OPC Unified Architecture
Information Modeling

Nodes und References

Nodes

Grundlage der Informationsmodellierung in OPC UA sind *Nodes* und *References* zwischen diesen Nodes und sind Grundelemente des *Object Model*. Jeder Node ist einer *Node Class* zugewiesen.

Objects und ihre Komponenten werden im Address Space durch eine Menge an Nodes mit Attributen beschrieben und untereinander durch Referenzen verbunden.

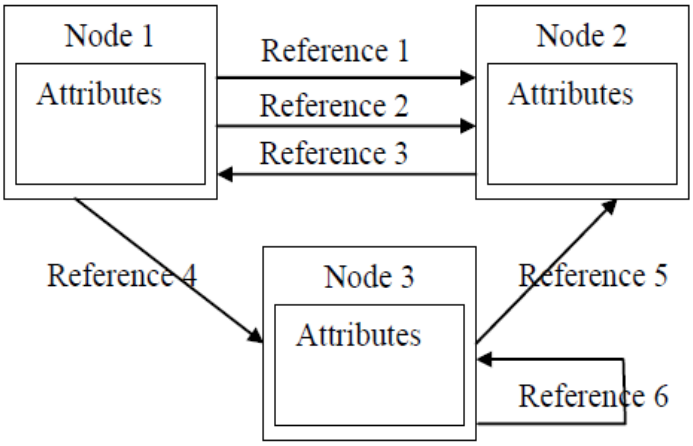

Abbildung 1: Nodes und References zwischen Nodes; Quelle "OPC Unified Architecture", p. 22

Die Base Node Class enthält alle Attribute die jeder Node beinhalten muss, abhängig von der Node Class werden weitere Attribute hinzugefügt. Die Attribute der Base Node Class sind:

Attribut-Name	Daten-Typ	Beschreibung
NodeId	NodeId	Ist der *Unique-Identifier* eines Nodes in einem OPC UA Server und wird verwendet um den Node in OPC UA Services zu adressieren
NodeClass	NodeClass	Identifiziert die Node Class
BrowseName	QualifiedName	Dieses Attribut speichert nicht-lokalisierte human-readable Namen zum Aufbau der Pfade im Address Space
DisplayName	LocalizedText	Enthält den Namen des Nodes, der dem Benutzer angezeigt werden soll
Description	LocalizedText	Enthält eine Beschreibung des Nodes
WriteMask	UInt32	Ist optional; spezifiziert die Attribute eines Nodes die vom Client geschrieben werden können
UserWriteMask	UInt32	Ist optional; spezifiziert die Attribute eines Nodes die vom Client geschrieben werden können, unter Berücksichtigung der User Access Rights. Die UserWriteMask kann die WriteMask nur weiter einschränken

References

Eine Referenz wird benutzt um zwei Nodes miteinander in Verbindung zu setzen. Referenzen sind Instanzen von ReferenceType Nodes. ReferenceType Nodes sind im Address Space abgebildet und sind vom Typ ReferenceType Node Class. Eine Reference Instanz ist Teil eines Nodes aber keine Node Class wird benutzt um Referenzen darzustellen. Daher erbt die ReferenceType Node Class alle Attribute von der Basenode Class, dies erlaubt den Clients Informationen über Referenzen in einem OPC UA Server abzufragen, in dem sie auf die Nodes im Address Space eines OPC UA Servers zugreifen

Die OPC UA Spezifikation definiert verschiedene ReferenceTypes. Diese modellieren die Beziehungen von Nodes zu einander. Das Konzept von ReferenceTypes ist ein erweiterbares, das bedeutet man kann auch eigene ReferenceTypes definieren. Um Referenzen organisieren zu können sind Referenzen in einer Typ Hierarchie angeordnet.

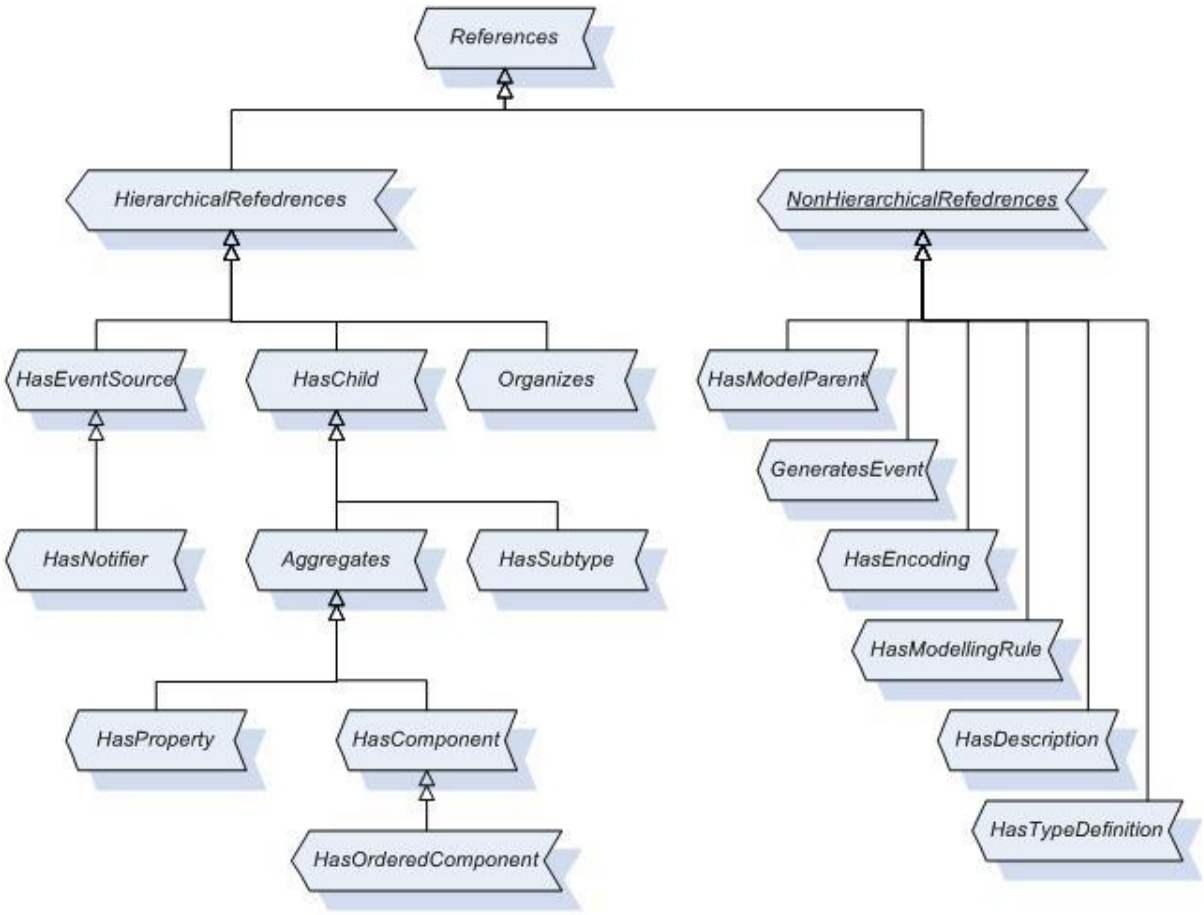

Abbildung 2: Standard ReferenceType Hierarchy; Quelle: http://www.commsvr.com/

Die für alle Referenzen zusätzlichen Attribute sind:

Attribut	Datentyp	Beschreibung
IsAbstract	Boolean	Beschreibt ob der Referenz-Typ Verbindungen abbildet oder ob er dieser nur für Organisation der Referenztyp Hierarchie verwendet wird
Symmetric	Boolean	Ist die Referenz symmetrisch → hat sie in beide Richtungen die gleiche Bedeutung
InverseName	LocalizedText	Optional; Beschreibt die Semantik der Referenz in inverser Richtung

Referenzen zeigen auf existente wie auch auf nicht existente Nodes. Referenzen können auf Nodes innerhalb eines OPC UA Servers oder auf Nodes in einem OPC UA Server zeigen der nicht für den Client erreichbar ist. Weiters können Referenzen auch noch zu Schleifen führen.

Referenzen unterliegen der Einschränkung das Referenzen vom selben Typ, mit der selben *direction* zwischen zwei Nodes nicht zweimal vorkommen dürfen, da Referenzen über den Source Node, den Target Node und den Reference Type eindeutig definiert sind. Dies gilt auch für Sybtypen von RefrenceTypes. (Es ist erlaubt zwei Nodes mit unterschiedlichen Typen von hierarchischen Referenzen zu verbinden, aber nicht dieselben zwei Nodes einmal mit HasComponent und ein weiteres Mal mit einem Sybtyp von HasComponent)

Objects, Variables und Methods

Die wichtigsten NodeClasses in OPC UA sind Object, Variable und Method. Objects besitzen Variablen und Methoden und feuern Events. Nodes der NodeClass Object werden verwendet um den Address Space zu strukturieren, anders als in OOP können sie nur Variables und Methods oder weitere Objects gruppieren.

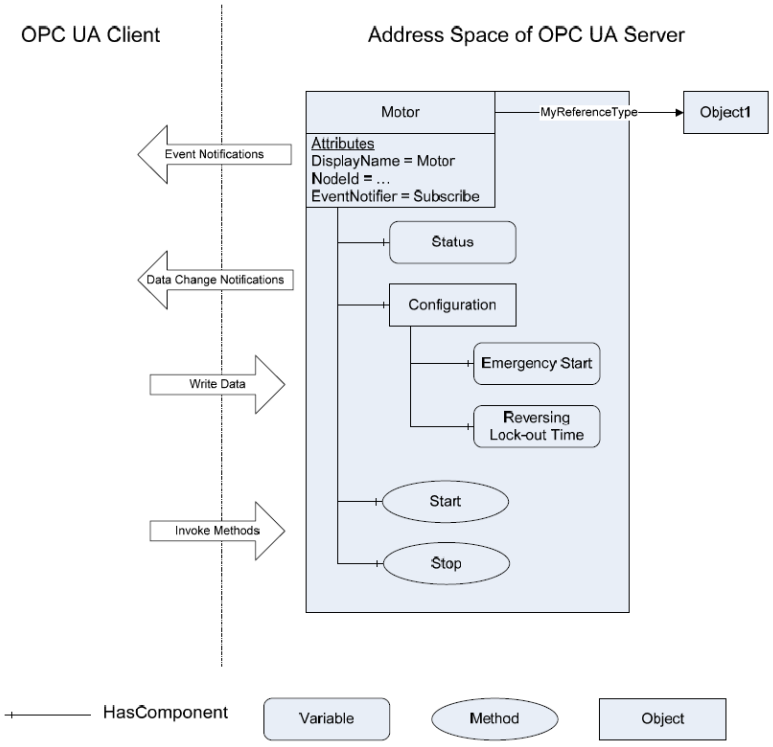

Abbildung 3: Überblick von Objects, Variables, Methods; Quelle: „OPC Unified Architecture“, p. 31

Zusätzliche Attribute der BaseObjectType Nodes:

Attribut	Datentyp	Beschreibung
EventNotifier	Byte	Eine Bitmaske die angibt ob der Object Node benutzt werden kann um ein Event zu subscriben. Weiters gibt sie an ob die History of Events verfügbar u./o. veränderbar ist.

Zusätzliche Attribute der BaseVaribleType Nodes:

Attribut	Datentyp	Beschreibung
Value	Nicht fixiert; durch andere Attribute und Typ definiert	Der/Die aktuelle/n Wert/e der Variable. Der Datentyp der Variable ist definiert durch DataType, ValueRank und ArrayDimensions
DataType	NodeId	Datentypen werden durch Nodes im Address Space repräsentiert. Dieses Attribute enthält eine NodeId
ValueRank	Int32	Identifiziert ob die Variable ein Array ist und wenn es ein Array ist spezifiert es die Dimension des Array
ArrayDimensions	UInt32[]	Optional; Enthält für jede Dimension des Array die Länge der korresponierenden Dimension
AccessLevel	Byte	Eine Bitmaske; gibt an ob das Value Attribut lesbar, schreibbar und ob die history von Value lesbar und änderbar ist
UserAccessLevel	Byte	Eine Bitmaske; gleich wie AccessLevel nur unter Beachtung von User Access Rights
MinimumSamplingIntervall	Duration	Optional; gibt an in welchem Intervall der OPC UA Server Veränderungen des Werts detektieren kann. (In welchem Intervall ein Device zu pollen ist)
Historizing	Boolean	Gibt an ob der Server eine History anlegen soll

Zusätzliche Attribute/Properties der Method NodeClass:

Attribut	Datentyp	Beschreibung
Executable	Boolean	Das Flag legt die momentane Ausführbarkeit fest
UserExecutable	Boolean	Dasselbe wie Executable beachtet aber die user access rights
Property		
InputArguments	Argument[]	Optional; definiert den geordneten Array von Inputargumenten
OutputArguments	Argument[]	Optional; geordneter Array an Outputargumenten

Argument-Struktur:

Name	Datentyp	Beschreibung
Name	String	Name des Arguments
DataType	NodeId	NodeId des Datentyp Nodes
ValueRank	Int32	Definiert ob das Argument; Skalar, Array oder Matrix ist
ArrayDimensions	UInt32[]	Definiert die der Arrays oder der Matrizen
Description	LocalizedText	Die Beschreibung

Daten Typen

Alle Attribute außer Value Attribute und VariableTypes haben einen fixen Datentyp. Datentypen werden ebenfalls im Address Space abgebildet. Der Server kann auch weitere Datentypen definieren, durch das Abbilden im Address Space können Clients Informationen über diese Datentypen einholen.

OPC UA unterscheidet vier Arten von DataTypes:

1. *Built-In DataTypes*: Sie können nicht erweitert werden. Zu ihnen zählen neben den „Standardtypen“ wie Int32, Double, Boolean auch UA spezifische Typen wie NodeId, LocalizedText und QualifiedName
2. *Simple DataTypes:* Sind SubTypen von Built-In DataTypes und können von Information Models definiert werden
3. *Enumeration DataTypes*
4. *Structured DataTypes*: Ein Beispiel wäre DataType Argument

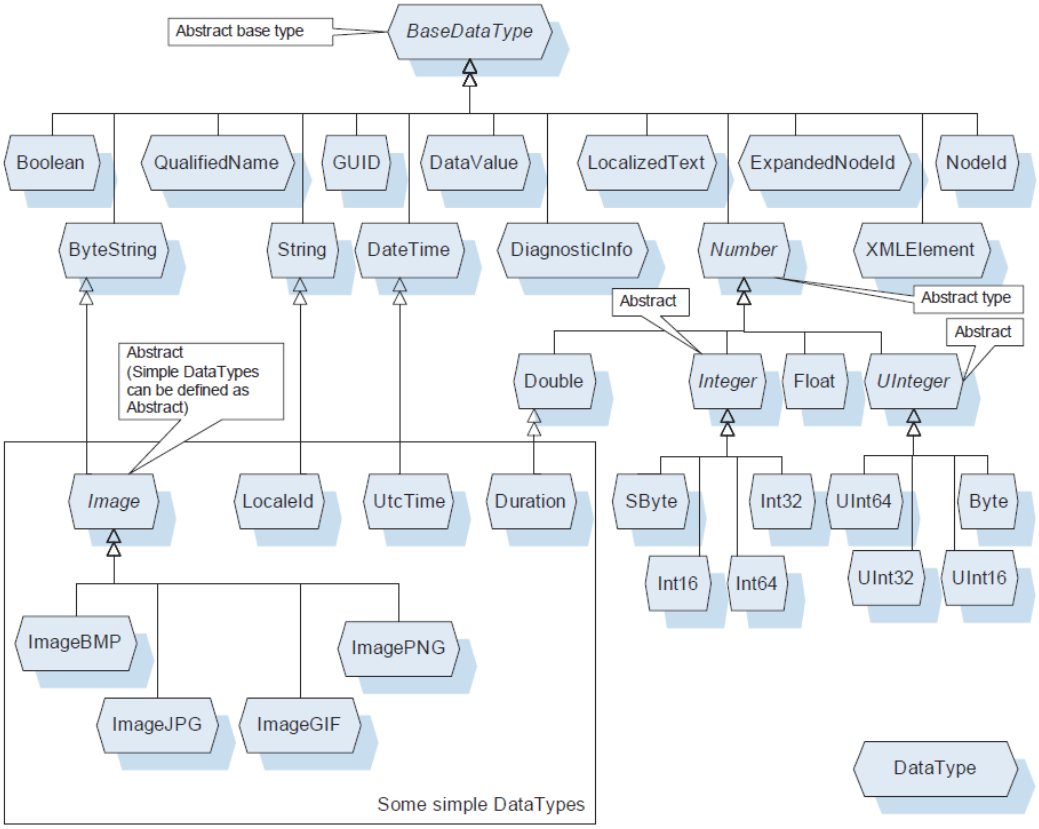

Abbildung 4: Datentyp Hierarchie der Built-In Data Types; Quelle: "OPC Unified Architecture", p.63

Modelling Rules

Jede Instanz welche durch einen Typ Defintion referenziert wird (ObjectType, VariableType) wird zur InstanceDeclaration wenn es eine dazugehörige *ModellingRule* gibt. Eine ModellingRule gibt an wie die Instanzen des ObjectType zueinander in Beziehung stehen. Es gibt drei Wahlmöglichkeiten:

1. Die erste Möglichkeit ist dass die InstanceDeclaration *mandatory* ist. Das bedeutet jede Instanz muss ein Gegenstück haben, vom gleichen Typ der InstanceDeclaration oder einem Subtyp dieses Typs.
2. Die InstanceDeclaration ist *optional.*
3. *Constraint*: Dies bedeutet die InstanceDeclaration definiert Beschränkungen für die Instanzen der TypeDefinition. Beispiel einer solchen Beschränkung ist eine Kardinaltitätseinschränkung: Ein Motor besteht aus 1 bis maximal 12 Zylinder, jeder Zylinder hat 2 oder 4 Ventile.

Modelling Rules werden durch Objekte vom Typ ModellingRule repräsentiert. Jede ModellingRule hat eine Variable vom Typ NamingRule. InstanceDeclarations referenzieren ein ModellingRule Objekt mit dem Referenztyp HasModellingRule zum Spezifizieren ihrer ModellingRule:

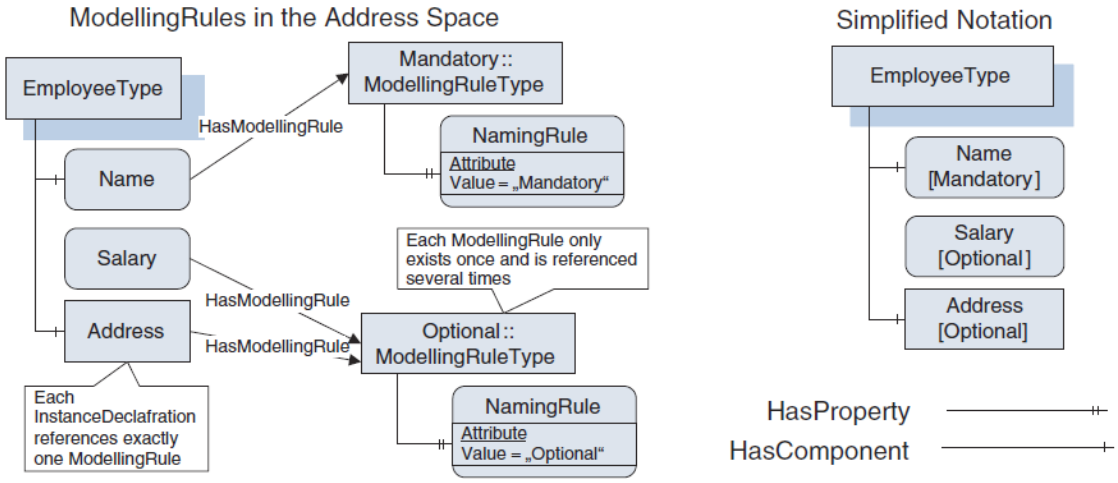

Abbildung 5: Modellierungsregeln im Address Space; Quelle: "OPC Unified Architecture", p. 49

Die ModellingRules Optional und Mandatory spezifizieren nicht wie ein Server mit Instanzdeklarationen umzugehen hat wenn eine neue Instanz erzeugt wird. Der Server kann neue InstanceDeclaration Nodes anlegen oder auf bereits existierende verweisen. Instanzen einer TypeDefinition müssen nur auf eine Instanz referenzieren die den gleichen Browsepath und den gleichen Typ besitzen. Ein Server kann daher so etwas wie static class Variable erzeugen. → Variablen und Methoden können von verschiedenen Objekten geteilt werden.

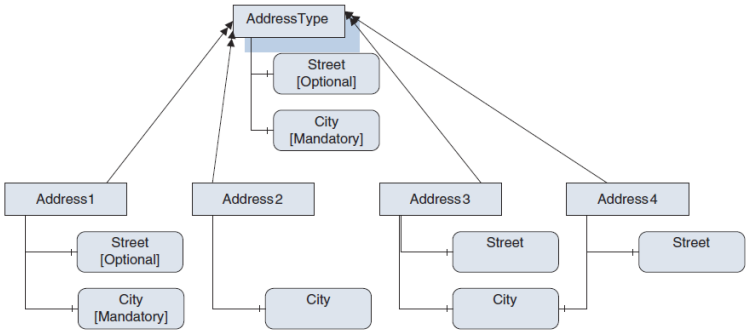

Abbildung 6: ein Anwendungsbeispiel; Quelle: "OPC Unified Architecture", p. 50

Eine besondere ModellingRule der OPC UA Spezifikation ist ExposesItsArray und kann für Arrays eines VariableTypes verwendet werden.

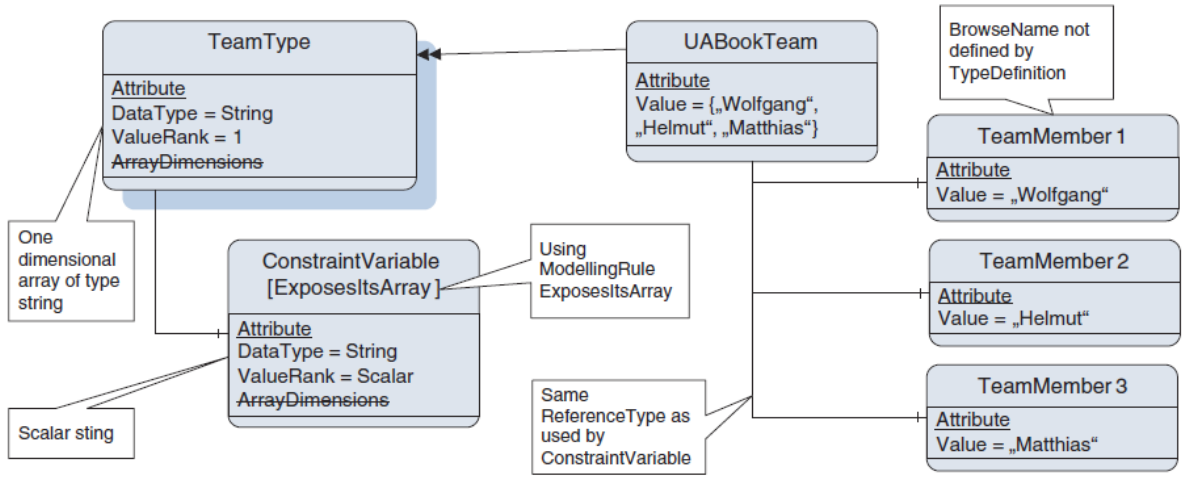

Abbildung 7: ExposesItsArray ModellingRule; Quelle: "OPC Unified Architecture", p. 53

Proxy Objects

Referenzen sind simple Konstruktionsn in OPC UA und verbinden nur zwei Nodes miteinander. Die einzige Information die zur Verfügung steht ist der ReferenceType und die Richtung der Referenz. Möchte man zusätzliche Informationen in Referenzen oder ModellingRules so kann dies nicht direkt durchgeführt werden. Man benötigt sogenannte Proxy Objects. Anstatt einer Referenz mit zusätzlichen Informationen definiert man sich ein Objekt welche Referenzen auf Target und Source Nodes enthält.

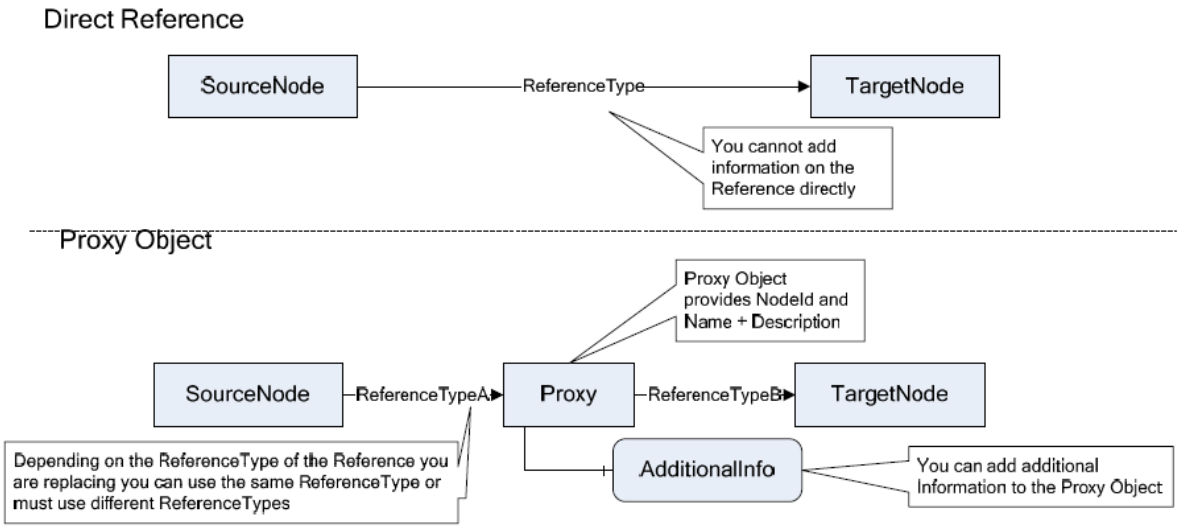

Abbildung 8: Prinzip von Proxy Objects; Quelle: "OPC Unified Architecture", p. 104

Views

Ein View wird benötigt um die Anzahl an sichtbaren Nodes und References in einem großen Address Space einzuschränken. Mit dem Konzept von Views können OPC UA Server ihren Address Space für besondere Tasks oder Use Cases anpassen.

Ein View wird repräsentiert durch einen Node im Address Space. Dieser Node ist sozusagen ein Entry Point für den Inhalt des Views. Alle Nodes die in einem View abgebildet werden, müssen durch den View Node referenziert werden oder können durch einen Parentnode mit ihm verbunden sein.

Die NodeId des View Nodes kann als Filter benutzt werden. Durch die Benutzung von Views kann ein Server Referenzen auf andere Nodes einschränken.

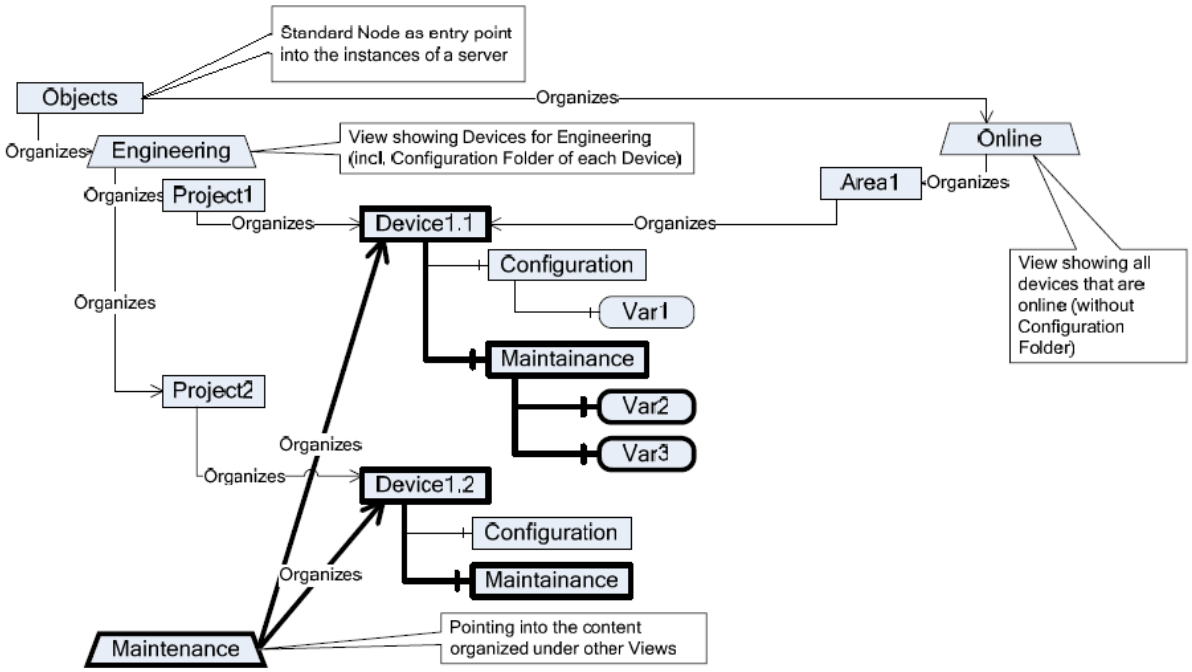

Abbildung 9: Views im Address Space; Quelle: "OPC Unified Architecture", p.73

Events

Nachdem *subscribing* beim EventNotifier werden Events über eine Notification angezeigt. OPC UA definiert eine Basishierarchie von EventTypes die erweitert werden kann. Damit Clients auf Events zugreifen können muss der Server die EventType Hierarchie in seinem Address Space verfügbar machen.

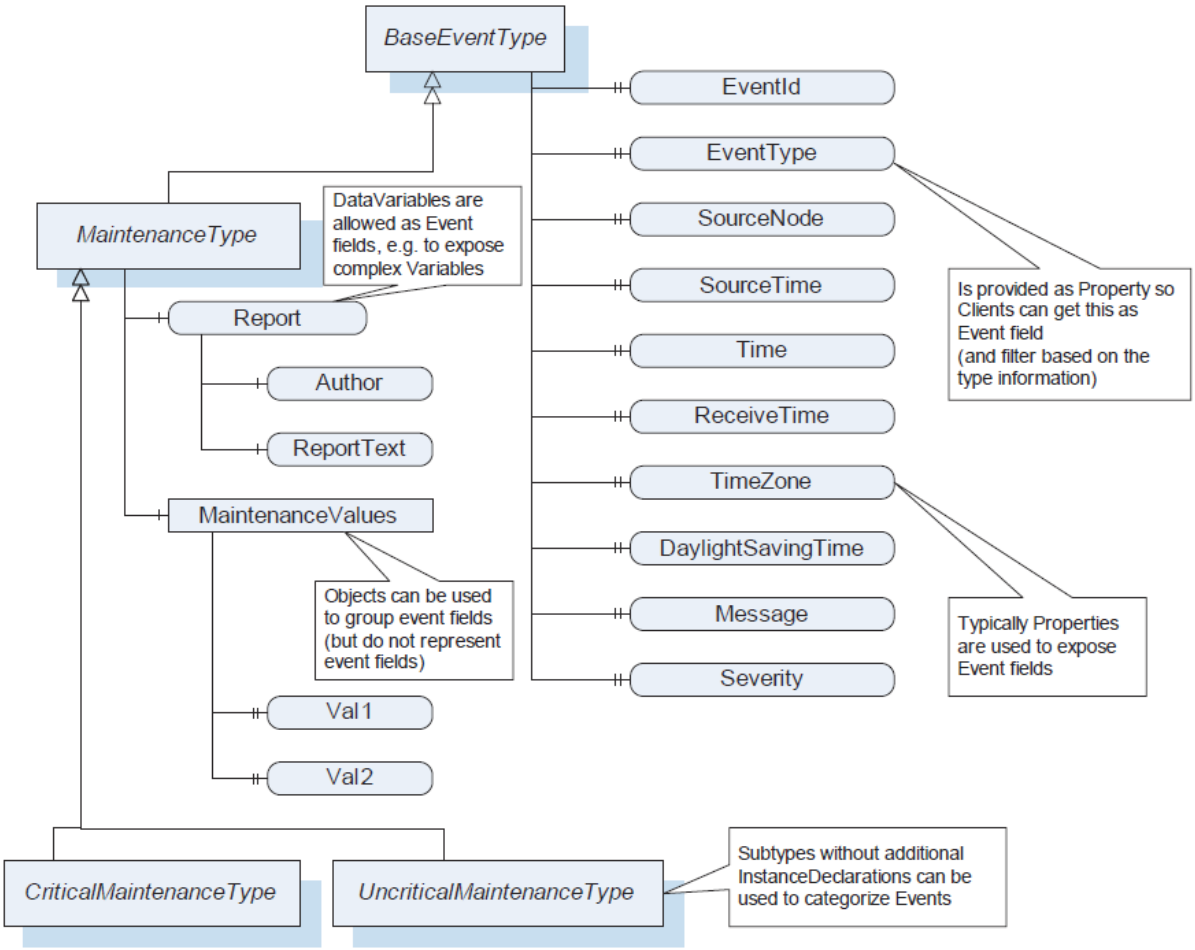

Abbildung 10: Die Event-Typ Hierarchie; Quelle: "OPC Unified Architecture", p.77

Address Space und Information Model

Die OPC UA Spezifikation definiert das *Base Information Model*, welches Grundtypen enthält. Ausgehend vom Base Information Model können weitere Information Models abgeleitet werden.

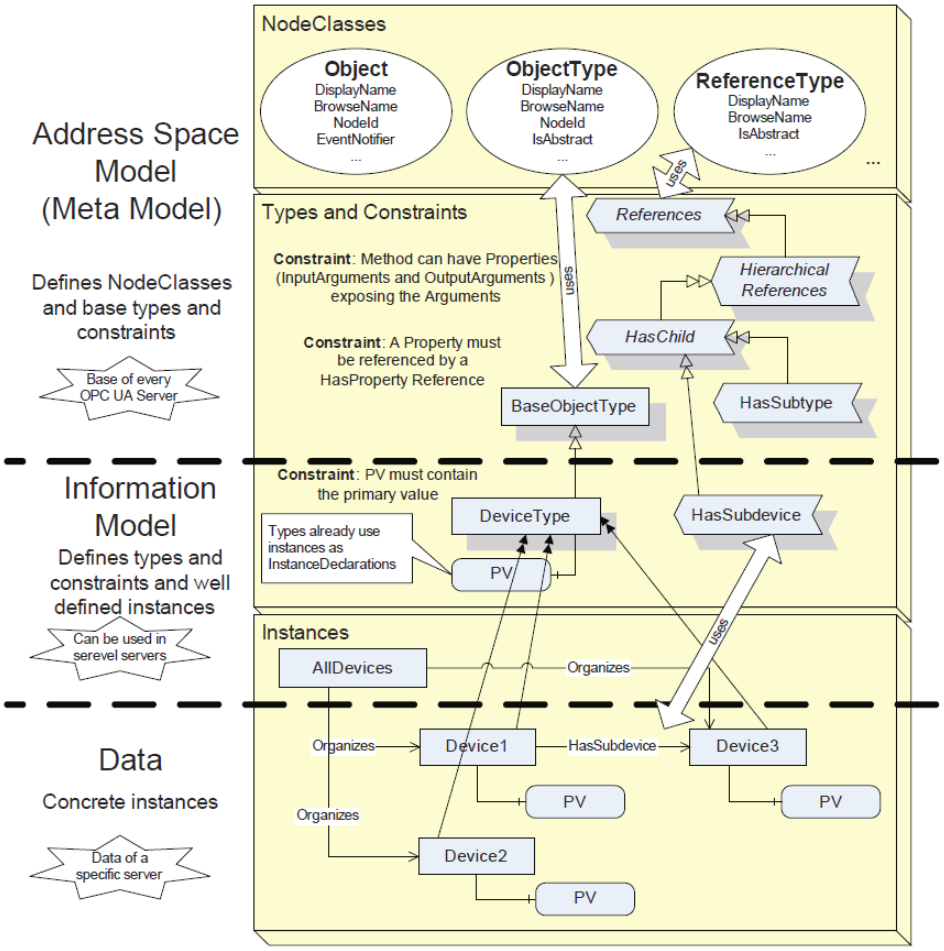

Abbildung 11: Address Space Model, Information Model und Daten; Quelle: "OPC Unified Architecture", p. 82

Literaturverzeichnis:

1. **Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm;** „OPC Unified Architecture“; ©2009 Springer Verlag Berlin Heidelberg; ISBN 978-3-540-68898-3
2. http://www.commsvr.com/UAModelDesigner/Index.aspx
3. http://www.opcfoundation.org/

Alle Grafiken wurden „OPC Unified Architecture“ entnommen; P 20 – P 125