spezifizierten Ausgabe antworten. Einzige Anforderung an diese Methoden ist eine recht kurze Ausführungszeit um die Antwort nicht zu verzögern. Für länger dauernde Prozesse wird nahegelegt Programme zu verwenden.

NodeClass Objekte sind Typen ohne Wert, welche Variablen und Methoden beinhalten, und dienen als Strukturierungselemente im AddressSpace. Weiters können Objekte als EventNotifier verwendet werden auf die die Clients horchen um Events zu empfangen.

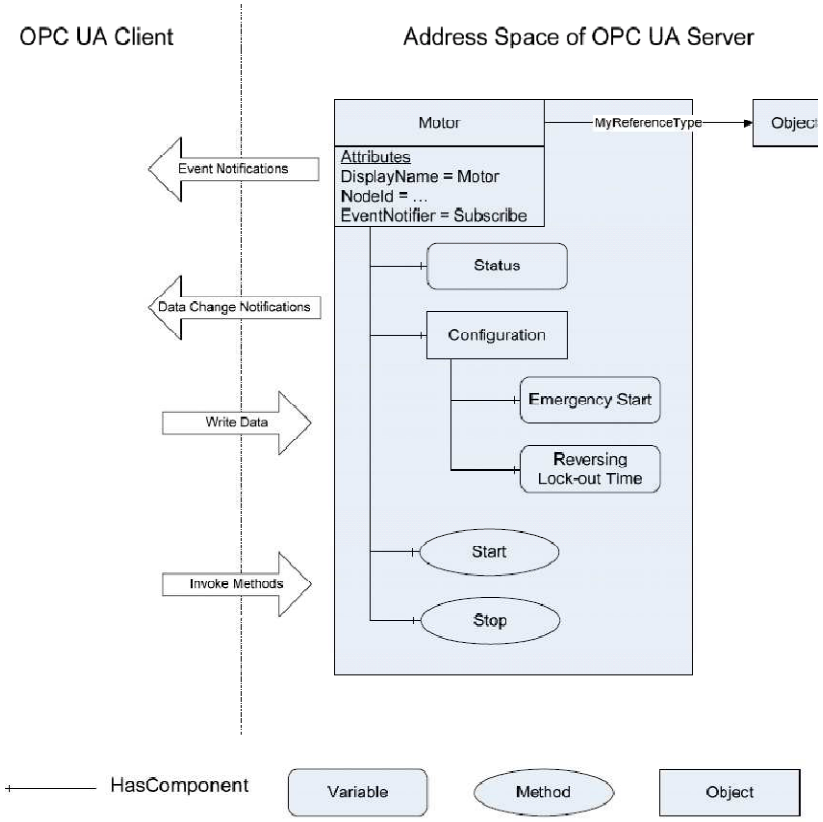

Abbildung 3

Ein zusätzliches Attribut des ObjektTypes ist das EventNotifier Byte welches angibt ob das Objekt auf Events subscriben kann, ob auf diese zugegriffen werden kann und ob diese geändert werden können.

Variablen sind da diese Werte halten um mehrere Attribute wie Value, DataType, ValueRank, ArrayDimensions, MinimumSamplingInterval, Duration (wie schnell Änderungen wahrgenommen werden), Historizing (wird eine History / ein Verlauf für diesen Wert geführt).

Methoden haben zusätzlich die Attribute Executable und UserExecutable sowie die Eigenschaften InputArguments und OutputArguments.

Typen für Objekte und Variablen

Ein großer Vorteil von OPC UA ist das Typen von Werten sich nicht nur auf den Datentyp beschränken, sondern auch eine Beschreibung des Gerätes, welches den Wert zur Verfügung stellt, ermöglicht. Dies Objekttypen können dann durch Ableitungen weiter Herstellerspezifisch ausgebaut werden. Im Gegensatz zu Objekten und Variablen ist dies bei Methoden nicht möglich, da diese über deren BrowserName und die Ein- und Ausgabeparameter bestimmt sind.

Anzumerken ist jedoch das jeder Typ außer den Basistypen von einem anderen Typ abgeleitet sein muss. Das heißt das jeder ObjektTyp von BaseObjektType und jeder VariableType von BaseVariableType abgeleitet und somit eine Spezialisierung dieses ist.

Einfache Objekttypen

Objekttypen können simpel oder komplex sein, wobei komplexe Typen eine Struktur von Nodes bereitstellen die in jeder Instanz sichtbar sind und simple Typen nur semantisch Eigenschaften für das Objekt definieren. Über das ObjektTypAttribute IsAbstract kann man bestimmen ob dieses abstrakt ist oder nicht. Im Falle eines abstrakten Objekttyps kann dieses nicht direkt Instanziert werden und dient sozusagen nur der Strukturierung der Objekttypenhierarchie.

Einfache Variablentypen

Wie bei den Objekttypen gibt es auch bei den Variablentypen simple und komplexe Typen, wobei die simplen semantische Angaben und Einschränkungen der Instanz beinhalten und der komplexe Typ eine Struktur von Nodes der Instanz bereitstellt.

Variablentypen haben die Attribute Value, ValueRank, ArrayDimensions, IsAbstract und VariableType. Wird mit IsAbstract definiert das es sich um einen abstrakten Variablentypen handelt so muss ein Client, der auf eine Instanz dieses Typs zugreifen will, sicherstellen dass er mit allen möglichen abgeleiteten Typen dieses Umgehen kann.

Instanzierung

Eine InstanzDeklaration ist eine benannte Einheit um komplexe Objekttypen zu definieren. Eine Besonderheit von InstanzDeklarationen ist das diese über ihren BrowsePath eindeutig identifizierbar sind. Dies ist notwendig, da InstanzDeklarationen meist in einem anderen Node leben als die Instanzen selbst, welche über die NodeId identifizierbar sind. Ein komplexer Objekttyp mit einer InstanzDeklaration ist vergleichbar mit einer Klasse in Java (siehe Abbildung 4), außer dass man in OPC UA nichts über die Implementierung von Methoden an sich aussagen kann. Jedoch hat man die

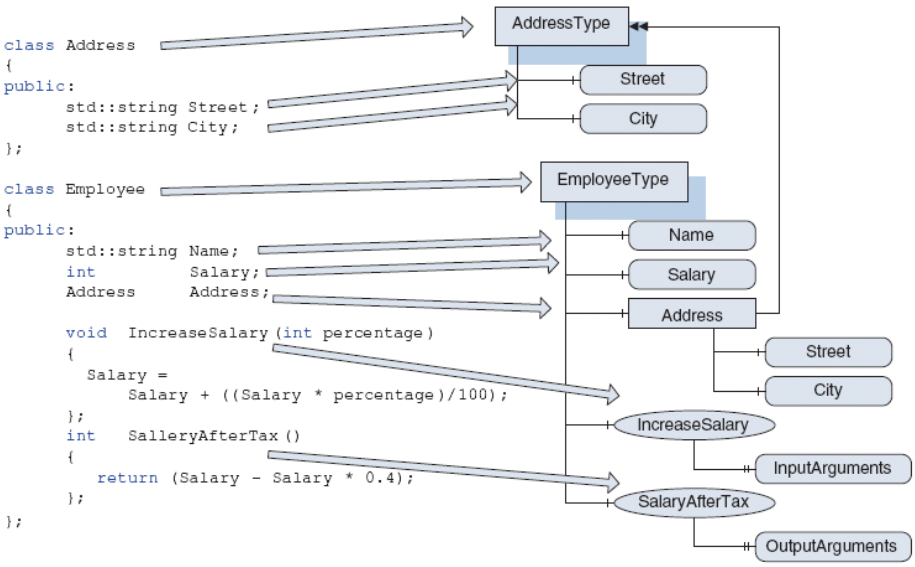

Abbildung 4

Möglichkeit ohne Änderung des Objekttyps, Komponenten zur Instanz hinzuzufügen.

Modelling Rules

Jede Instanz welche von einer Typdefinition referenziert wird, wird zu einer Instanz Deklaration wenn es eine Modelling Rule dafür gibt. Eine Modelling Rule beschreibt was mit einer Instanz Deklaration in Hinsicht auf die Instanzen des Objekttyps erlaubt ist.

Für die Modelling Rules gibt es 3 Richtlinien:

- Mandatory: Jede Instanz muss ein Gegenstück zur Instanz Deklaration mit demselben BrowserPath und demselben Typ der Instanz Deklaration, sei es Objekt-, Variablen- oder Subtyp haben.
- Optional: Jede Instanz kann so ein Gegenstück haben, muss sie aber nicht.
- Constraint: Die Instanz Deklaration legt Einschränkungen auf Instanzen der TypDefinition.

Anzumerken ist hierbei, dass solange nicht durch auferlegen einer Einschränkung bei der TypDefinition das hinzufügen von Referenzen zu Instanzen der Typdefinition erlaubt ist.

Datenvariablen und Eigenschaften

Es wird in OPC UA zwischen Datenvariablen und Eigenschaften unterschieden.

Datenvariablen werden dazu verwendet die Werte eines Objektes wie zum Beispiel die Temperatur eines Temperatursensors zu speichern. Diese können simpel oder komplex sein, das heißt sie können also auch nur Teile der von einem Messgerät aufgezeichneten Daten enthalten.

Eigenschaften geben die Charakteristik eines Nodes wieder, das ist zum Beispiel welcher Sensortyp verwendet wurde, bzw. in welcher Einheit der Datenwert vorliegt.

Datentypen

In OPC UA wird zwischen folgenden Datentypen unterschieden:

- Built-In DataTypes: Sind fix von der OPC UA Spezifikation vorgegebene ausdefinierte Typen die nicht erweitert werden können. Beispiele sind: Int32, Boolean und Double
- Simple DataTypes: sind untertypen der Built-In DataTypes. Ein Beispiel wäre Duration was einen Untertyp von Double darstellt, welcher nur Werte im Millisekunden Bereich annehmen darf (siehe dazu auch Abbildung 5).
- Enumeration DataTypes: Stellen eine endliche Meng vergebener Namen dar, aus denen einer gewählt werden kann. Bei der Übertragung wird dieser wie Int32 behandelt. Ein Beispiel dafür wäre die NodeClass.

- Structured DataTypes: Stellen benutzerdefinierte, komplexe Datentypen dar. Ein Beispiel wäre der Argument Datentyp welcher ein Argument einer Methode definiert und somit den Namen, den Datentyp und die Beschreibung dieses festlegt.

Weiters gibt es auch noch andere, abstrakte, Datentypen welche in keine der genannten Kategorien einteilbar sind und nur zur Organisation der Datentyphierarchie dienen.

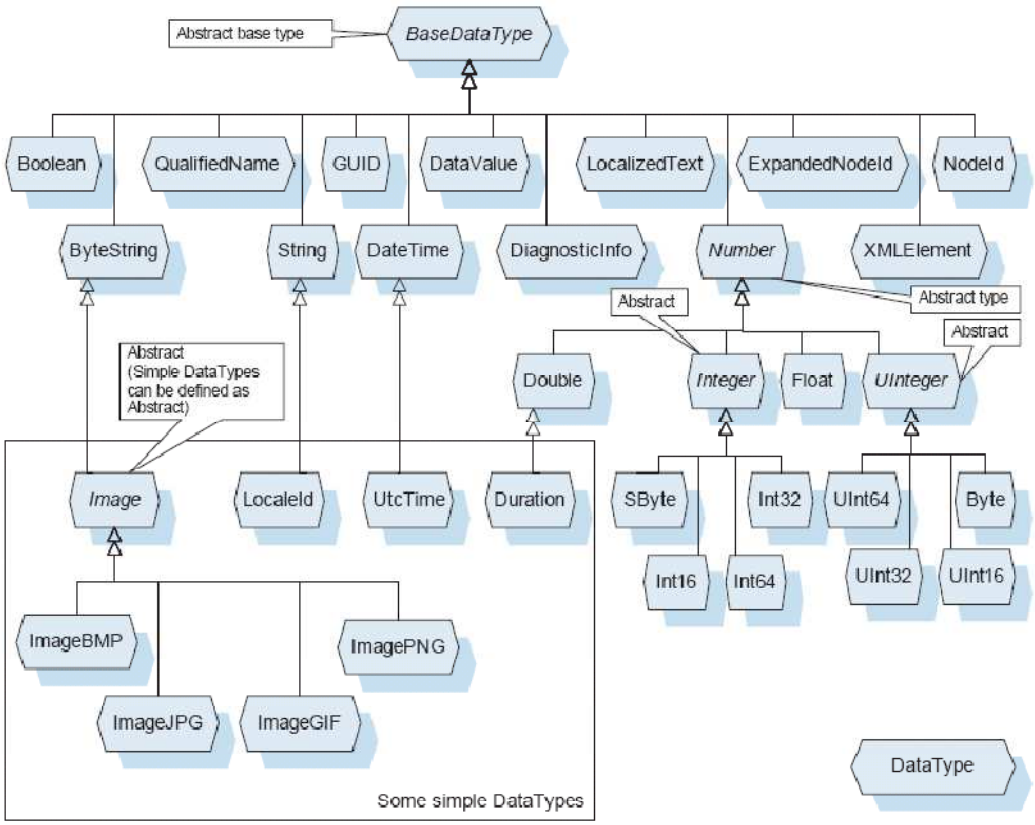

Abbildung 5

View

Um einem Client die Sicht auf den Server zu bieten, die er auch benötigt ist es möglich am Server verschiedene Ansichten zu definieren welche die Anzahl der sichtbaren Nodes und Referenzen reduzieren. Views können serverseitig kombiniert werden, jedoch hat der Client keinen Einfluss darauf.

Jeder Client greift dann nur noch auf die View zu die für ihn am besten passt, so ist es möglich das Clients aus der MES Ebene eine komplett andere Sicht auf die Daten angeboten bekommen als, ein Terminal welches zur Überwachung eines Maschinenstatus dient.

Events

Events werden über den Objekttyp EventType referenziert. Hier kann wiederum von den Clients festgelegt werden, welche Events sie bekommen wollen, und die anderen ausgeblendet werden.

Quellen

- http://www.commsvr.com/UAModelDesigner/Index.aspx

  Online Buch über OPC UA.
- OPC Unified Architecture (ISBN 978-3-540-68898-3)

  Überblick über Classic OPC, OPC UA und OPC UA Information Model

  Buch von Wolfgang Mahnke, Stefan-Helmut Leitner und Matthias Damm
- http://blog.matrikonopc.com/index.php/thoughts-on-the-opc-ua-information-model/

  Thougths on OPC UA Information Model by Eric Murphy.