183.168 Dezentrale Automation VO

Information Modeling in OPC UA

Dominik Bunyai, e0625305
dominik.bunyai@student.tuwien.ac.at

Inhaltsverzeichnis

1 Einführung

Das OPC [1] UA [2] Informations- Modell beschreibt standardisierte Knoten im Adressbereich eines Servers. Diese Knoten [3], sowie alle zur Diagnose verwendeten Daten haben einen standardisierten Datentyp. [2]

2 Grundlegende Konzepte

In OPC UA gibt es einige wenige grundlegende Konzepte, welche bei der Modellierung von Informationen angewandt werden:

- Verwendung objektorientierter Techniken wie Hierarchien und Vererbung
- Typinformationen können in der selben Weise abgefragt werden wie die Instanzen selbst
- Eine vollständige Vernetzung der Knoten ermöglicht die Weitergabe der Information in vielen Richtungen
- Erweiterbarkeit der Typhierarchien sowie der Art der Referenzen zwischen den Nodes
- Keine Limitierung bei der Informationsmodellierung um ein passendes Modell für die zu Verfügung stehenden Daten zu ermöglichen
- OPC UA Informations- Modellierung geschieht stets serverseitig [1]

2.1 Knoten

Grundlegende Modellierungstypen sind Knoten und Referenzen zwischen Knoten.

Jeder Knoten kann in eine Knotenklasse eingeteilt werden. Knoten können wiederum Instanzen sowie Typen darstellen. Die Eigenschaften der Knoten werden durch sogenannte Attribute beschrieben, wovon abhängig von der Knotenklasse eine unterschiedliche Anzahl an Attributen zur Verfügung steht. [1]

Zur eindeutigen Identifizierung eines Knotens innerhalb eines OPC Servers besitzt jeder Knoten das Attribut *NodeID*. Bei jeder Anfrage an den Server

[1] OLE (Object Linking and Embedding) for Process Control
[2] Unified Architecture
[3] sog. Nodes

wird diese eindeutige Identifizierung verwendet. Auch bei der Abfrage der vorhandenen Knoten werden die *NodeIDs* als Ergebnis geliefert.

Weitere Attribute, durch welche jeder Knoten beschrieben wird:

NodeClass Gibt die Klasse des Knoten an. Dadurch werden weitere vorhandene Attribute erklärt.

BrowseName Dieses Attribute wird beim Durchsuchen und Programmieren eines OPC Servers verwendet. Der *BrowseName* besteht aus einem Namespace und einer nicht lokalisierten Zeichenkette.

DisplayName & Description Diese zwei Attribute sind sprachenabhängig und geben weitere Informationen über den Knoten.

WriteMask & UserWriteMask Optionale Argumente die die Schreibrechte definieren. [1]

2.2 Referenzen

Referenzen sind Zeiger auf einen Knoten im selben oder in einem anderen OPC Server, durch Speicherung seine *NodeID*. Ausserdem wird die Semantik der Referenz, sowie die Richtung gespeichert. Referenzen werden nicht aktualisiert, auch wenn der Zielknoten nicht mehr vorhanden ist. Innerhalb eines Knotens sind die gespeicherten Referenzen unsortiert, das heißt ein zweimaliges Abfragen seiner Referenzen kann zu unterschiedlichen Ergebnissen führen.

Um die Semantik einer Referenz darzustellen, werden sogenannte Referenztypen verwendet. Diese wurden in der Spezifikation von OPC UA festgelegt (die Definition weiterer Referenztypen ist allerdings möglich). [1]

2.3 Objekte, Variablen und Methoden

Die wichtigsten Knotenklassen sind Objekte, Variablen und Methoden, welche in den selben Beziehungen zueinander stehen, wie das bereits aus der objektorientierten Programmierung bekannt ist. Objekte besitzen Variablen und Methoden und können "Events feuern" [4].

Knoten der Klasse Variable stehen für einen Wert, welcher von den OPC Clients gelesen und verändert werden kann. OPC Clients können sich auch zur Benachrichtigung bei Änderung des Wertes registrieren.

[4] engl.: to fire Events

Knoten der Methoden- Knotenklasse stellen eine von OPC Clients aufrufbare Methode dar. Jeder Methode spezifiziert seine Eingangs- Parameter sowie den zu erwartenden Ausgangswert. Allerdings erfüllt eine Methode nur eine kurze Aufgabe und liefert das Ergebnis sofort zurück. Für längere Aufgaben (stellen eines Ventils) existieren sogenannte Programme.

In der Praxis sieht die Definition einer Methode in OPC wie in Abbildung 1 aus:

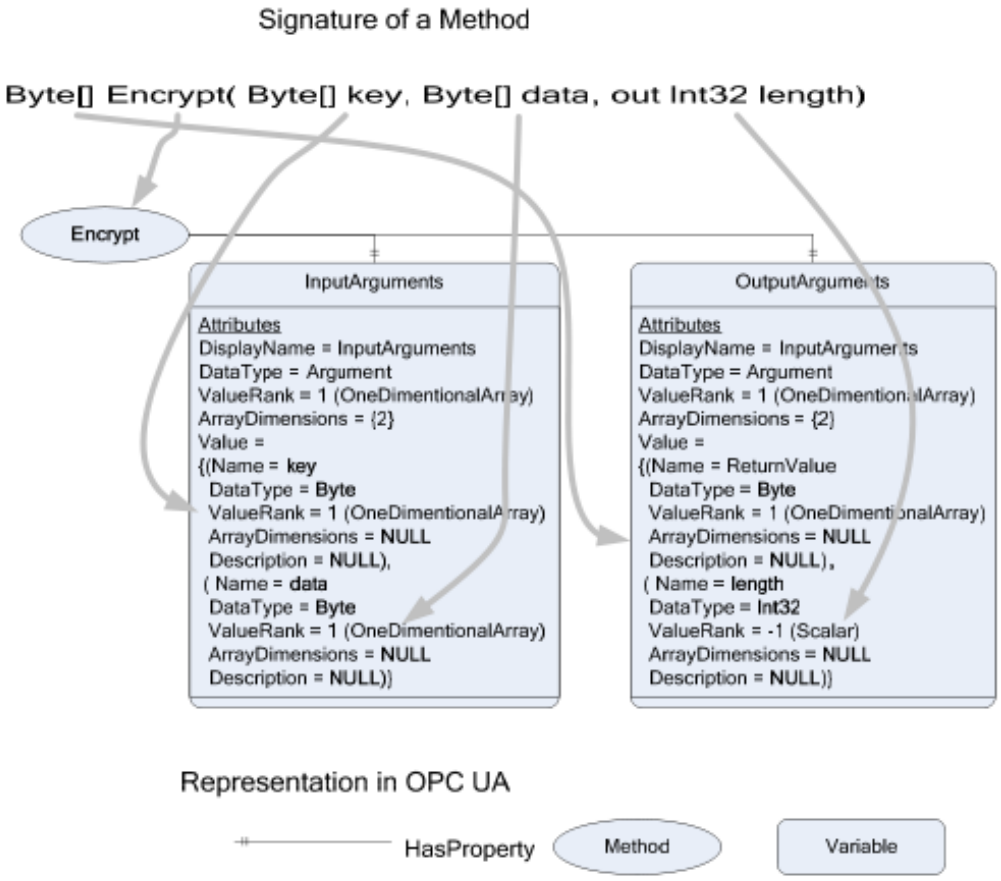

Abbildung 1: Definition einer Methode in Pseudo Code für OPC UA [1]

Die Knotenklasse Objekt dient dem Zusammenfassen von Methoden, Variablen sowie Objekten in Speichereinheiten. Falls ein Objekt einen Wert besitzt, befindet sich dieser als Attribut der Klasse Variable in Adressbereich.

2.4 Objekt- und Variablentypen

Objekte können in vorgegebene Objekttypen eingeteilt werden. So kann einem Objekt der Typ Temperatursensor zugewiesen werden. Daraus wissen alle Clients, dass dieses Objekt ein Temperatursensor ist. Natürlich können Hersteller darauf aufbauend noch weitere Eigenschaften ihrer Sensor implementieren. Doch der Basistyp bleibt ein Temperatursensor.

Es gibt einfache und komplexe Objekttypen. Komplexe Objekttypen besitzen eine Struktur unterhalb der ersten Informationsebene. Einfache Objekttypen definieren lediglich die Semantik des Objekttyps.

Genauso wie bei den Objekttypen, gibt es auch einfache und komplexe Variablentypen. Die komplexen Variablentypen definieren die Struktur unter ihnen, die einfachen Variablentypen geben nur Informationen über die Semantik des Variablentyps.

2.4.1 Einfache Objekttypen

Die einzige Erweiterung eines einfachen Objekttyps besteht in der Information der Abstraktheit. Ist ein Objekttyp abstrakt kann dieser nicht als Typ eines Objekts referenziert werden. Abstrakte Objekttypen dienen lediglich der Strukturierung der Typhierarchie.

2.4.2 Komplexe Objekttypen

Komplexe Objekttypen besitzen eine definierte Struktur unterhalb der ersten Informationsebene. Bei der Verwendung von Komplexen Objekttypen wird sicher gestellt, dass bei Instantiierung eines Objekts dieses Objekttyps das neu entstandene Objekt die selbe Struktur aufweist, wie der Objekttyp.

2.4.3 Einfache Variablentypen

Einfache Variablentypen geben Informationen über die Semantik des Variablentyps. Des weiteren kann ein einfacher Variablentyp auch die Verwendung des Werteattributs einer Variable einschränken. Alle Variablentypen bauen auf dem Grundtyp *BaseDataVariableType* auf, der von OPC UA vorgegeben wird. Subtypen des *BaseDataVariableType* können dann Beschränkungen in den Zugriffsberechtigungen vornehmen.

2.4.4 Komplexe Variablentypen

Komplexe Variablentypen sind den komplexen Objekttypen sehr ähnlich. Der größte Unterschied liegt in der Instantiierung (siehe 2.4.5). Während komplexe Objekttypen auch Methoden und Objekte instantiieren können, sind komplexe Variablentypen nur zur Instantiierung von Variablen im Stande.

2.4.5 Instanzendeklaration

Instanzen von Objekttypen, inklusive der gesamten Objekthierarchie, werden ohne Werte erzeugt. Um die Instanzen an einen Objekttypen zu binden werden hierarchische Referenzen erzeugt, welche auf den Objekttyp verweisen. Instanzendeklarationen können über ihren Objekttypen eindeutig identifiziert werden. Ein komplexer Objekttyp mit Instanzen von Methoden und Variablen ist einer objektorientierten Klasse mit Variablen und Methoden sehr ähnlich, da die Instanzen, wie in einer Klasse durch den Namen (*BrowseName*) eindeutig identifiziert werden. Der größte Unterschied zu objektorientierten Klassen liegt in der Erweiterbarkeit. Anstatt die Klasse abzuleiten um eine Variable hinzuzufügen, kann diese direkt zum definierten Objekttypen erweitert werden.

2.4.6 Modellierungsregeln

Bei jeder Referenzierung einer Instanz durch einen Objekttyp oder Variablentyp, gibt es drei Möglichkeiten eine Modellierungsregel hierfür zu erstellen:

1. Notwendig: Jede Instanz muss ein Gegenstück mit dem selben *BrowseName* besitzen. Das kann ein Objekt oder eine Variable sein, muss aber vom selben Typ oder von einem Subtyp der Instanz sein.
2. Optional: Jede Instanz kann ein Gegenstück, wie bei Punkt 1 beschrieben, besitzen, muss aber keines haben.
3. Richtungsweisend: Durch die Instanz kann eine weitere Verhaltensweise vorgegeben werden. Zum Beispiel muss beim Objekttyp Array jedes Element einen Subtyp angehören.

3 Weiterführende Konzepte

3.1 Datenvariablen und Eigenschaften

In OPC UA werden zwei Variablentypen definiert:

- Datenvariablen und
- Eigenschaften

Datenvariablen werden zur Darstellung von Daten verwendet (wie Temperatur oder Durchflussgeschwindigkeit). Diese können komplex sein – das heißt sie beinhalten Subtypen, welche die Daten sowie Eigenschaften enthalten, welche diese beschreiben.

Eigenschaften werden verwendet, um Informationen über einen Variablentyp weiterzugeben, welche nicht in den Attributen enthalten sind. Dies könnte zum Beispiel die Einheit der gemessenen Temperatur sein. Eigenschaften sind immer einfache Variablen, da diese Informationen nicht mehr in Subtypen unterteilt werden können.

Um zwischen Datenvariablen und Eigenschaften unterscheiden zu können wurden einige Hilfsregeln entworfen. Eine davon besagt, dass Datenvariablen ihre Werte sehr oft und schnell verändern können, und Eigenschaften ihre Werte über einen langen Zeitraum beibehalten, bzw. diese niemals verändern. Datenvariablen können von Servern verändert werden. Eigenschaften liegen oft in Konfigurations- Datenbanken.
Jeder Knoten kann Eigenschaften über eine *HasProperty* Referenz besitzen und alle Eigenschaften gehen auf den *PropertyType* zurück. Datenvariablen gehören zu einem Objekt oder einem Objekttypen und werden durch eine *HasComponent* Referenz durch das Objekt referenziert.

3.2 Eltern- Objekt/-Variable/-Methode in Modellen

Um Eigenschaften, welche von mehreren Clients verwendet werden, ohne gröbere Probleme bei der Referenzenverwaltung ändern zu können, muss die Information vorhanden sein, welcher Knoten der Elternknoten ist, also eine Referenz darauf hat. Die zugehörigen *HasModelParent* Referenzen müssen an alle Instanzen weitergegeben werden. Clients können davon ausgehen, dass sie über Änderungen informiert werden, insofern die Modellierungsregeln (siehe 2.4.6) vorhanden sind.

3.3 Ansichten

Um die Anzahl der sichtbaren Knoten und Referenzen zu reduzieren, wurden sogenannte Ansichten eingeführt. Diese Ansichten können zur Einteilung von Knoten und Referenzen in Gruppen genutzt werden, sodass bei Wartungsangelegenheiten nur die dafür interessanten Knoten und Referenzen sichtbar sind.

3.4 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.

Literatur

[1] W. Mahnke, et. al.: *OPC Unified Architecture*, Springer Berlin Heidelberg (2009)

[2] The OPC Unified Architecture e-book, `http://www.commsvr.com/UAModelDesigner/Index.aspx`, Stand: 02.05.2010