

# Online-Kommunikation mittels OPC-UA vs. Engineering-Daten (offline) in AutomationML

## Eine Möglichkeit der Integration und Kombination

R. Henßen, Dr.-Ing. M. Schleipen, Fraunhofer IOSB, Karlsruhe

### Kurzfassung

OPC UA wird als Plattform-unabhängiger Kommunikations- und Datenaustauschstandard (IEC 62541) [1], [2] in Automatisierungssystemen verwendet. AutomationML (Automation Markup Language) dient als offenes und standardisiertes Datenaustauschformat (IEC 62714) [3], [4] zur Beschreibung von Anlagen und Anlagenkomponenten. Ziel dieser Veröffentlichung ist eine vereinfachte Erstellung von OPC UA Informationsmodellen basierend auf existierenden AutomationML-Modellen. Hierfür wurden Analogien zwischen beiden Standards und die Integration AutomationML-spezifischer OPC UA-Informationsmodell-Elemente genauer betrachtet.

### Abstract

OPC UA (OPC Unified Architecture) is a platform-independent communication and data exchange standard (IEC 62541) [1], [2] to be used in automation systems. AutomationML (Automation Markup Language) serves as open standard data exchange format (IEC 62714) [3], [4] for describing production plants or plant components. The goal of this contribution is to simplify the creation of OPC UA information models based on existing AutomationML data by examining the analogies between AutomationML and the OPC UA information model and the integration of AutomationML specific OPC UA information model elements.

### 1. Einleitung

Die Informationen im Engineering sind essentiell für dessen Effizienz. [5] Sie sind Dreh- und Angelpunkt für die Interoperabilität zwischen verschiedenen (IT-)Systemen. Interoperabilität wird vom IEEE Glossar [6] definiert als die Fähigkeit von zwei oder mehr Systemen oder Komponenten, Informationen auszutauschen und die ausgetauschte Information nutzen zu können. Wichtig ist dabei nicht nur, dass die Daten ausgetauscht werden, sondern auch, dass alle Kommunikationspartner die ausgetauschten Daten verstehen. Hierfür muss die Bedeutung der Daten, also die Semantik klar sein, damit die Informationen genutzt werden können. Standards sind hierfür der Schlüssel.

Für die Kommunikation und Verarbeitung der auszutauschenden Informationen bietet sich als existierender Standard die OPC UA (IEC 62541) an, die auf allen Ebenen der Automatisierung zum Einsatz kommen kann (siehe auch [7]). Sie ermöglicht eine plattform- und protokollunabhängige Kommunikation basierend auf Webservices, unterstützt aber auch Sicherheit, Zuverlässigkeit und Redundanz. Das integrierte vollvernetzte Informationsmodell bietet die Möglichkeit zur Integration standardisierter aber auch herstellerspezifischer Informationen. Informationsmodelle für OPC UA Server werden in der Regel manuell oder mit Hilfe firmenspezifischer Automatismen erzeugt.

Für die Beschreibung der kommunizierten Inhalte kann das XML-basierte Datenaustauschformat AutomationML (IEC CDV 62714-1) genutzt werden, das Produktionsanlagen und ihre Komponenten in verschiedenen Aspekten (Anlagenhierarchie, Geometrie, Kinematik, Ablaufplanung und Verhalten) einheitlich beschreibt. Engineering-Formate wie AutomationML können durch die Online-Kommunikation mittels OPC UA und ihre Informationsmodelle operationalisiert werden.

Der vorliegende Beitrag versucht das Engineering-Format AutomationML fit für den Betrieb zu machen, gleichzeitig aber auch den Anwendungsbereich von OPC UA zu vergrößern, indem Analogien zwischen AutomationML und den Informationsmodellen der OPC UA genutzt werden. Hierfür müssen die Basistypen des OPC UA Informationsmodells durch AutomationML-spezifische Typen erweitert und entsprechende Abbildungsvorschriften erstellt werden. Kapitel 2 beschreibt die OPC UA. Kapitel 3 erläutert die Grundlagen von AutomationML. In Kapitel 4 werden dann die Analogien zwischen beiden Standards und nötige neue OPC UA Typdefinitionen an Hand eines Beispiels vorgestellt. Kapitel 5 fasst den Beitrag zusammen und gibt einen kurzen Ausblick auf zukünftige Arbeiten.

## **2. OPC-UA für die Online-Kommunikation**

OPC UA ermöglicht den Produktionsdatenaustausch zwischen Steuerungen oder IT-Systemen unterschiedlicher Hersteller. OPC UA Server beinhalten ein Informationsmodell, das es Nutzern erlaubt, ihre Daten und deren Semantik in strukturierter Art und Weise zu definieren. Der Adressraum eines OPC UA Servers basiert auf einem Informationsmodell, das als vollvernetzter Graph angesehen werden kann, mit Knoten, deren Eigenschaften und Verbindungen zwischen diesen Knoten. Normalerweise werden Informationsmodelle für OPC UA Server manuell während der Implementierung erstellt [8] oder basieren auf Herstellerspezifischen Automatismen. Ein Server-Adressraum beinhaltet die folgenden Typen von Elementen:

- Object (Objekt): Ein Knoten, der ein physisches oder abstraktes Element eines Systems repräsentiert. Objekte werden mit Hilfe des OPC UA Object modelliert. Systeme und Geräte sind Beispiele für Objekte. Objekte können als Instanzen eines Objekttyps definiert werden. [9]
- ObjectType (Objekttyp): Ein Knoten, der eine Typdefinition für ein Objekt repräsentiert. [9]
- Variable (Variable): Eine Variable ist ein Knoten, der einen Wert enthält. [9]
- VariableType (Variablentyp): Ein Knoten, der eine Typdefinition für eine Variable repräsentiert. [10]
- DataType (Datentyp): Eine Instanz eines Datentyp-Knotens, die gemeinsam mit einem Wertebereich-Attribut (ValueRank) genutzt wird, um den Datentyp einer Variablen zu definieren. [10]
- ReferenceType (Referenztyp): Ein Knoten, der eine Typdefinition für eine Referenz repräsentiert. Der Referenztyp legt die Semantik der Referenz fest. Der Name eines Referenztyps legt die Verbindung zwischen Start- und Zielknoten fest, z.B. ‚A enthält B‘. [9]
- Method (Methode): Eine aufrufbare Softwarefunktion, die zu einem Objekt gehört. [9]
- View (Sicht): Eine definierte Untermenge eines Adressraums, die für einen Client von Bedeutung ist. [9]

Im Folgenden werden einige vordefinierte Referenztypen genutzt, die nun erläutert werden:

- HasComponent: Dient zur Abbildung von Teil-von-Beziehungen. Der Zielknoten der Referenz vom Typ HasComponent ist Teil des Startknotens. Dieser Referenztyp wird genutzt, um Objekte oder Objekttypen mit ihren untergeordneten/enthaltenen Objekten, Datenvariablen, Methoden und Variablen oder Variablentypen mit ihren Datenvariablen zu verbinden. [10]
- HasProperty: Dient zur Identifikation der Eigenschaften eines Knotens. [10]
- HasTypeDefinition: Dient zur Verbindung von Objekten oder Variablen mit ihren Objekttypen oder Variablentypen. [10]
- HasSubType: Dient zum Aufbau von Referenztyp-Hierarchien, um auszudrücken, dass Typen einen untergeordneten Typ besitzen. [10]

Bild 1 zeigt die grafische Notation der OPCFoundation zur Modellierung von Elementen in Adressräumen (siehe [9]). Bild 2 zeigt die grafische Notation der OPCFoundation zur Modellierung von Referenzen in Adressräumen (siehe [9]). Es existiert ebenfalls ein XML-Format in

Form eines XML-Schemas, um Adressräume zu beschreiben, das die OPCFoundation definiert hat.

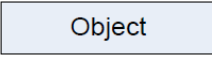
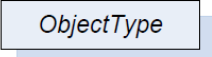
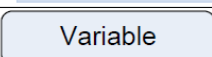
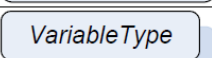
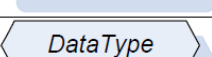
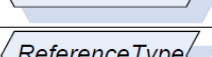
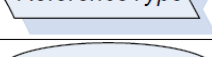
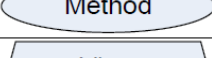
NodeClass	Graphical Representation
Object	
ObjectType	
Variable	
VariableType	
DataType	
ReferenceType	
Method	
View	

Bild 1: Elementtypen in OPC UA Adressräumen [9]





ReferenceType	Graphical Representation
HasComponent	
HasProperty	
HasTypeDefinition	
HasSubtype	

Bild 2: Referenztypen in OPC UA Adressräumen [9]

Die OPCFoundation definiert Basistypen, von denen neue Objekte und Typen abgeleitet werden können. Verschiedene andere Organisationen definieren so genannte Companion Spezifikationen, z.B. ‚OPC UA For Devices‘ (DI) und ‚OPC UA For Analyzer Devices‘ (ADI). DI dient zur Beschreibung von funktionell gegliederten Geräte-Parametern und –Methoden. ADI beschreibt spezielle Analysegeräte, wie z.B. Spektrometer. [11]

Eine solche Companion Spezifikation für AutomationML existiert aktuell noch nicht, ist aber in Arbeit und wird von einer gemeinsamen Arbeitsgruppe der OPCFoundation und des AutomationML e.V. unter Leitung des Fraunhofer IOSB (Fr. Schleipen) erarbeitet. Die hier vorgestellten Arbeiten werden dort als Vorschlag eingebracht.

Die OPC UA ist wegen ihrer zahlreichen Schnittstellen gerade für IT Systeme innerhalb der Produktion wie z.B. MES (Manufacturing Execution Systems) als Daten- und Integrationsplattform interessant und wichtig.

### 3. AutomationML für (offline) Engineering-Daten

AutomationML wurde speziell für den Austausch von Engineering-Informationen für Produktionsanlagen entwickelt. AutomationML definiert kein eigenes neues Datenformat, sondern nutzt verschiedene existierende Standards für unterschiedliche Aspekte und definiert Regeln zur Verwendung und Kombination von diesen. CAEX (IEC 62424) [12] wird als Dachdatenformat in AutomationML genutzt. Ein AutomationML Modell besteht also aus mindestens einem CAEX Modell. Andere genutzte Formate sind Collada [13] für Geometrie und Kinematik, sowie PLCOpenXML [14] für das Verhalten. Der vorliegende Beitrag fokussiert auf die CAEX-Modelle von AutomationML. CAEX beinhaltet objekt-orientiert Konzepte wie Klassen und Instanzen, die in Baumstrukturen (Hierarchien) abgelegt werden, ebenso wie die Möglichkeit durch Relationen beliebige Netze zu beschreiben.

#### 3.1 Struktur von AutomationML

Im Folgenden soll die Hauptstruktur von AutomationML-Modellen und deren Elemente beschrieben werden. Diese werden in drei Bildern dargestellt.

Jede AutomationML-Datei (eine XML-Datei) kann verschiedene Bibliotheken enthalten: Die ‚InterfaceClassLibs‘, um Schnittstellen zu definieren. ‚RoleClassLibs‘, um semantische Rollendefinitionen festzulegen und ‚SystemUnitClassLibs‘, um wiederverwendbare AutomationML-Objekte zu beschreiben. Die vierte Hierarchie in einer AutomationML-Datei ist die ‚Instance-Hierarchy‘, die die Beschreibung der tatsächlichen Anlage beinhaltet. Bild 3 bildet diese Basisstruktur einer AutomationML-Datei ab und zeigt weiterhin die Hauptobjekte in jeder der vier Hierarchien. Die Verbindungspfeile im Bild repräsentieren eine ‚Besteht-aus‘-Relation.

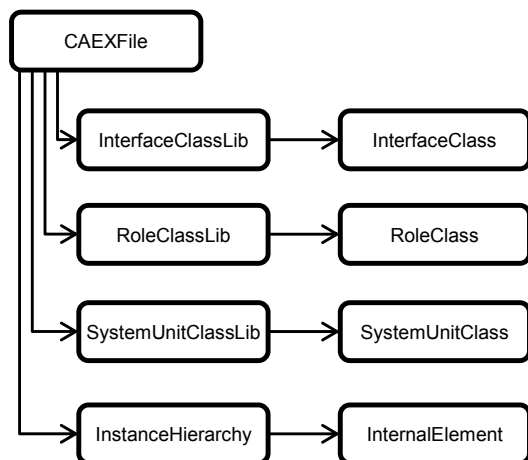


Bild 3: AutomationML Hauptstruktur

Hauptelemente von AutomationML und deren mögliche Relationen untereinander sind in Bild 4 dargestellt. Die hierarchische Baumstruktur setzt voraus, dass jeder Elementtyp Unterelemente desselben Typs (gekennzeichnet als ChildElement-Relation im Bild) besitzen kann. ‚SystemUnitClasses‘ und ‚RoleClasses‘ sind in einer Vererbungsstruktur definiert. Relationen vom Typ RefBaseClassPath definieren solche Vererbungssachverhalte. ‚RoleClasses‘ können ‚SystemUnitClasses‘ oder ‚InternalElements‘ zugeordnet werden. Im letzten Fall kann dies auf zwei Arten geschehen: per ‚RoleRequirement‘ oder ‚SupportedRoleClass‘. Die verbleibenden Verbindungen zwischen Elementen im Bild 4 beschreiben die Verbindungen zwischen ‚InternalElements‘ und ‚SystemUnitClasses‘. Ein ‚InternalElement‘ kann von einer ‚SystemUnitClass‘ abgeleitet sein.

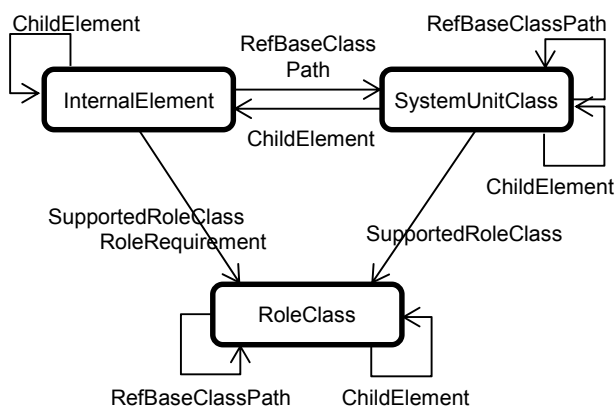


Bild 4: AutomationML Hauptelemente

Im Rahmen von AutomationML stellen ‚SystemUnitClasses‘ nur Vorlagen dar, die nach der Instanziierung der ‚SystemUnitClass‘ in einem ‚InternalElement‘ verändert werden können. Die Rückverbindung drückt aus, dass ‚SystemUnitClasses‘ Unterelemente vom Typ ‚InternalElement‘ enthalten können. Diese Unterelemente müssen im Fall einer Instanziierung der ‚SystemUnitClass‘ in einem ‚InternalElement‘ enthalten sein.

‚RoleClasses‘ beschreiben Funktionen eines physischen oder logischen Anlagenobjekts unabhängig von einer technischen Implementierung. Sie ermöglichen die abstrakte und Hersteller-unabhängige Objektspezifikation. Die Zuordnung einer ‚RoleClass‘ zu einem Objekt (‚SystemUnitClass‘ oder ‚InternalElement‘) repräsentiert die Spezifikation grundlegender Funktionalität oder Anforderungen.

Bild 5 zeigt die Basisstruktur einer ‚RoleClass‘, ‚SystemUnitClass‘ oder eines ‚InternalElement‘. Jeder dieser Elementtypen kann beliebig verschachtelte Attribute (‚Attribute‘) oder Schnittstellen (‚Interface‘) enthalten (siehe Relation vom Typ ChildElement). Schnittstellen sind so genannte ‚ExternalInterfaces‘ und sollten von einer ‚InterfaceClass‘ abgeleitet sein. Sie

können mit anderen ‚ExternalInterfaces‘ durch ‚InternalLinks‘ verbunden sein. ‚InterfaceClasses‘ sind hierarchisch beschrieben und sollten unabhängige Vererbungsrelationen beinhalten. ‚InterfaceClasses‘ und ‚ExternalInterfaces‘ können ebenfalls beliebig verschachtelte Attribute enthalten.

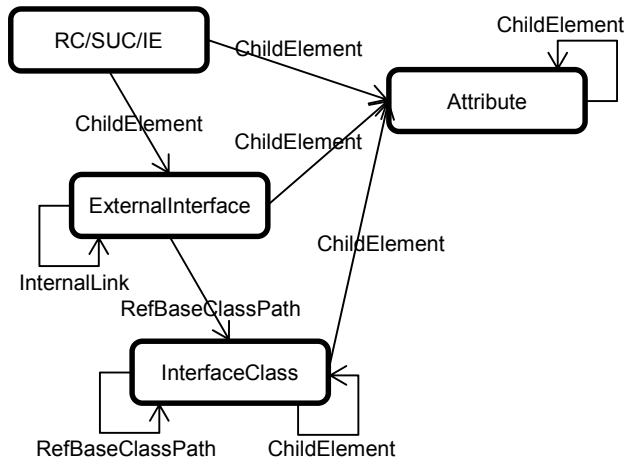


Bild 5: AutomationML main object details

Weiterhin definiert AutomationML verschiedene erweiterte Konzepte innerhalb der Spezifikation Teil 1, sowie eine Grundmenge an semantischen Rollenbeschreibungen in der Spezifikation Teil 2. Das Prozess-Produkt-Ressourcen-Konzept erlaubt eine Strukturierung von Engineering-Daten auf oberer Ebene und erlaubt Prozess-, Produkt- oder Ressourcen-spezifische Sichten, ebenso wie die Verbindungen zwischen diesen drei Elementen.

Das AutomationML Port-Konzept erlaubt eine Gruppierung von Schnittstellen (‚InterfaceClasses‘) zu einem komplexen Stecker. Das AutomationML Gruppen-Konzept ermöglicht die Speicherung von verschiedenen Sichten auf AutomationML-Objekt. Das AutomationML Facetten-Konzept erlaubt die Speicherung und Gruppierung von Attributen und Schnittstellen bei einem AutomationML-Objekt. Das PropertySet-Konzept wurde definiert, um Attributen Semantik zuordnen zu können, ähnlich dem Konzept der Rollen auf Objektebene.

### 3.2 AutomationML-Beispiel

Bild 6 zeigt ein AutomationML Beispiel und dessen Baumstruktur. Das Beispiel enthält eine ‚InstanceHierarchy‘ namens ‚TestProject‘, die eine Produktionslinie mit Robotern enthält.

Die ‚AutomationMLInterfaceClassLib‘ und die ‚AutomationMLRoleClassLib‘ sind verkürzte Versionen der Standardbibliotheken, da sie nur die verwendeten Elemente enthalten, um das Beispiel möglichst kurz zu halten. Die Projekt-spezifische ‚RoleClassLib‘ ‚TestRoleLib‘ definiert eine Rolle Roboter (‚Robot‘), die ein Attribut, sowie eine Kommunikationsschnittstelle enthält und von der Standard-Rolle ‚Resource‘ abgeleitet ist. Die Detailinformationen der Rolle sind in

der Ansicht in Bild 6 nicht sichtbar. Die zweite Rolle ‚SpecialRobot‘ ist von der ersten Rolle abgeleitet. Die ‚SystemUnitClassLib‘ ‚ABCSystemUnitClassLib‘ definiert Beispieltypen für einen Roboter (zugeordnet zur Rolle ‚Robot‘), sowie für eine Produktionslinie.

Das ‚TestProject‘ enthält nun also die Instanzen ‚MainLine‘ – die Produktionslinie – und den ersten Roboter ‚RobotI‘, die die Typen (die ‚SystemUnitClasses‘) 1:1 instanzieren. Für den zweiten Roboter ‚RobotII‘ existiert keine Objekt in der ‚SystemUnitClassLib‘. Seine speziellen Anforderungen werden durch die Zuordnung der Rolle ‚SpecialRobot‘ ausgedrückt.

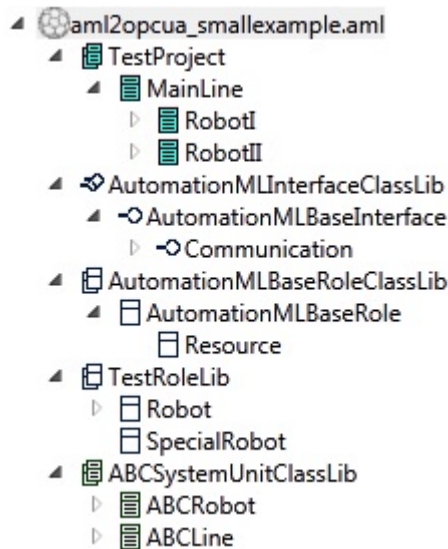


Bild 6: Objektbaum des AutomationML-Beispiels

## 4. Erzeugung eines OPC UA Informationsmodells auf Basis von AutomationML-Modellen

### 4.1 Analogien zwischen AutomationML und OPC UA Informationsmodellen

Nun folgt ein kurzer Vergleich zwischen OPC UA Informationsmodellen und AutomationML Modellen. Beide Standards unterstützen einen objekt-orientierten Ansatz und basieren auf einem Typ-Instanz-Konzept. Die Analogie zwischen OPC UA Objekten und InternalElements bei AutomationML ist offensichtlich. Aber bereits die Vererbungsmechanismen werden unterschiedlich genutzt. ‚SystemUnitClasses‘ sind bei AutomationML nur Vorlagen zur Instanziierung. Danach können die entstandenen ‚InternalElements‘ angepasst und Teile hinzugefügt oder entfernt werden.

OPC UA hingegen behandelt Instanziierungen mittels der ‚HasTypeDefinition‘ Relation explizit, die so genannten ‚ModellingRules‘ legen fest, ob Unterknoten nach der Instanziierung existieren müssen oder nicht. Im Fall einer Abbildung von AutomationML ‚InternalElements‘ auf OPC UA ‚Objects‘ müssten die ‚ModellingRules‘ aller Unterknoten auf den Wert ‚Optional‘ gesetzt werden. Weiterhin müsste es einen grundlegenden und leeren ‚SystemUnitClass‘ Typ in OPC

UA geben, um einen Typ für ‚InternalElements‘ ohne eine Referenz auf eine ‚SystemUnitClass‘ verfügbar zu machen.

Die Typisierung ist in beiden Standards nicht auf Objekte beschränkt, sondern wird auch für Schnittstellen und Relationen verwendet. Ebenso existiert in beiden Standards die Unterscheidung zwischen Objekten (Objects in OPC UA, InternalElements in AutomationML) und deren Eigenschaften (Properties in OPC UA, Attribute in AutomationML). Beide Standards ermöglichen die Beschreibung von beliebigen Netzen. OPC UA beschreibt Semantik innerhalb der ‚ObjectTypes‘ wohingegen AutomationML hierfür ‚RoleClasses‘ nutzt, um Objekten eine Semantik aufzuprägen, die nicht unbedingt der Typdefinition innerhalb einer ‚SystemUnitClass‘ entsprechen muss und somit mehr Freiheitsgrade zulässt.

#### **4.2 Erweiterung der Basistypen von OPC UA Informationsmodellen durch AutomationML-spezifische Typen**

Die Hauptstruktur einer AutomationML-Datei aus Bild 3 ist in Bild 7 in leicht veränderter Form abgebildet. ‚InstanceHierarchies‘ aus AutomationML können an beliebigen Positionen in einem OPC UA Informationsmodell abgelegt werden. Die Bibliotheken sollten in einem gemeinsamen Ordner abgelegt werden.

Die hierfür notwendigen AutomationML-spezifischen OPC UA Typen werden nun nachfolgend aufgeführt.

Neue OPC UA ‚ReferenceTypes‘:

- ReferenceType ‚HasAMLChild‘
  - Untertyp of ‚HasChild‘
  - Hierarchische Relation innerhalb der AutomationML Hierarchie (der AutomationML Baumstruktur).
  - Diese Relationen haben in AutomationML keine spezielle Bedeutung. Eventuell müssen sie daher nicht abgebildet werden.

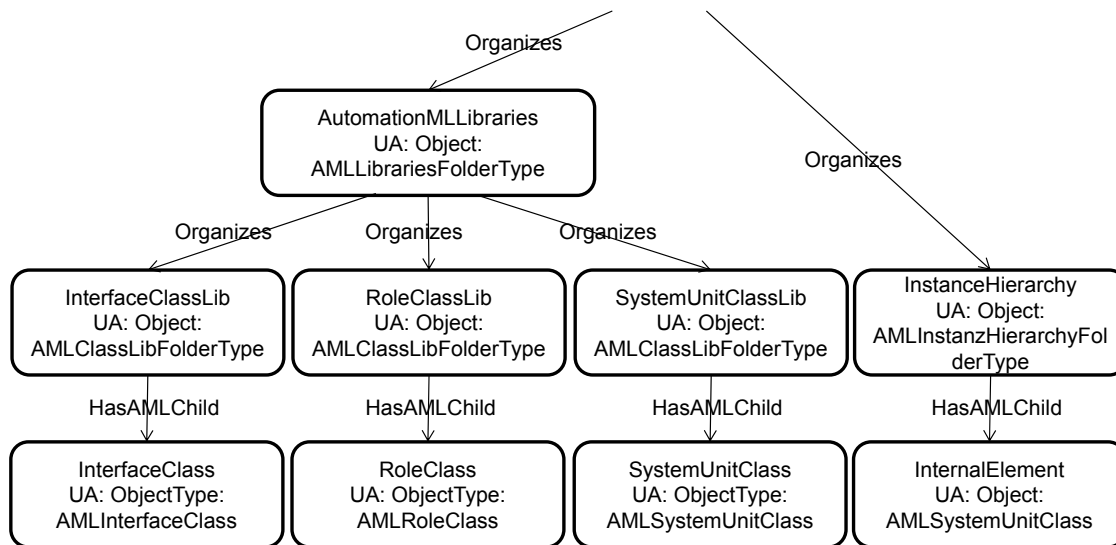


Bild 7: AutomationML Hauptstrukturen in einem OPC UA Informationsmodell.

Neue OPC UA ‚FolderTypes‘ (alle nachfolgenden ‚ObjectTypes‘ sind Untertypen vom Objekttyp ‚FolderType‘):

- ObjectType ‚AMLLibrariesFolderType‘
  - Ordner, um ausschließlich AutomationML Bibliotheken abzulegen. Da OPC UA Zielknoten mit dem gleichen ‚BrowseName‘ erlaubt, ist es möglich verschiedene Versionen von ein- und derselben Bibliothek gleichzeitig abzulegen. Dies ist in AutomationML-Modellen verboten. Gleichzeitig wird es dadurch aber möglich, verschiedene AutomationML-Dateien in einem OPC UA Informationsmodell abzubilden.
- ObjectType ‚AMLClassLibFolderType‘
  - Typ für die drei AutomationML Bibliotheken. Alle AutomationML Bibliotheken sollten auf Objekte dieses Typs abgebildet werden.
  - Objekte diesen Typs dürfen nur Referenzen des Typs ‚HasAMLChild‘ enthalten und auf Zielknoten des Typs ‚CAEXObjectType‘ verweisen.
- ObjectType ‚AMLInstanzHierarchyFolderType‘
  - Typ für die AutomationML ‚InstanceHierarchies‘
  - Objekte diesen Typs dürfen nur Referenzen des Typs ‚HasAMLChild‘ enthalten und auf Zielknoten des Typs ‚AMLSystemUnitClass‘ verweisen.

Weitere neue OPC UA ‚ObjectTypes‘:

- ObjectType ‚CAEXObjectType‘
  - Untertyp von ‚BaseObjectType‘
  - Abstrakter Objekttyp als Basis für alle AutomationML-Klassen

- ObjectTypes 'AMLInterfaceClass', 'AMLRoleClass' und 'AMLSystemUnitClass'
  - Untertypen von 'CAEXObjectType'
  - Dies sind die grundlegenden Objekttypen für die AutomationML-Klassen.

Bild 8 zeigt die Abbildung der Relation zwischen den Hauptelementen von AutomationML aus Bild 4. Die Struktur des Bilds 4 wurde dabei nicht verändert, um die Abbildung direkt an Hand der Grafik nachvollziehen zu können.

Ein neuer OPC UA Typ wird verwendet:

- ReferenceType 'HasAMLRoleReference'
  - Untertyp von 'NonHierarchicalReferences'
  - Der Startknoten sollte ein 'Object' sein und der Zielknoten ein Untertyp von 'AMLRoleClass'.

Um die Vererbungsmöglichkeiten der Rollen in AutomationML ausreichend abzubilden, müssen AutomationML 'RoleClasses' auf OPC UA 'ObjectTypes' abgebildet werden. 'RoleClasses' können nicht mit Hilfe von Referenzen vom Typ 'HasTypeDefinition' zugeordnet werden, da OPC UA keine mehrfache Vererbung unterstützt. Die beiden AutomationML Mechanismen 'SupportedRoleClass' und 'RoleRequirement' werden auf gleiche Art und Weise behandelt.

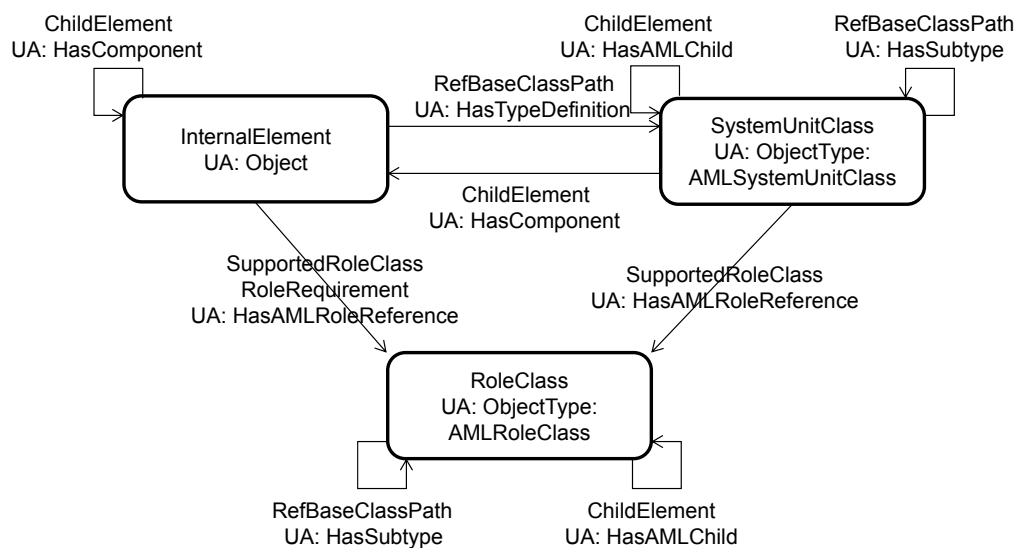


Bild 8: AutomationML Elemente in OPC UA.

Relationen vom Typ ChildElement innerhalb der Klassenstrukturen werden auf den neuen OPC UA Relationstyp 'HasAMLChild' abgebildet. Kindelemente vom Typ 'InternalElement' werden mittels der 'HasComponent' Relation referenziert. Die Vererbungsrelationen innerhalb von 'SystemUnitClasses' und 'RoleClasses' werden mittel der OPC UA relation 'HasSubtype' abgebildet.

Abbildungen der AutomationML Objekt-Details aus Bild 5 sind in Bild 9 dargestellt. Die Analogie zwischen den beiden Bildern ist offensichtlich.

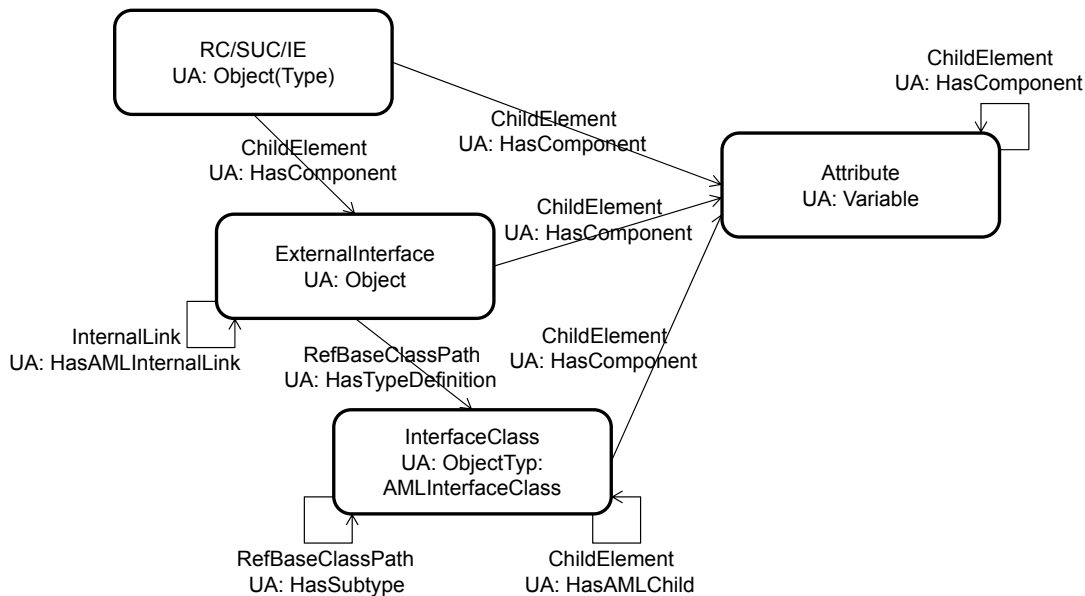


Bild 9: AutomationML Objektdetails in OPC UA.

Es wird ein neuer OPC UA Typ verwendet:

- ReferenceType 'HasAMLInternalLink'
  - Untertyp von 'NonHierarchicalReferences'
  - Symmetrische Referenz
  - Start- und Zielknoten sollten ein 'ExternalInterface' sein, das nicht innerhalb der einer 'RoleClass' abgelegt sein darf.

'Attribute' und 'ExternalInterfaces' werden mittels Relationen vom Typ 'HasComponent' und nicht mit Hilfe von Relationen vom Typ 'HasAMLChild' abgebildet. Relationen vom Typ 'HasAMLChild' werden nur innerhalb der Klassen-/Typstrukturen verwendet.

'InterfaceClasses' sind die Typdefinitionen von 'ExternalInterfaces', daher wird diese Beziehung mit Hilfe einer 'HasTypeDefinition' Relation abgebildet. Ebenso wie bei den 'RoleClasses' und 'SystemUnitClasses' werden Relationen zwischen 'InterfaceClasses' als 'HasAMLChild' Relation im Falle der XML-Baumstruktur abgebildet und als 'HasSubtype' relation im Falle einer Vererbungsrelation.

### 4.3 Transformiertes AutomationML-Beispiel in OPC UA

Das in Bild 6 gezeigte AutomationML Beispiel wird nun in OPC UA abgebildet. Die nachfolgenden Bilder sind das Ergebnis einer manuellen Abbildung zwischen AutomationML und OPC UA. Die grafische OPC UA Notation aus Bild 1 und Bild 2 wird hierfür genutzt. B

Bild 10 zeigt die Klassenstruktur des Beispiels. Auf der rechten Seite des Bilds ist die Vererbungsstruktur der Rollen zu sehen. Die Rolle ‚SpecialRobot‘ lässt sich über die Rollen ‚Robot‘ und ‚Resource‘ zurückführen auf die ‚AutomationMLBaseRole‘ und darüber auf die Basistypen ‚CAEXObjectType‘ (AutomationML) und den ‚BaseObjectType‘ von OPC UA. Die Rolle ‚Robot‘ beinhaltet zwei Komponenten, eine Schnittstelle ‚CommunicationInterface‘, sowie ein Attribut ‚Axes‘. Auf der linken Seite des Bilds 10 werden die ‚SystemUnitClasses‘ dargestellt. Die Klasse ‚ABCRobot‘ mit Referenz zur Rolle ‚Robot‘ beinhaltet die gleichen Komponenten wie die Rolle. Ebenso gibt es die ‚SystemUnitClass‘ ‚ABCLine‘ und den gemeinsamen Elternknoten ‚AutomationMLBaseSystemUnit‘, der als Vaterknoten aller ‚SystemUnitClasses‘ im OPC UA Modell fungiert.

Weiterhin ist die Schnittstellen-Klasse ‚Communication‘ zu sehen, die von der ‚AutomationMLBaseInterface‘ abgeleitet ist. Jeder Schnittstelle (siehe ‚Robot‘ und ‚ABCRobot‘) wird diese ‚InterfaceClass‘ als Typdefinition zugeordnet.

Die Abbildung der ‚InstanceHierarchy‘ ist in Bild 11 dargestellt. Das Objekt ‚AMLLibraries‘ vom Typ ‚AMLLibrariesFolderType‘ ist der Ordner aller AutomationML-Bibliotheken. Daher enthält er ebenfalls die ‚InstanceHierarchy‘ des Beispiels namens ‚TestProject‘. Diese ‚InstanceHierarchy‘ ist oben links dargestellt und enthält die ‚MainLine‘, sowie die zwei Roboter. Jede Instanz hat eine eigene Typreferenz zur entsprechenden ‚SystemUnitClass‘. ‚RobotII‘ hat zusätzlich dazu eine Referenz auf die Rolle ‚SpecialRobot‘. Alle Bibliotheken und ihre Klassen sind nochmals dargestellt, allerdings ohne die sie definierenden Komponenten.

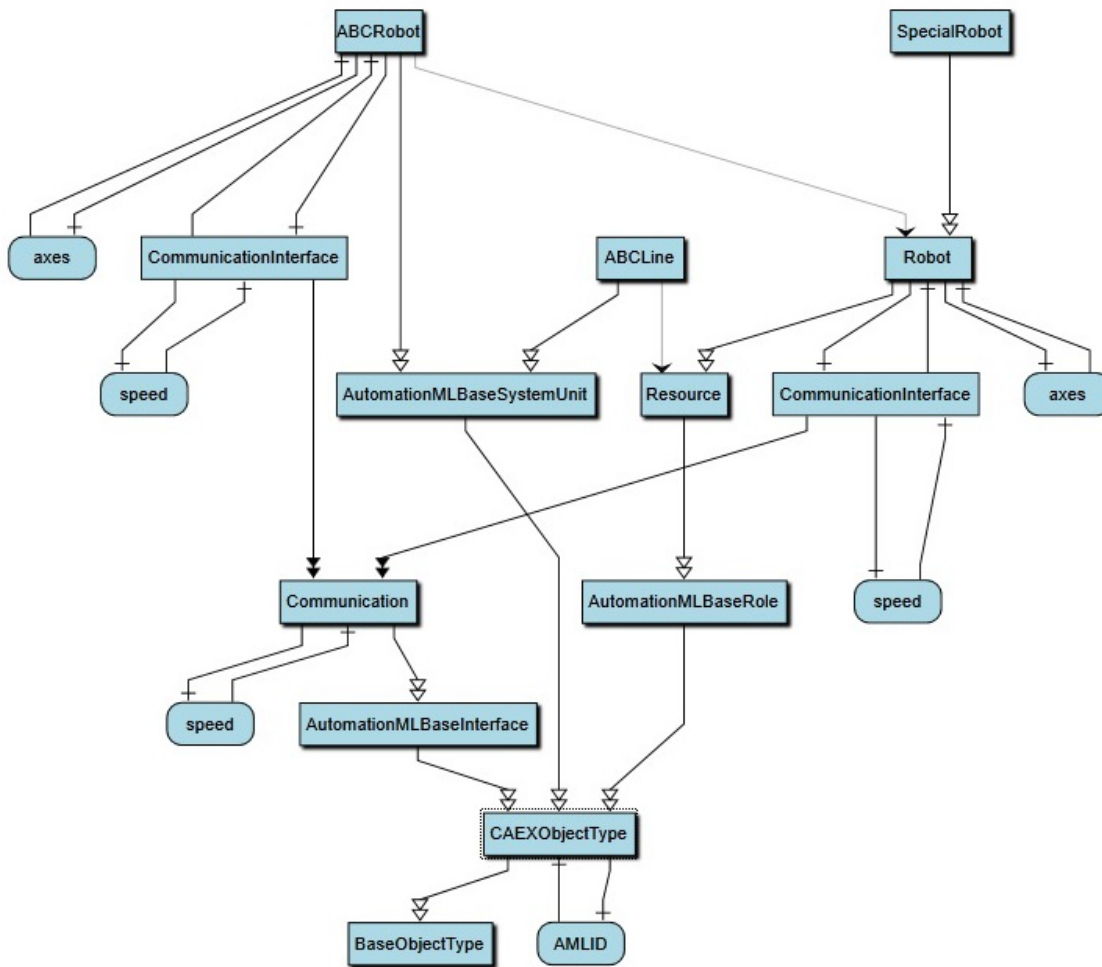


Bild 10: Klassenbaum des Beispiels in OPC UA.

## 5. Zusammenfassung und Ausblick

Ziel des vorliegenden Beitrags ist es aufzuzeigen, wie die Standards OPC UA und AutomationML kombiniert werden können, mit dem Ziel die einfache Erstellung von OPC UA Informationsmodellen zu unterstützen. Datenaustauschstandards wie AutomationML können mit Hilfe eines spezifischen OPC UA-Informationsmodells operationalisiert werden.

Hierbei können verschiedene Analogien zwischen AutomationML und OPC UA-Informationsmodellen, wie z.B. die hierarchische Struktur oder die Typisierung von Schnittstellen und Relationen, ausgenutzt werden. Darüber hinaus können die Basistypen der OPC-UA-Informationsmodelle durch spezifische Erweiterungen (neue Objekte und Typen) für AutomationML ergänzt werden. Wenn die Abbildungsregeln wohldefiniert sind, kann die Umsetzung dieser automatisch erfolgen und viel Zeit und Aufwand bei der Erstellung von OPC-UA Informationsmodellen einsparen, sowie deren Qualität erhöhen.

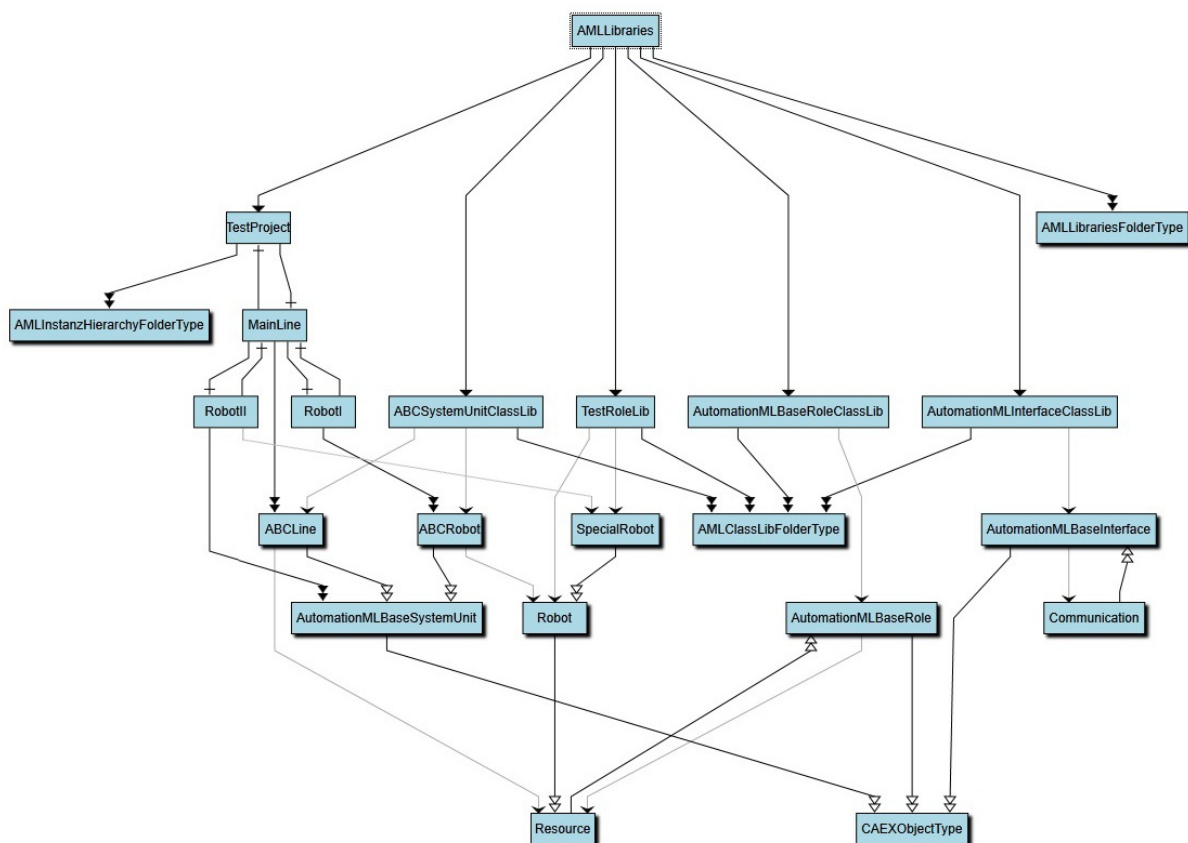


Bild 11: AutomationML InstanceHierarchy des Beispiels in OPC UA

Es existiert aktuell keine Companion-Spezifikation von AutomationML für OPC UA. Gemeinsamkeiten und notwendige Erweiterungen werden an Hand eines konkreten Beispiels im vorliegenden Beitrag aufgezeigt. Eine gemeinsame Arbeitsgruppe der OPC Foundation und des AutomationML e.V. befasst sich aktuell mit diesem Thema unter Leitung des Fraunhofer IOSB. Der vorliegende Beitrag wird in diese Arbeitsgruppe als Vorschlag eingebracht.

Ein OPC UA Server, der Anlagen-Beschreibungen im Format AutomationML erhält, wird im Rahmen des EU-Projekts SkillPro (Skill-based Propagation of "Plug&Produce"-Devices in Reconfigurable Production Systems by AML) [15] umgesetzt und dient als Implementierung und Evaluierung der hier vorgestellten Konzepte. Vorarbeiten auf OPC-UA- und AutomationML-Basis wurden unter Anderem im Forschungsprojekt DigET (Digitaler Engineering-Tisch) gefördert durch die Landesstiftung Baden-Württemberg geleistet.

## 6. Literatur

- [1] IEC 62541 (Normenreihe):2010-2012, OPC Unified Architecture.
- [2] Mahnke, W., Leitner, S-H. Damm, M. (2009): OPC Unified Architecture, Springer.

- [3] IEC 62714 (Normenreihe), Engineering data exchange format for use in industrial systems engineering – Automation Markup Language AML (Whitepaper verfügbar unter <http://www.automationml.org/>).
- [4] Draht, R. (Hrsg.) (2010): Datenaustausch in der Anlagenplanung mit AutomationML, Springer.
- [5] Sauer, O., Jasperneite, J.: Adaptive information technology in manufacturing, Proceedings of the CIRP International Conference on Manufacturing Systems, Madison/Wisconsin, 01.-03.06.2011.
- [6] IEEE Glossary: Interoperability, [http://www.ieee.org/education\\_careers/education/standards/standards\\_glossary.html](http://www.ieee.org/education_careers/education/standards/standards_glossary.html), Stand 18.04.2014.
- [7] Bratukhin, A.; Sauter, T.: Bridging the gap between centralized and distributed manufacturing execution planning, Proceedings of: IEEE ETFA, 13.-16. September 2010, Bilbao, Spanien, 2010.
- [8] Schleipen, M., Sauer, O., Wang, J.: Semantic integration by means of a graphical OPC Unified Architecture (OPC UA) information model designer for Manufacturing Execution Systems. In: Sihn/Kuhlang (Ed.), Sustainable Production and Logistics in Global Networks. 43rd CIRP International Conference on Manufacturing Systems, 26-28 May 2010, Vienna: Neuer Wissenschaftlicher Verlag, S. 633-640, 2010.
- [9] IEC 62541-1:2010 OPC Unified Architecture Part 1: Overview and Concepts
- [10] IEC 62541-3:2010 OPC Unified Architecture Part 3: Address Space Model
- [11] OPC Foundation, 21. August 2013  
([http://opcfoundation.org/Default.aspx/02\\_news/02\\_news\\_display.asp?id=1064&MID=News](http://opcfoundation.org/Default.aspx/02_news/02_news_display.asp?id=1064&MID=News))
- [12] IEC 62424:2008, Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools.
- [13] COLLADA 1.4.1: March 2008 COLLADA – Digital Asset Schema Release 1.4.1  
([http://www.khronos.org/files/collada\\_spec\\_1\\_4.pdf](http://www.khronos.org/files/collada_spec_1_4.pdf)).
- [14] PLCopen XML 2.0:December 3rd 2008 and PLCopen XML 2.0.1: May 8th 2009, XML formats for IEC 61131-3 (<http://www.plcopen.org>)
- [15] SkillPro - Skill-based Propagation of "Plug&Produce"-Devices in Reconfigurable Production Systems by AML, EU FP7, [www.skilpro-project.eu](http://www.skilpro-project.eu), Stand 18.04.2014.
- [16] DigET – Digitaler Engineering-Tisch, Landesstiftung Baden-Württemberg, [www.dig-et.de](http://www.dig-et.de), Stand 18.04.2014.