Das Adressraum-Konzept in OPC Unified Architecture

Stefan Szucsich
e0728333@student.tuwien.ac.at
Vienna University of Technology, Vienna, Austria

4. Mai 2010

Kurzfassung

Web-Services eröffnen der Integration von Automationssystemen in existierende IT-Netze neue Möglichkeiten. Ein aus dem Konzept der serviceorientierten Architektur hervorgetretener Standard ist OPC Unified Architecture (OPC UA) von der OPC Foundation. Zunächst wird ein kurzer Überblick über OPC UA gegeben. Anschließend befasst sich dieser Artikel mit dem in OPC UA verwendeten Adressraum-Konzept. Hierzu werden die definierten Konstrukte und Konzepte des Meta-Modells anschaulich erklärt.

1 Einführung

Die Integration von Gebäude-Automationssystemen in bestehende IT-Netze erweist sich häufig als kompliziert und schwierig. Aus diesem Grund sind Automationssysteme heutzutage noch oftmals als isolierte Inseln anzutreffen. In den letzten Jahren haben sich allerdings durch die Entwicklung des Webs neue Möglichkeiten für eine einfachere Integration ergeben: Web-Services finden mehr und mehr Einzug in die Welt der Automationssysteme. Dies führt — an Stelle von informationsorientierten Ansätzen — vermehrt zu serviceorientierten Ansätzen der Systemintegration.

Neben dem von der OPC Foundation spezifizierten OPC Unified Architecture (OPC UA) [1] gibt es aktuell zwei weitere Standards von anderen Organisationen. Diese sind Open Building Information Exchange (oBIX) [2] von der Organization for the Advancement of Structured Information Standards (OASIS) sowie Building Automation and Control Network - Web Services (BACnet/WS) [3] von der American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE).

Der folgende Abschnitt soll einen kurzen Überblick über OPC UA geben. Im dritten Abschnitt wird das Adressraum-Konzept von OPC UA im Detail beschrieben. Abschnitt 4 fasst die wichtigsten Punkte zusammen und beschließt diesen Artikel.

2 OPC UA im Überblick

Zu den wichtigsten Erneuerungen von OPC UA zählen dessen Plattform-Unabhängigkeit sowie die serviceorientierte Architektur. Während die bekanntesten Vorgänger von OPC UA — OPC Data Access (OPC DA), OPC Alarm & Events (OPC A&E) und OPC Historical Data Access (OPC HDA) — auf der plattformgebundenen Component Object Model (COM) bzw. Distributed Component Object Model (DCOM) Technologie von Microsoft aufbauen, führt die serviceorientierte Architektur von OPC UA gemeinsam mit der Verwendung der Web-Services Technologie zur Plattform-Unabhängigkeit. Ein weiterer Motivationsgrund für OPC UA war die Anforderung der Verwendung komplexer Daten. Mehr Informationen über das klassische OPC sowie dessen Nachfolger können in [4] nachgelesen werden.

Die folgenden Unterabschnitte befassen sich mit den OPC UA Komponenten, dem Kommunikations-Stack, dem Meta-Modell sowie den Services.

2.1 OPC UA Komponenten

Die OPC UA Architektur spezifiziert Clients und Server als Systemkomponenten [1]. Clients können per Request Informationen von einem oder mehreren Servern in ihrem System anfordern. Die Funktionalität von Clients bzw. Servern wird durch Applikationen festgelegt. Diese Applikationen verwenden die Client bzw. Server API um auf den darunterliegenden Kommunikations-Stack zugreifen zu können.

2.2 Kommunikations-Stack

Der Kommunikations-Stack bildet gemeinsam mit dem Meta-Modell das Fundament von OPC UA. Er besteht aus zwei unterschiedlichen Transportmechanismen die je nach Anforderung ausgewählt werden können [5]. Für effiziente Datentransfers kann das binäre TCP Protokoll *UA Native* verwendet werden. Hierbei werden die Daten einfach binär repräsentiert. Es eignet sich dadurch besonders bei stark beschränkten Netzwerk- und Computer-Ressourcen. Alternativ dazu ist das *UA Web Services* Protokoll spezifiziert. Hier werden die Daten im XML-Format repräsentiert und mittels Simple Object Access Protocol (SOAP) über HTTP übertragen.

2.3 Meta-Modell

Das Meta-Modell definiert Regeln und Basisblöcke die benötigt werden um Informationsmodelle zu bilden. Zusätzlich dazu werden Basistypen festgelegt. Anhand des Meta-Modells können anwendungsspezifische Informationsmodelle aufgebaut und in weiterer Folge verwendet werden. Details über den Aufbau des Meta-Modells folgen in Abschnitt 3.

2.4 Services

Wie bereits erwähnt basiert OPC UA auf einer serviceorientierten Architektur. Die gesamte Funktionalität eines OPC UA Servers wird den Clients anhand von Services zur Verfügung gestellt. Diese Services werden durch die jeweiligen

Request- und Response-Nachrichten definiert und zu Gruppen zusammengefasst. Jede Gruppe von Services ermöglicht den Zugriff auf bestimmte Aspekte des Servers. [1] enthält eine Zusammenfassung aller Service-Gruppen.

Da ein Server nicht alle spezifizierten Services anbieten muss, werden in der OPC UA Spezifikation zusätzlich Profile definiert. Anhand von Profilen wird vorgegeben welche Services tatsächlich implementiert werden müssen.

3 Das OPC UA Adressraum-Konzept

Der Adressraum in OPC UA ist definiert als die Gesamtheit aller Informationen die ein OPC UA Server für Clients sichtbar macht [1]. Die beiden Begriffe Adressraum und Meta-Modell stehen eng miteinander in Verbindung [6]: Im Meta-Modell werden alle notwendigen Konstrukte definiert aus denen der Adressraum aufgebaut wird. In den folgenden Unterabschnitten werden diese im Detail beschrieben. Die Informationen hierfür sind im Speziellen [4] entnommen. Die in den Abbildungen verwendete graphische Notation ist in [7] beschrieben.

3.1 Knoten und Referenzen

Die Grundlage des Adressraums bilden *Knoten* und *Referenzen*. Ein Knoten gehört je nach Verwendungszweck zu einer bestimmten *Knotenklasse*. Jeder Knoten wird durch *Attribute* näher beschrieben. Die Menge aller Attribute, die einen Knoten beschreiben, wird durch die Knotenklasse vorgegeben. Werden zusätzliche Informationen benötigt, so müssen *Eigenschaften* verwendet werden. Es gibt jedoch auch einige Attribute die alle Knoten — unabhängig von ihrer Knotenklasse — gemeinsam haben. Eines dieser Attribute ist die `KnotenID`. Sie identifiziert einen Knoten eindeutig innerhalb eines Servers und bildet somit die Adressierungsgrundlage für Services. Eine vollständige Auflistung der gemeinsamen Attribute findet sich in [4].

Referenzen beschreiben Beziehungen zwischen je zwei Knoten. Sie können keine Attribute oder Eigenschaften besitzen und sind eindeutig identifizierbar durch den Quell- und Zielknoten, den *Referenztyp* und die Richtung. Es können zwei Arten von Referenzen unterschieden werden — symmetrische und asymmetrische. Für symmetrische Referenzen spielt die Richtung der Referenz keine Rolle, hingegen für asymmetrische sehr wohl. Als Beispiel für eine symmetrische Beziehung gilt die "hat-gleichen-Typ-wie"-Beziehung. Diese ist für beide Richtungen äquivalent. Auf der anderen Seite gilt "ist-Superklasse-von" in einer Richtung bzw. "ist-abgeleitet-von" in der anderen Richtung als asymmetrische Beziehung. Weiters kann zwischen unidirektionalen und bidirektionalen Referenzen unterschieden werden.

Referenzen können aber auch als Zeiger verstanden werden: Ein Zeiger gehört zum Quellknoten und speichert die Knoten-ID des Zielknotens. Zusätzlich wird der OPC UA Server, der Referenztyp und die Richtung direkt im Quellknoten abgelegt. Zeiger können etwa auch auf nicht (mehr) existente Knoten verweisen oder Schleifen bilden. Abbildung 1 zeigt zwei Knoten deren Referenzen als Zeiger gespeichert sind.

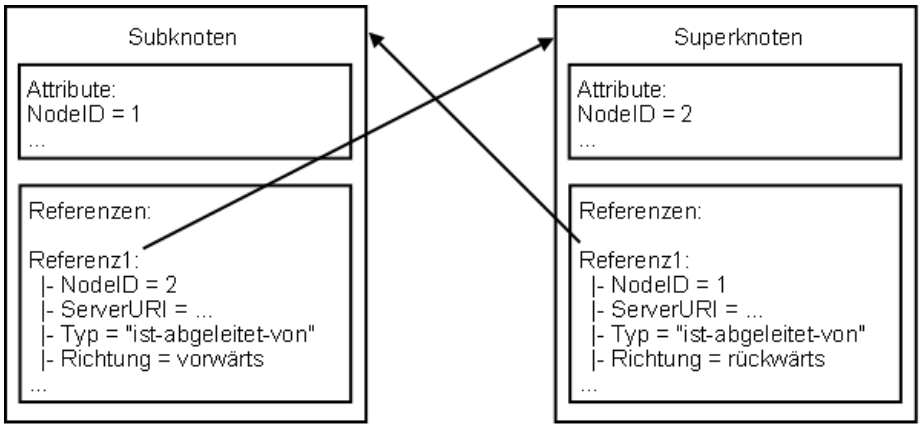

Abbildung 1: Knoten mit Referenzen als Zeiger [4]

3.2 Referenztypen

Da Referenzen selbst keine Attribute oder Eigenschaften besitzen können, erfolgt die semantische Bestimmung von Referenzen durch *Referenztypen*. Jede Referenz besitzt einen bestimmten Referenztyp und somit dessen Eigenheiten. Die OPC UA Spezifikation [1] definiert einige grundlegende Referenztypen. Ein OPC UA Server kann jedoch auch selbst Referenztypen definieren. Referenztypen sind hierarchisch organisiert.

Im Gegensatz zu Referenzen werden Referenztypen als Knoten im Adressraum repräsentiert. Dadurch können Clients mittels Zugriff auf diese Knoten im Adressraum des Servers Informationen über Referenzen erhalten. Zusätzlich zu den bereits erwähnten gemeinsamen Attributen besitzen Referenztypen weitere Attribute. Die beiden wichtigsten sind die Attribute `IsAbstract` und `Symmetric`, beide vom Typ `Boolean`. Das `Symmetric` Attribut zeigt an ob es sich um eine symmetrische Beziehung handelt. Mit Hilfe des `IsAbstract` Attributs lassen sich abstrakte Referenztypen definieren, die nur zu organisatorischen Zwecken in der Referenztyp-Hierarchie verwendet werden können.

Abbildung 2 zeigt die OPC UA Referenztyp-Hierarchie. Wie daraus ersichtlich ist, werden zwei abstrakte Referenztypen die von der Wurzel der Referenztyp-Hierarchie abgeleitet werden unterschieden. Diese sind hierarchische bzw. nicht hierarchische Referenzen. Hierarchische Referenzen sollten verwendet werden, wenn eine Hierarchie modelliert werden soll. Ansonsten sollten nicht hierarchische Referenzen benutzt werden. Um sicher zu stellen, dass ein Referenztyp nicht von beiden zuvor genannten Referenztypen abgeleitet wird, erlaubt OPC UA nur eine einfache Vererbung, d.h. jeder Referenztyp besitzt maximal einen Basistypen.

3.3 Objekte, Variablen und Methoden

Die Knotenklassen *Objekt*, *Variable* und *Methode* verhalten sich ähnlich den gleichnamigen Konzepten der objektorientierten Programmierung. Objekte können aus Variablen, Methoden und anderen Objekten bestehen. Sie dienen der Strukturierung des Adressraums und enthalten — im Gegensatz zu Va-

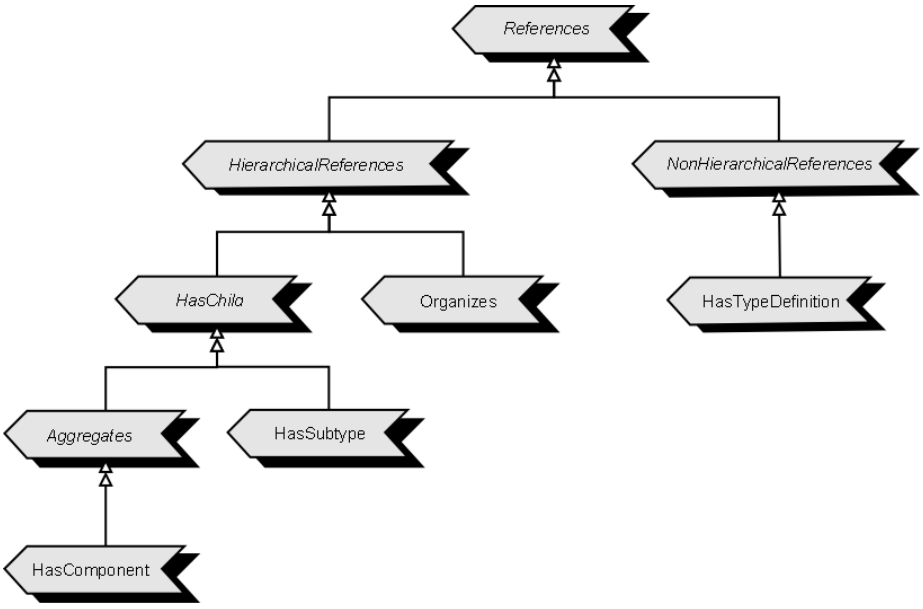

Abbildung 2: Die OPC UA Referenztyp-Hierarchie [4]

riablen — keine Werte. Weiters können Objekte als sogenannte *Event-Notifier* agieren. Dies ermöglicht Clients automatisch über Ereignisse, wie zum Beispiel der Änderung einer Variable, benachrichtigt zu werden.

Variablen repräsentieren Werte eines bestimmten Typs. Sie besitzen eine Vielzahl von zusätzlichen Attributen die etwa auch den genauen Datentyp festlegen. Mit Hilfe dieser Attribute lassen sich auch mehrdimensionale Datenarrays erzeugen. Eine Liste der Attribute findet sich in [4].

Methoden können von Clients aufgerufen werden und liefern ein bestimmtes Ergebnis zurück. Sie legen die Ein- bzw. Ausgabeargumente fest und werden mittels Services ausgeführt.

Abbildung 3 zeigt ein Objekt namens `Lüfter` bestehend aus einem weiteren Objekt (`Drehzahlmesser`), der Variable `Status` und den Methoden `ein` bzw. `aus`. Die einzelnen Knoten sind mit Referenzen des Typs `HasComponent` verbunden.

3.4 Objekt- und Variablentypen

OPC UA erlaubt Typinformationen auf Variablen- sowie auf Objektebene. Sowohl *Objekttypen* als auch *Variablentypen* lassen sich in einfache und komplexe Typen unterscheiden. Wie auch bei den Referenztypen gibt es abstrakte Objekt- bzw. Variablentypen. Diese können nicht referenziert werden und dienen der Strukturierung der jeweiligen Typ-Hierarchie.

Simple Objekttypen legen eine bestimmte Semantik für das jeweilige Objekt fest. Bei komplexen Objekttypen wird zusätzlich eine Struktur von Knoten, die unterhalb des Objekttyps liegt, definiert. Als Knoten innerhalb eines komplexen Objekttyps können Objekte, Methoden und Variablen verwendet werden. Sie werden *Instanzdeklarationen* genannt, da sie keine tatsächlichen Instanzen

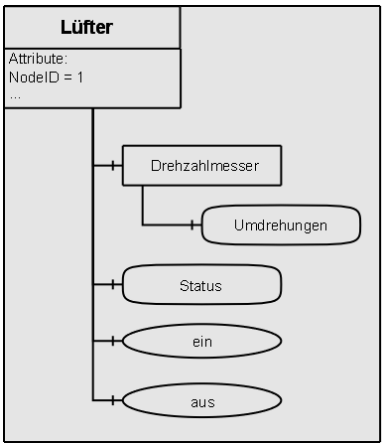

Abbildung 3: Objekte, Variablen und Methoden

sind. Jede Instanz eines komplexen Objekttyps besitzt die durch den Objekttyp definierte Struktur ebenfalls. Abbildung 4 zeigt einen komplexen Objekttyp namens `LüfterTyp` und ein zugehöriges Objekt. Das Objekt `Lüfter` referenziert den Objekttypen mit einer Referenz vom Typ `HasTypeDefinition`.

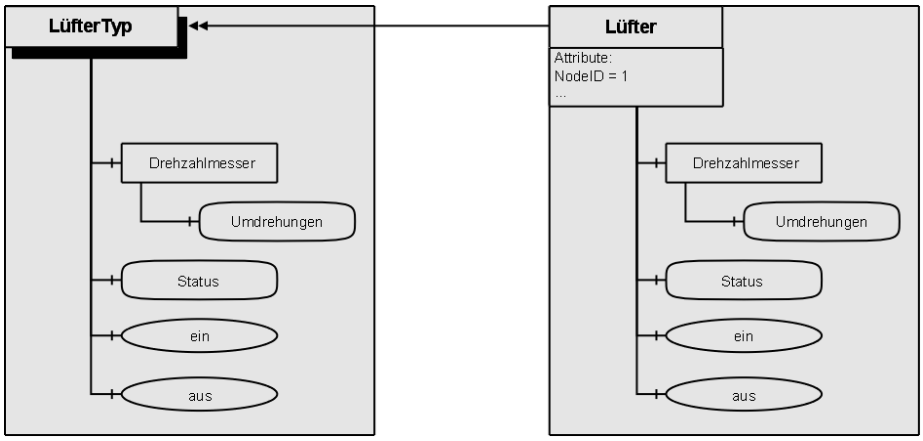

Abbildung 4: Ein komplexer Objekttyp mit zugehörigem Objekt

Simple bzw. komplexe Variablentypen verhalten sich ähnlich den jeweiligen Objekttypen. Die innere Struktur eines komplexen Variablentyps darf allerdings nur aus Variablen bestehen. Objekte und Methoden dürfen nicht verwendet werden.

Jede Instanzdeklaration muss genau eine *Modellierungsregel* referenzieren. Damit wird festgelegt ob die Instanzdeklaration verpflichtend, optional oder eine Beschränkung ist. Dies wird durch die `NamingRule` Eigenschaft der Modellierungsregel bestimmt. Beispiel für eine Beschränkung ist die Einschränkung von Referenzen: Eine Instanz eines Typs soll eine bestimmte Anzahl an Instanzen eines anderen Typs referenzieren.

3.5 Eigenschaften und Datenvariablen

Datenvariablen und *Eigenschaften* lassen sich aufgrund ihrer Semantik nicht immer eindeutig voneinander unterscheiden. Eigenschaften spiegeln üblicherweise die Charakteristik eines Knotens wieder. Sie sind einfach und können nie die Quelle einer hierarchischen Referenz sein. Datenvariablen können hingegen komplex aufgebaut sein und repräsentieren die Daten eines Objektes. Eine Datenvariable könnte beispielsweise die durch einen Sensor gemessene Temperatur festhalten. Die Temperatureinheit könnte in einer der Datenvariable gehörenden Eigenschaft festgelegt sein.

Es gibt aber auch einige syntaktische Unterschiede zwischen Eigenschaften und Datenvariablen. Eine Eigenschaft muss zumindest von einem Knoten referenziert werden. Zudem besitzen alle Eigenschaften den gleichen Variablentyp `PropertyType`. Datenvariablen müssen zu einem Objekt und einem Objekttyp gehören. Sie können aber auch zusätzlich von Variablen oder Variablentypen referenziert werden. Der Referenztyp ist dabei allerdings ein anderer als bei Referenzen auf Eigenschaften.

3.6 Datentypen

Jeder Datentyp wird im Adressraum durch einen Knoten des Datentyps `NodeClass` repräsentiert. Dies ermöglicht OPC UA Servern das Definieren von eigenen Datentypen. Das einzige zusätzliche Attribut des Datentyps `NodeClass` ist das bereits bekannte `IsAbstract` Attribut. Wie Referenztypen sind auch Datentypen hierarchisch strukturiert. Dabei kommt ebenfalls das Prinzip der einfachen Vererbung zur Geltung.

In OPC UA lassen sich vier verschiedene Arten von Datentypen unterschieden. *Built-in Datentypen* sind in der OPC UA Spezifikation festgelegte Datentypen (siehe Abbildung 5). Diese sind nicht erweiterbar. Es handelt sich hierbei um grundlegende Datentypen wie `Boolean`, `Byte` und `Double`, aber auch um OPC UA spezifische Datentypen wie `NodeID`.

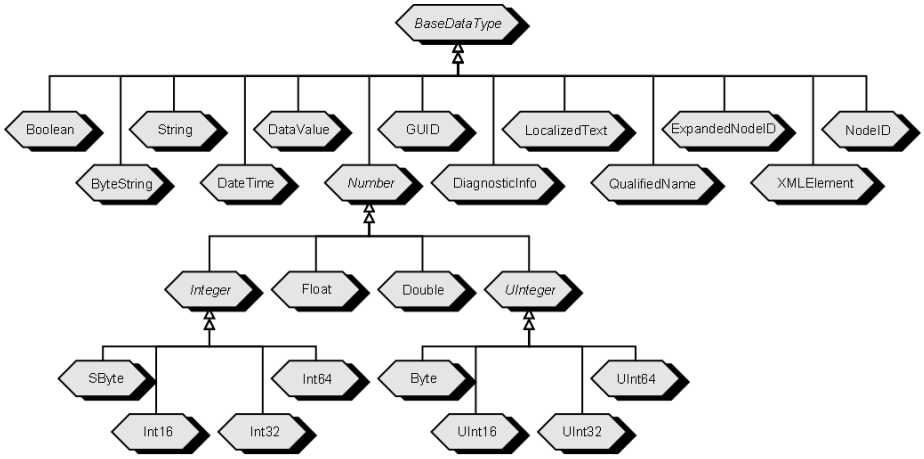

Abbildung 5: Die OPC UA Built-in Datentypen [4]

Einfache Datentypen sind Subtypen von Built-in Datentypen. Sie ermöglichen

die Erweiterung von Built-in Datentypen um zusätzliche Informationen und können per Informationsmodell definiert werden.

Aufzählungs-Datentypen stellen Mengen von benannten Werten dar und werden vom abstrakten Datentyp `Enumeration` abgeleitet. Sie können ebenfalls per Informationsmodell definiert werden. Zu den Aufzählungs-Datentypen zählt beispielsweise der Datentyp `NodeClass`.

Strukturierte Datentypen repräsentieren strukturierte Daten und werden vom abstrakten Datentyp `Structure` abgeleitet. Sie erlauben die Definition von komplexen Datentypen. Im Gegensatz zu den zuvor genannten Arten von Datentypen, muss für strukturierte Datentypen ein benutzerdefiniertes Encoding zur Verfügung gestellt werden. Der Datentyp `Argument` zur Spezifizierung der Argumente einer Methode gehört zu dieser Gruppe von Datentypen.

3.7 Views

Eine Untermenge des Adressraums wird als *View* bezeichnet [1]. Eine View beschränkt die Anzahl der für einen Client sichtbaren Knoten und Referenzen beispielsweise auf das Notwendigste um eine bestimmte Aufgabe durchführen zu können. Views können auf zweierlei Arten betrachtet werden: Einerseits wird eine View als Knoten im Adressraum dargestellt. Dieser Knoten repräsentiert den Einstiegspunkt der View, von dem aus alle Knoten der View direkt oder indirekt mittels Referenzen erreichbar sein müssen. Andererseits kann eine View auch als Filter betrachtet werden: Beim Durchsuchen des Adressraums kann die Knoten-ID des View-Knotens als Filterparameter verwendet werden. Dadurch können Server Referenzen zu anderen Knoten begrenzen. Somit wird nur ein Teil des Adressraums für Clients sichtbar. Weiters können Views als Event-Notifier verwendet werden.

3.8 Events

Events werden durch den OPC UA Server per Benachrichtigung an alle bei einem Event-Notifier eingetragenen Clients gesendet. Jedem Event ist ein bestimmter *Eventtyp* zugeordnet. Eventtypen sind von OPC UA durch eine erweiterbare Hierarchie strukturiert. Sie werden im Adressraum als Knoten der Klasse `ObjectType` repräsentiert.

Events selbst sind typischerweise nicht als Knoten im Adressraum sichtbar. Für Eventtypen dieser Events werden abstrakte Objekttypen verwendet.

3.9 Historischer Zugriff

Der historische Zugriff in OPC UA unterteilt sich in den Zugriff auf historische Daten, die Historie von Events und die Historie der Struktur des Adressraums.

Die Historie von Daten erlaubt den Zugriff auf den Verlauf von Werten von Variablen, beispielsweise dem Wert eines Drucksensors innerhalb der letzten fünf Stunden. Das Aufzeichnen der Werte sowie der Zugriff auf die Historie wird mittels eigener Attribute der Knotenklasse `Variable` geregelt.

Der Zugriff auf historische Events erlaubt das Abfragen der aufgetretenen Ereignisse innerhalb eines bestimmten Zeitraums. Dies geschieht über die jeweiligen Event-Notifier, d.s. Objekte oder Views.

Neben den Zugriffen auf historische Daten bzw. Events, können auch Informationen über die Änderungen der Struktur des Adressraums gesammelt werden. Dazu dient die Eigenschaft `NodeVersion` eines Knotens sowie das Event `ModelChangeEvent`. Ein Server muss einem Knoten entweder beides oder keines von beiden zur Verfügung stellen.

4 Zusammenfassung

In den vorhergehenden Abschnitten wurden die Konzepte und Konstrukte zum Aufbau des Adressraums in OPC UA vorgestellt. Die grundlegendsten Bestandteile sind Knoten und Referenzen. Jeder Knoten gehört zu einer Knotenklasse und wird durch Attribute und Eigenschaften beschrieben. Nahezu alle weiteren Konstrukte werden als Knoten im Adressraum dargestellt.

Referenzen stellen Beziehungen zwischen Knoten dar. Sie können durch Quell- und Zielknoten, Referenztyp und Richtung eindeutig identifiziert werden, werden aber nicht als Knoten im Adressraum abgelegt. Die Bedeutung von Referenzen wird durch den Referenztyp festgelegt. Referenztypen sind hierarchisch organisiert und werden durch Knoten im Adressraum repräsentiert.

Zu den weiteren Konstrukten gehören Objekte, Variablen und Methoden. Objekte werden zur Strukturierung des Adressraums verwendet und können aus Variablen, Methoden sowie anderen Objekten bestehen. Sie besitzen einfache oder komplexe Objekttypen. Komplexe Objekttypen legen zusätzlich zur Semantik der Objekte eine Struktur von Instanzdeklarationen fest. Jede Instanz besitzt diese Struktur ebenfalls. Variablen repräsentieren Werte eines bestimmten Typs. Wie auch Objekte gehören Variablen einfachen oder komplexen Typen an. Methoden werden von Clients mittels Services aufgerufen und liefern Ergebnisse zurück.

Um zusätzliche Informationen in Knoten ablegen zu können werden Eigenschaften verwendet. Datenvariablen repräsentieren hingegen Daten eines Objektes.

Datentypen werden als Knoten im Adressraum abgelegt. Sie sind hierarchisch organisiert und unterteilen sich in vier Gruppen. Neben den Built-in Datentypen gibt es einfache und strukturierte Datentypen sowie Aufzählungs-Datentypen.

Eine View ist eine Untermenge des Adressraums. Sie beschränkt die für Clients sichtbaren Knoten und Referenzen. Somit können nicht benötigte Teile eines großen Adressraums ausgeblendet werden.

Historischer Zugriff kann bei OPC UA auf Daten, Events sowie die Struktur des Adressraums erfolgen.

Literatur

[1] OPC Foundation. *OPC Unified Architecture, Part 1: Overview and Concepts*, February 2009. http://www.opcfoundation.org.

[2] OASIS. *oBIX 1.0 Committee Specification 01*, December 2006. http://www.obix.org.

[3] American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. *ANSI/ASHRAE Addendum c to ANSI/ASHRAE Standard 135-2004*, October 2006. http://www.bacnet.org.

[4] *OPC Unified Architecture*. Springer Berlin Heidelberg, 2009.

[5] T. Hannelius, M. Salmenpera, and S. Kuikka. Roadmap to adopting OPC UA. In *INDIN 2008. 6th IEEE International Conference on Industrial Informatics*, pages 756 –761, July 2008.

[6] S. Leitner and W. Mahnke. OPC UA – Service-oriented Architecture for Industrial Applications. Technical report, ABB Corporate Research Center.

[7] The OPC Unified Architecture e-book. http://www.commsvr.com/UAModelDesigner/Index.aspx. Eingesehen am 20.04.2010, 09:54.