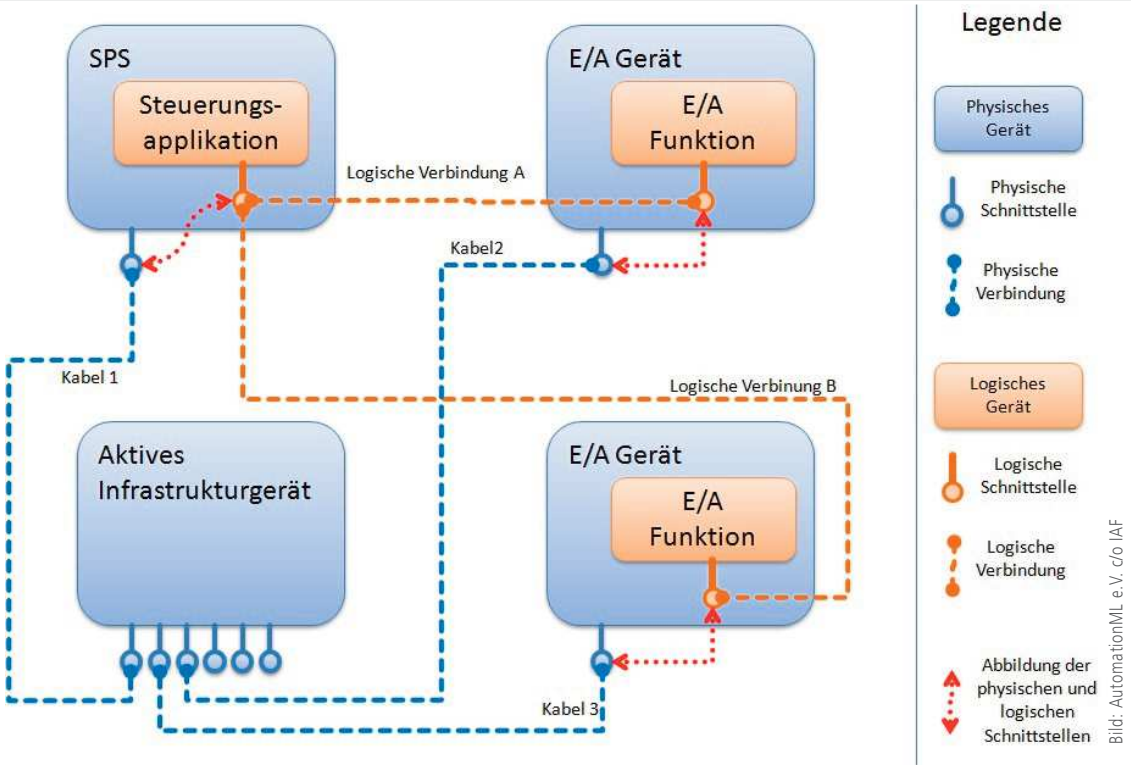

Bild 1: Kommunikationsnetzwerk mit logischen und physischen Geräten und Verbindungen

AutomationML – Kommunikation

Serie AutomationML Teil 7: Modellierung und Austausch von Entwurfsdaten für Kommunikationssysteme

Automatisierung erfordert die Kommunikation zwischen Geräten. Die Planung solcher Kommunikationssysteme erfolgt meist in mehreren Phasen, in denen Planungsergebnisse aus früher gelegenen Phasen importiert und neue Daten für den Kommunikationssystementwurf erzeugt werden. Die nahtlose Weitergabe derartiger Planungsstände von Werkzeug zu Werkzeug ist jedoch bis heute problematisch, da kein passendes Datenaustauschformat für eine konsistente und verlustfreie Informationsweitergabe zwischen Entwurfswerkzeugen existiert. Dieser Beitrag schlägt eine im Rahmen des AutomationML e.V. erarbeitete Methodik vor, mit der Kommunikationssysteme mit AutomationML modelliert und ausgetauscht werden können.

Auf den ersten Blick erscheint ein Kommunikationssystem als ein einfaches Netzwerk physikalischer Geräte. Bei näherer Betrachtung wird jedoch deutlich, dass Geräte auch dann miteinander kommunizieren können, wenn sie physikalisch nicht direkt miteinander verbunden sind. Darüber hinaus kann ein Kabel oder Gerät durchaus mehrere Protokolle verarbeiten. Diese Überlegungen führen zur Unterscheidung zwischen physischen und logischen Geräten und Verbindungen (Bild 1). Eine logische Topologie betrachtet die Steuerungsapplikation aus Sicht der zwischen Applikationsteilen ausgetauschten Variablen und ignoriert die tatsächlichen physikalischen Verbindungen. Hierbei können die Anforderungen an den Datenaustausch hinsichtlich der Kommunikationseigenschaften sowie Eigenschaften der Steuerungsapplikationsteile wie Abarbeitungszeiten oder Eigenschaften der logischen Schnittstellen wie eine Portnummer modelliert werden. Die physikalische Topologie bildet die realen Kommunikationsgeräte und die zwischen ihnen aufgebauten Kommunikationsverbindungen mit Datenpaketaustausch ab. Hier sind neben den eigentlichen kommunizierenden Steuerungsgeräten auch aktive und passive Infrastrukturkomponenten, ihre Eigen-

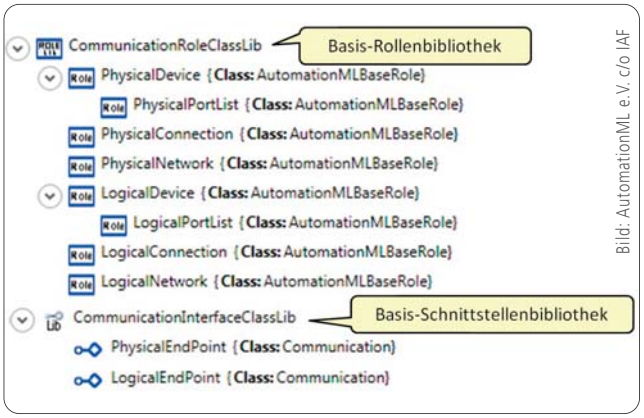

Bild 2: Rollen- und Interface-Klassen für die Kommunikationsmodellierung

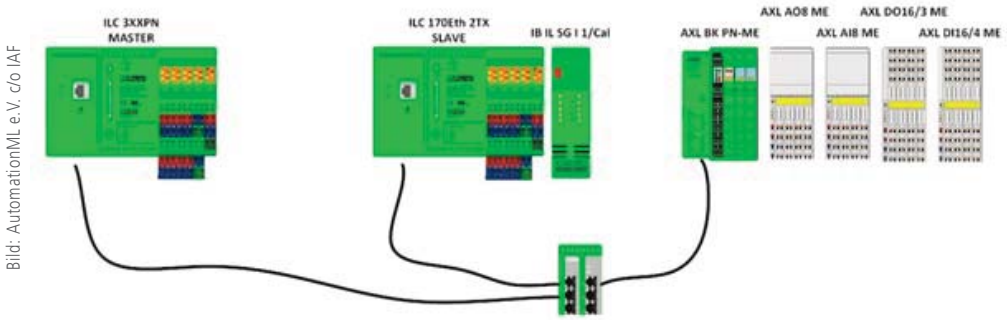

Bild: AutomationML e.V. c/o IAF

Bild 3: Kommunikationssystembeispiel

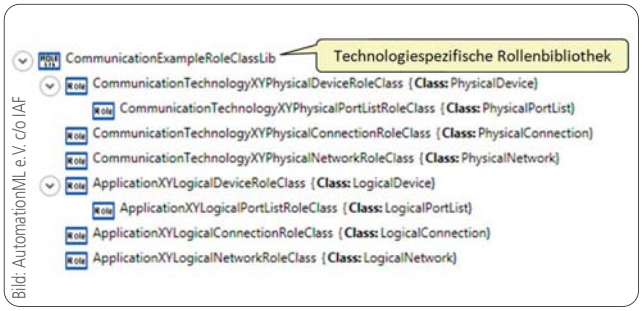

Bild: AutomationML e.V. c/o IAF

Bild 4: Technologieabhängige Rollenklassen für das Anwendungsbeispiel

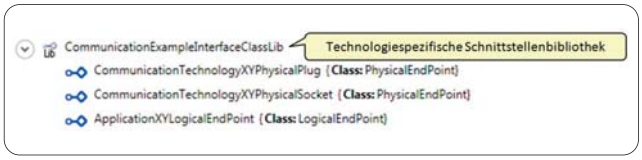

Bild 5: Technologieabhängige Interface-Klassen für das Anwendungsbeispiel

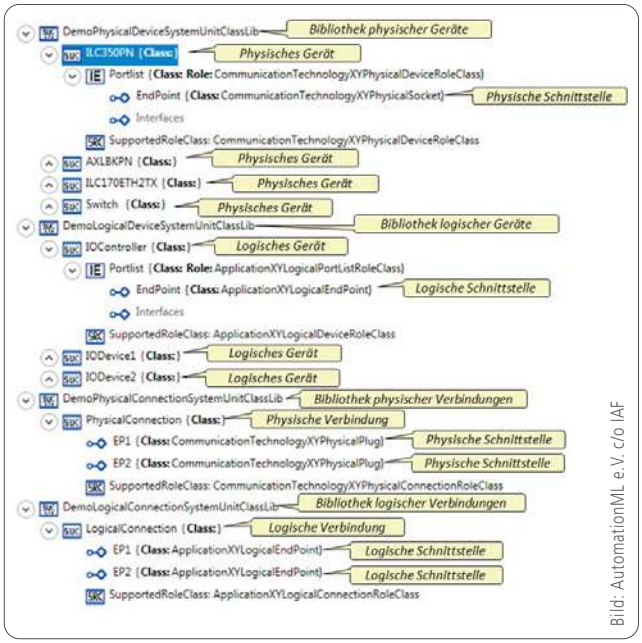

Bild: AutomationML e.V. c/o IAF

Bild 6: <SystemUnitClassLib>s für das Anwendungsbeispiel

schaften (wie Adressen, Schutzklassen oder Durchgangsraten) und die Anforderungen an die Kommunikation (wie Laufzeiten und Auslastungen) von Bedeutung. Beide Topologien stehen zueinander in Beziehung: die physikalische Topologie implementiert eine oder mehrere logische. Dieses Konzept ist flexibel auch für komplexe Systeme verwendbar.

Technologieunabhängige Basis-Bibliotheken

Zur Modellierung solcher Strukturen mit AutomationML wurden im AutomationML e.V. technologieunabhängige Bibliotheken erstellt, die alle notwendigen Basisklassen für eine Beschreibung von Kommunikationssystemen enthalten. Die entstandenen Bibliotheken sind in Bild 2 dargestellt.

- Die <CommunicationRoleClassLib> beinhaltet acht Basisrollenklassen für physikalische Netzwerke, physikalische Geräte und physikalische Verbindungen sowie für logische Netzwerke, logische Geräte und logische Verbindungen. Diese Klassen sind gemäß den Regeln von AutomationML von der Basisrolle <AutomationMLBaseRole> abgeleitet.
- Die Interface-Klassenbibliothek <CommunicationInterfaceClassLib> stellt zwei Basis-Interface-Klassen für die Modellierung bereit. Die dort enthaltenen Interface-Klassen werden für die Beschreibung der physischen und logischen Schnittstellen verwendet.

Anwendungsbeispiel

Für eine bessere Erklärung des Vorgehens soll an dieser Stelle ein Teil der Lemgoer Lernfabrik des Anwendungszentrums Industrial Automation (INA) des Fraunhofer IOSB modelliert werden. Im Beispiel ist eine Steuerung über die Kommunikationstechnologie XY (z.B. ethernetbasiertes Profinet) mit zwei E/A Geräten verbunden (Bild 3). Dabei dienen die E/A Geräte zum Anschließen der Sensoren und Aktoren an das Steuerungssystem.

Modellierung in fünf Schritten

Schritt 1: Im ersten Schritt wird für jede der o.g. acht technologieunabhängigen Rollenklassen jeweils eine technologiespezifische Rollenklasse abgeleitet – diese Bibliothek kann später wiederverwendet werden. Im Beispiel wären dies die CommunicationTechnologyXY und die ApplicationXY stellvertretend für beispielsweise Profinet, EthernetIP, Interbus oder Sercos sowie für gebräuchliche Applikationen wie die Steuerung von Sensoren und Aktoren oder die Webserver basierte Konfiguration [1]. Daraus entsteht eine technologieabhängige Rollenbibliothek mit acht Rollenklassen (Bild 4).

Schritt 2: Im sich daran anschließenden zweiten Schritt werden von den generischen Interface-Klassen entsprechende technologie- und applikationsbezogene Klassen abgeleitet (Bild 5).

Schritt 3: Im dritten Schritt wird eine <SystemUnitClassLib> erstellt, die alle physikalischen und logischen Geräte und Verbindungen als <SystemUnitClass> modelliert, beispielsweise die Steuerung ILC350PN. Dabei wird die Bedeutung der einzelnen Elemente durch Verknüpfung zu den Klassen aus Schritt 2 festgelegt. Auch diese Bibliothek ist später wiederverwendbar. Da für verschiedene Geräte im Bereich der Steuerungstechnik zumeist schon Gerätebeschreibungen existieren, ist hier kein großer Aufwand zur Erstellung der Bibliotheken zu erwarten. Im Anwendungsbeispiel ergeben sich vier physikalische Geräte (drei Steuerungsgeräte mit jeweils einer physikalischen Schnittstelle sowie den Switch als physikalisches Gerät mit sechs Schnittstellen) sowie drei logische Geräte. Ebenso entstehen Klassen für die Verbindungen, die im Falle der physikalischen Verbindungen den Kabeln oder auch Funkstrecken entsprechen. Die sich für das Beispiel ergebenden <SystemU-

Literatur

[1]F. Klasen, V. Oestreich, M. Volz: Industrielle Kommunikation mit Feldbus und Ethernet, VDE-Verlag, 2010.

nitClassLib>s sind in Bild 6 dargestellt. Spezielle Eigenschaften der physikalischen und logischen Geräte und Verbindungen sowie der in ihnen befindlichen Schnittstellen werden in AutomationML durch entsprechende Attribute an den <InternalElement>s, den <SystemUnitClass>es und den <InterfaceClass>es modelliert. Im Ergebnis sind alle Voraussetzungen für die Netzwerkmodellierung gegeben. Im vierten und fünften Schritt wird jetzt das eigentliche Netzwerkmodell erstellt.

Schritt 4: Im vierten Schritt wird mit der Gerätemodellierung begonnen. Dazu werden zuerst alle physischen Geräte als neue <InternalElement>s in einer entsprechenden <InstanceHierarchy> von den passenden <SystemUnitClass>es instanziiert. Enthält ein physisches Gerät logische Geräte, werden diese als Kinder eingefügt. Dann werden alle relevanten Parameter mit konkreten Werten belegt.

Schritt 5: Abschließend werden die Geräte verbunden. Dazu werden zunächst in der <InstanceHierarchy> <InternalElement>s erzeugt, die von den Rollen <PhysicalNetwork> und <LogicalNetwork> abgeleitete Rollen besitzen. Sie dienen als Container für alle physikalischen bzw. logischen Verbindungen. Diese werden als nächstes durch Instanziierung der entsprechenden <SystemUnitClass>es erstellt und mit entsprechenden Parametern belegt. Zur Darstellung des Zusammenhangs zwischen Geräten und Verbindungen werden als Letztes die entsprechenden Interfaces der Geräte und Verbindungen über <InternalLink>s miteinander identifiziert. Das Modellierungsergebnis für das Anlagenbeispiel ist in Bild 7 gezeigt.

Fazit

In fünf Schritten zeigt dieser Beitrag auf, wie Kommunikationsnetzwerke mit AutomationML abgebildet werden können. Die vorgestellte Methode ist universell und sowohl für einfache als auch für komplexe und verschachtelte Kommunikationsnetzwerke geeignet. Sie basiert auf vorgefertigten technologieunabhängigen Basisklassen, auf deren Basis sich wiederverwendbare technologiespezifische Rollen- und Schnittstellenbibliotheken erstellen lassen. Die beschriebene Methodik ermöglicht ein praxisnahes und iteratives Vorgehen: Logische und physikalische Netzwerke können (müssen aber nicht) zur gleichen Zeit entstehen oder gemeinsam genutzt werden. Ebenso ermöglicht dieses Vorgehen eine stufenweise Datenweitergabe. Bei der Übertragung der Entwurfsdaten können entweder die <RoleClassLib>s, <InterfaceClassLib>s und <SystemUnitClassLib>s weitergegeben werden, alternativ könnten sie sogar im Internet veröffentlicht werden. Sind sie dem Empfänger bereits bekannt, können sie referenziert werden und müssen nicht mit übertragen werden. Derzeit beginnen erste Arbeiten an der Umsetzung von Pilotanwendungen zur Nutzung der Darstellungsmethode. Die Ergebnisse dieser Arbeit sollen im Rahmen der Standardisierung von AutomationML in die IEC62714 einfließen. Feldbusorganisationen sind eingeladen, die beschriebene Vorgehensweise aufzugreifen und für die von Ihnen unterstützten Kommunikationstechnologien entsprechende <RoleClassLib>s, <InterfaceClassLib>s und <SystemUnitClassLib>s bereitzustellen. Dies würde sicherstellen, dass bisherige Entwicklungen in die Gerätemodellierung nahtlos übernommen werden können (insbesondere Parameterbenennungen, Geräteprofile, etc.) und eine breite Akzeptanz des Vorgehens erreichbar ist. ■

www.automationml.org

Autor: Dr.-Ing. Matthias Riedl, Bereichsleiter Integrierte Kom. , Institut f. Automation und Kommunikation e.V., Magdeburg

Autor: apl. Prof. Dr.-Ing. habil. Arndt Lüder, Leiter Center Verteilte Systeme (CVS), Otto-von-Guericke Universität Magdeburg

Autorin: Nicole Schmidt, Wissenschaftliche Mitarbeiterin CVS, Otto-von-Guericke Universität Magdeburg; Inst. für Arbeitswissenschaft, Fabrikautom. und Fabrikbetrieb

Autor: Dr.-Ing. Rainer Drath, Senior Principal Scientist, ABB AG Forschungszentrum Deutschland

Autor: Benno Heines, Gruppenleiter Research & Development Control Systems, Phoenix Contact Electronics GmbH, Bad Pyrmont

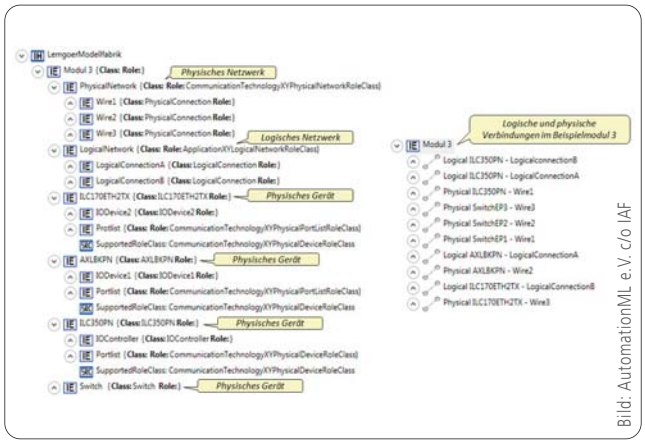

Bild 7: Netzwerkbeschreibung für das Anwendungsbeispiel