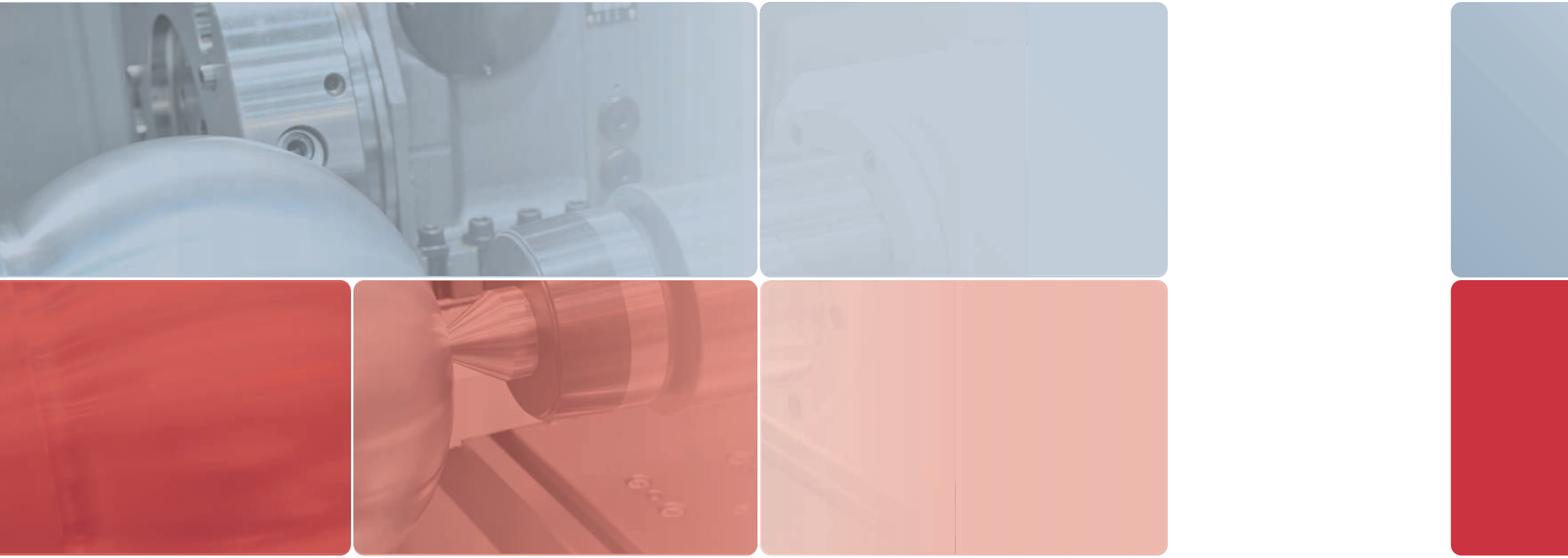
SPS MAGAZIN

AutomationML – Fachexperten erklären das Format

Whitepaper



AutomationML – Fachexperten erklären das Format

Dieses Whitepaper umfasst Beiträge, die den Aufbau des AutomationML Datenaustauschformats selbst beschreiben und die Möglichkeiten, die es durch seinen Aufbau bietet. Es werden tiefere Einblicke in die Möglichkeiten und Notwendigkeiten zur Implementierung von AutomationML Schnittstellen gegeben, um die Anwendung in Software-Werkzeugen zu erleichtern. Exemplarische Anwendungsbeispiele zeigen auf, wo AutomationML bereits eingesetzt wird bzw. werden kann.

AutomationML – ein Überblick

Ein Standard für die Verbesserung des Datenaustauschs von Engineeringwerkzeugen

Bild 1: Aktuell von AutomationML abgedeckte Aspekte

Durch die zunehmende Bedeutung der Software im Bereich der Industrie, und speziell in der Automatisierungstechnik, haben sich neue Schnittstellenprobleme gezeigt. Wenn man sich den ganzen Prozess von der Produktentwicklung bis hin zur Qualitätssicherung betrachtet, dann gibt es immer dieselben Daten, die durch den gesamten Prozess hindurch benötigt und geschleust werden. Aktuell gibt es viele Systembrüche der unterschiedlichen Engineeringwerkzeuge. AutomationML hat zum Ziel, diesen Datenaustausch signifikant zu vereinfachen. Wir stellen AutomationML im Rahmen einer sechsteiligen Serie vor und beginnen mit einem Überblick.

Die AIDA-Gruppe (Automatisierungsinitiative Deutscher-Automobilhersteller) stellte im Jahre 2005 in einer gemeinsamen Untersuchung fest, dass alleine über 50% der Kosten der gesamten Anlagensteuerungstechnik auf das Engineering entfielen (Bild 2). Daraufhin startete Daimler ein Projekt, mit dem Ziel, diese Engineeringkosten mindestens zu halbieren. Bald war klar, dass einen erheblichen Anteil des Engineeringaufwandes das Übertragen der Engineering-Daten von einem Tool zum anderen verursacht. Nach ersten Evaluierungen von Austauschformaten initiierte Daimler im Oktober 2006 die Entwicklung und Standardisierung der AutomationML als Zwischenformat der Digitalen Fabrik, zusammen mit den Firmen ABB, Kuka, Rockwell Automation, Siemens, netAllied und Zühlke sowie den Universitäten Karlsruhe und Mag-

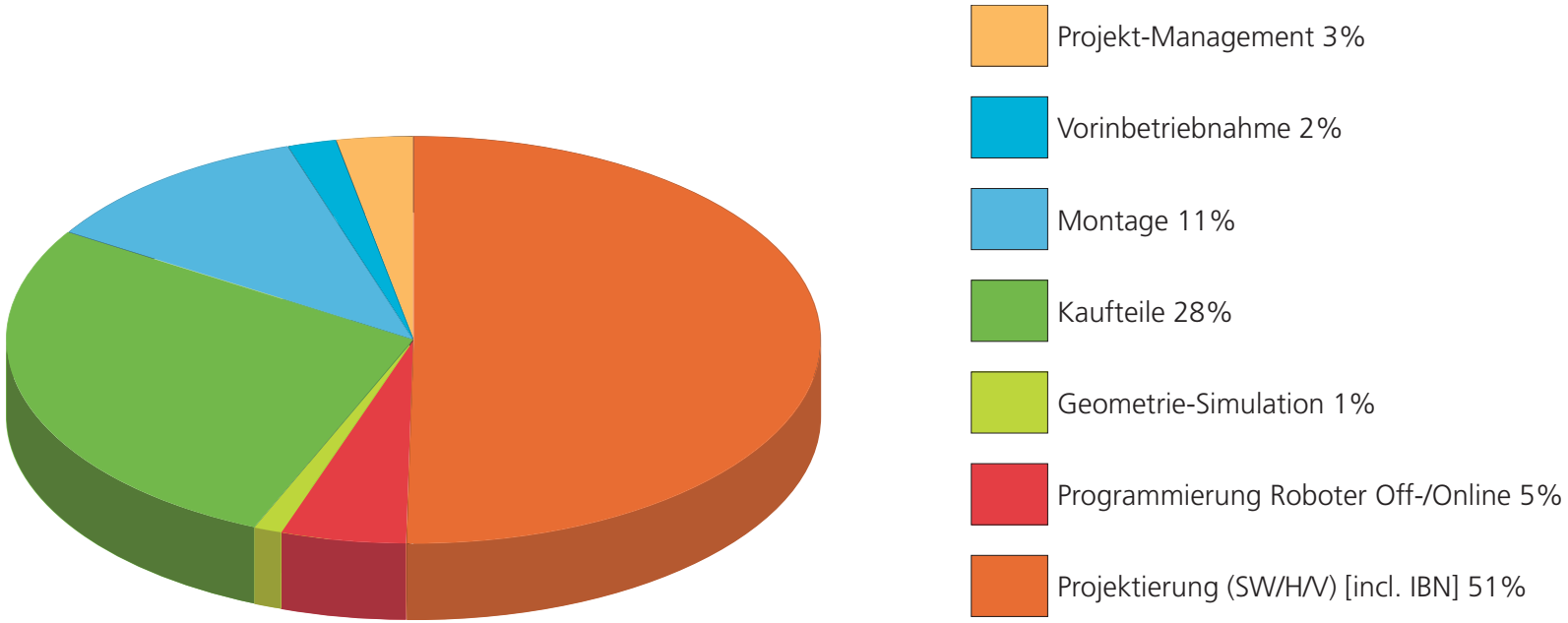


Bild 2: Kostenstruktur der Steuerungstechnik im Anlagenbau aus dem Jahre 2005

deburg. Im Frühjahr 2009 öffnete sich das bis dahin geschlossene Industriekonsortium durch die Gründung eines Vereins.

Motivation und Zielsetzung

Wie in Bild 2 zu sehen ist, stellt in der Automatisierungstechnik die Planung der Anlagen einen großen Kostenfaktor dar. Für die verschiedenen Aufgaben wie z.B. Hardwaredesign, Software-Projektierung oder die virtuelle Inbetriebnahme, werden unterschiedliche Engineering-Werkzeuge benötigt und eingesetzt. Teilweise kommen Werkzeuge in speziellen Nischen zum Einsatz und auch Hersteller-Tools zur Parametrierung und Programmierung von Komponenten sind am Prozess beteiligt. Des Weiteren wird jeder Planer das ihm vertraute, also sein spezielles 'Best in Class-Tool' einsetzen wollen. Aufgrund dieser heterogenen Toollandschaft stellt der Datenaustausch zwischen den Werkzeugen einen wichtigen Aspekt dar. Aktuell werden die Daten oft in proprietären Datenformaten übertragen und gespeichert, die sich nur mit einem sehr begrenzten Kreis an Werkzeugen, oft nur firmenspezifischen Tools, einlesen lassen. Stehen keine Schnittstellen zur Verfügung, müssen Daten in den verschiedenen Werkzeugen manuell eingegeben und nachgepflegt werden. Insgesamt führt dies zu einem sehr ineffizienten und fehleranfälligen Planungsprozess. Der AutomationML e.V. hat sich zum Ziel gesetzt, diesen Datenaustausch zwischen den Engineeringwerkzeugen zu verbessern und zu vereinheitlichen. Dabei wurde von Anfang an konsequent darauf geachtet, dass kein neues Datenformat entwickelt, sondern, dass auf bereits existierende Formate zurückgegriffen wird. Diese Formate sollten bzw. müssen folgenden Kriterien genügen:

- Sie müssen für jeden Anwender absolut frei zugänglich und kostenlos sein.
- Die Formate sollten nach ISO oder IEC standardisiert sein, bzw. dieses ist anzustreben.
- Die Formate sollten XML-basiert sein.
- Die Formate müssen noch gepflegt werden und der Besitzer des Formats muss für Weiterentwicklungsvorschläge unsererseits offen sein.

Mithilfe von AutomationML sollen mechatronische Objekte bis

hin zu ganzen Fertigungsanlagen in den für den Engineeringprozess relevanten Aspekten beschrieben werden. AutomationML umfasst aktuell folgende Bereiche (Bild 1):

- Topologie zur Abbildung von Strukturen, Attributen, Schnittstellen, Relationen,
  Geometrie und Kinematik,
- Logik und Verhalten,
- Semantik und Ontologien von Objekten auf Basis von Bibliotheken.

Weitere Bereiche wie z.B. Kommunikation und Mechatronik befinden sich in der Vorbereitung. AutomationML kombiniert existierende XML-Datenformate zur Abdeckung dieser Bereiche. CAEX, standardisiert über IEC62424, bildet das Dachformat zur Abbildung der Topologie. Aus diesem Format heraus wird auf das Format Collada verlinkt, in dem Geometrie und Kinematik abgebildet werden. Für die Umfänge Logik und Verhalten steht PLCopen XML zur Verfügung. Das Konzept von Rollenbibliotheken ermöglicht die Darstellung von Semantiken.

Einige Beispielapplikationen

Daimler hat als erster Nutzer des Formats sehr früh begonnen AutomationML in seine Planungsprozesse zu integrieren. So entstanden in den letzten Jahren u.a. folgende Anwendungen, die mittlerweile alle im industriellen Umfeld eingesetzt sind:

- Automatische Bahnplanung ‘Pathfinder’ für Roboter
- Visualisierung der Fabrikplanung mit VEO:factory
- Virtuelle Inbetriebnahme

Im Rahmen dieser Realisierungen wurden Topologieinformationen inklusive Rollen, Geometrie- und Kinematikinhalte sowie anwendungsspezifische Daten erfolgreich zwischen den Werkzeugen ausgetauscht. Die Anwendungen haben nachgewiesen, dass der Datenaustausch mit AutomationML funktioniert, auch wenn noch nicht alle Teile komplett integriert sind.

Ein kurzer Blick voraus

AutomationML erweitert seine Akzeptanz
AutomationML wird ein etablierter Standard für den Austausch von Engineering – Daten im Anlagen und Maschinenbau werden. Dazu stellt der Verein umfassende Dokumentationen und Informationsmaterial zur Nutzung von AutomationML sowie die verschiedensten Beispielprojekte aus den unterschiedlichen industriellen Anwendungsfeldern auf seiner Homepage www.automationml.org bereit.

AutomationML ist standardisiert
Der AutomationML e.V. hat in den vergangenen Jahren bereits viel Energie in die Standardisierung der zentralen Bestandteile von AutomationML gelegt. So erwarten wir in 2013 den Abschluss des ersten Teils des Standards IEC62714 -1 ‘Engineering data exchange format for use in industrial automation systems engineering – Part 1: Architecture and General Requirements’. Die anderen Teile werden momentan parallel in den Gremien der IEC und DKE Arbeitsgruppe vorangetrieben.

AutomationML e.V. ist innovativ
Der AutomationML e.V. hat sich zur Aufgabe gemacht, in bestimmten Zyklen die momentanen Lösungen zu überprüfen und gegebenenfalls anzupassen.

AutomationML e.V. bietet technologische Unterstützung
Der AutomationML e.V. bietet zur technologischen Unterstützung und Qualitätssicherung bei der Anwendung und der Umsetzung des Datenformats Konformitätstests und Referenzwerkzeuge an. Die AutomationML Engine und auch der Editor werden technologisch weitergeführt und weiterentwickelt.

AutomationML e.V. ist eine offene Community
Der AutomationML e.V. ist offen für Firmen, Institute und Hochschulen. Der Mitgliederbeitrag ist überschaubar. Mitglieder können ihre Ideen direkt in das Format einbringen. ■

www.automationml.org

Autor: Anton Hirzle, Senior Manager, Daimler AG

AutomationML - die Architektur

Serie AutomationML Teil 2: Ein Standard für die Verbesserung des Datenaustauschs von Engineeringwerkzeugen

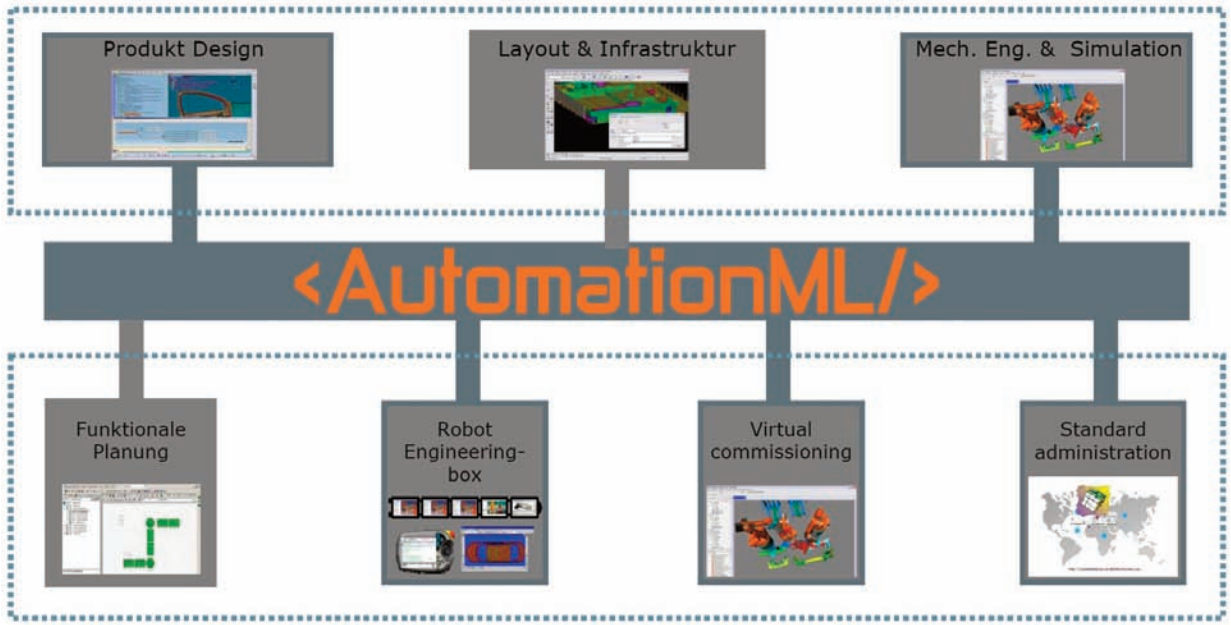


Bild 1: Angestrebte Nutzung von AutomationML im Entwurfsprozess

Durch die zunehmende Bedeutung der Software im Bereich der Industrie und speziell in der Automatisierungstechnik haben sich neue Schnittstellenprobleme gezeigt. Aktuell gibt es viele Systembrüche zwischen den unterschiedlichen Engineeringwerkzeugen. AutomationML hat zum Ziel, diesen Datenaustausch signifikant zu vereinfachen. Wir stellen AutomationML im Rahmen einer sechsteiligen Serie vor und fahren in Teil 2 fort mit der Architektur.

Der AutomationML e.V. betrachtet den Entwurfsprozess von Produktionssystemen. Dies geht vom Produktentwurf über den Anlagenentwurf und das 'Detailed Engineering' bis hin zur virtuellen Inbetriebnahme und zur Anlagenstandardisierung. Er ist dabei bestrebt, ein Datenaustauschformat zu entwickeln, das für jede beliebige Paarung von Werkzeugen der Entwurfskette als Datenformat zur Weitergabe von Entwurfsergebnissen genutzt werden kann. Dies ist in Bild 1 beispielhaft dargestellt. AutomationML stellt dabei das reine Datenformat dar und befasst sich nicht mit der Übertragung der Inhalte oder Schnittstellen zu entsprechenden Werkzeugen.

Architektur von AutomationML

Ausgangspunkt für AutomationML bildet die Abbildung der Struktur bzw. Topologie eines Produktionssystems. Diese umfasst die hierarchische Strukturierung der Anlagenobjekte, die jeweils durch individuelle Datenobjekte repräsentiert werden. Dabei wird die Anlage bzw. Teilanlage bis zu einem spezifischen, den Notwendigkeiten der jeweiligen Anwendungsfälle angepassten Detaillierungsgrad untergliedert. Für jedes Anlagenobjekt werden die entsprechenden auszutauschenden Informationen als Objekteigenschaften abgebildet. Zusammenhänge zwischen Anlagenobjekten werden über Relationen repräsentiert. Zur Darstellung der Struktur- und Topologieinformationen wird auf das bereits international in der IEC62424 standardisierte Datenformat CAEX (Computer Aided

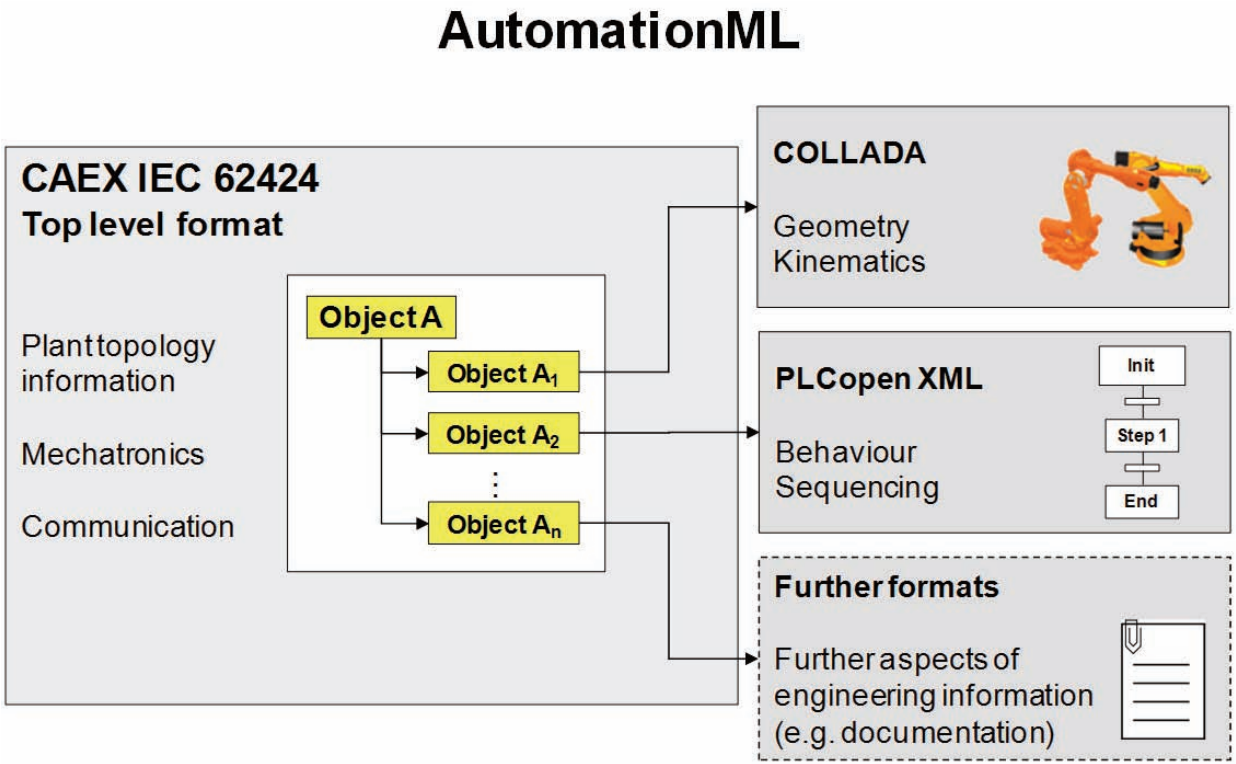


Bild 2: Toplevel-Architektur von AutomationML

Engineering Exchange) zurückgegriffen. Dieses bietet objektorientierte Grundkonzepte, die für die Darstellung von Anlagenstrukturen eingesetzt werden. Dazu spezifiziert AutomationML detailliert, wie welches Grundkonzept von CAEX einzusetzen ist und definiert darüber hinaus Einschränkungen und Regeln für den Einsatz von CAEX. Das erste der genutzten Grundkonzepte von CAEX ist die InstanceHierarchy. Sie ermöglicht die Abbildung der hierarchischen Anlagenstruktur eines Produktionssystems. Mithilfe der SystemUnitClassLibraries werden (herstellerspezifische) Bibliotheken von Anlagenkomponenten beschrieben. Die RoleClassLibraries ermöglichen die Definition und Nutzung von Semantiken für Anlagenkomponenten bzw. für ihre Beschreibungsmittel. Das vierte genutzte Grundkonzept sind die InterfaceClassLibraries. Sie umfassen die Definition verschiedener Schnittstellentypen, mit denen Zusammenhänge zwischen verschiedenen Anlagenkomponenten sowie Beziehungen zu extern abgespeicherten Informationen beschrieben werden können. Die Bibliothekskonzepte ermöglichen die Definition, Darstellung und Nutzung von Templates für einzelne Objekte. Über sie können allgemein verbindliche als auch anwenderspezifische Grundstrukturen beschrieben und ausgetauscht werden. Die InstanceHierarchy erfüllt für das eigentliche Engineering-Projekt den gleichen Zweck. Mittels des Konzeptes der Rollenbibliotheken (RoleClassLib) kann einzelnen Objekten sowohl in der SystemUnitClassLibrary als auch der InstanceHierarchy eine bestimmte Semantik zugeordnet werden. Dadurch wird den Objekten unabhängig von ihrer eigentlichen Benennung eine Bedeutung im Rahmen der Anlage zugewiesen. Das Konzept der Interfaces und Relationen ermöglicht es, Objekte über Hierarchiegrenzen hinweg in Beziehung zu setzen und schafft die Möglichkeit, auf Informationen außerhalb der CAEX-Beschreibung zu referenzieren. Dazu werden entsprechende InterfaceClass-Objekte im Rahmen einer InterfaceClassLibrary spezifiziert. Diese werden in der SystemUnitClassLibrary bzw. der InstanceHierarchy genutzt und mittels InternalLinks verbunden. Zudem ermöglichen Referenzen in den Objekten der InstanceHierarchy, die Instanziierung von Objekten der SystemUnitClassLibrary. Den einzelnen Objekten der Struktur- und Topologiebeschreibung können Geometrie- und Kinematikinformationen zugeordnet werden. Diese beschreiben die geometrischen Verhältnisse der einzelnen Objekte als Körper in beliebiger Detailliertheit sowie die Eigenschaften der Beweglichkeit dieser und ihren Zusammenhang.

XML-basiertes Datenformat als Grundlage

Grundlage der Geometrie- und Kinematikbeschreibung ist das aus der Computerspieleindustrie stammende, offene, XML-basierte Datenformat Collada in Version 1.4.1 und 1.5.0 (ISO/PAS 17506:2012). Dieses Format ermöglicht die Darstellung der geometrischen und kinematischen Eigenschaften von Körpern als geometrische und kinematische Szenen. Unter Nutzung von Referenzierungsmechanismen, die in CAEX in der InterfaceClassLibrary spezifiziert sind, werden die einzelnen Geometrie- und Kinematikbeschreibungen an die betreffenden Objekte der Struktur- und Topologiebeschreibung gebunden. In Analogie zu Geometrie- und Kinematikinformationen können den einzelnen Objekten der Struktur- und Topologiebeschreibung Logik- und Verhaltensinformationen zugeordnet werden. Dabei ist es möglich, die verschiedenen denkbaren Verhaltensweisen von Systemen, Komponenten und Geräten innerhalb eines Produktionssystems abzubilden. Die Logik- und Verhaltensbeschreibung basiert auf dem bekannten und praktisch weitreichend genutzten Datenformat PLCopen XML in Version 2.0 und 2.0.1, das initial für den Austausch von Steuerungsprojekten nach IEC61131-3 spezifiziert wurde. AutomationML definiert, wie mit diesem Format unterschiedlichste, in der Praxis verwendete Modelle für Logik- und Verhaltensinformationen modelliert werden. So können unter anderem Gantt Charts, PERT Charts, Impulsdiagramme, Sequential Function Charts, State Charts und Cause and Effect Matrizen konsistent und verlustfrei übertragen werden. Auch sie werden unter Nutzung von Referenzierungsmechanismen, die in CAEX in der InterfaceClassLibrary spezifiziert sind, mit den sie betreffenden Objekten der Struktur- und Topologiebeschreibung verbunden. AutomationML entwickelt derzeit weitere Anwendungsmöglichkeiten der genannten Formate und Strukturen z.B. für die Beschreibung von Kommunikationssystemen oder mechatronischen Einheiten.

Anwendungsbeispiele für AutomationML

Es wurden bereits einige Anwendungsmöglichkeiten durch die Mitglieder des AutomationML umgesetzt, von denen wiederum einige schon im produktiven Einsatz sind. So können z.B. die Konstruktionsdaten aus Catia V5 sowie die Bewegungsplanungen aus RobotStudio sicher und verlustfrei an Robcad über AutomationML übergeben werden, so dass die virtuelle Inbetriebnahme einer geplanten Roboterzelle ermöglicht wird. Dabei kommt das integrierte Format Collada zum Einsatz. Weiter können Ablaufbeschreibungen, die z.B. als Gantt Chart in Excel modelliert sind, oder Schrittketten, die in Delmia V5 erstellt wurden, direkt an Steuerungsprogrammierwerkzeuge wie Multiprog weitergegeben werden. Hierfür wird unter anderem das integrierte Format PLCopen XML genutzt. AutomationML bietet außerdem mit dem Austausch hierarchischer Anlagenstrukturen die Möglichkeit, die Anlagenstruktur, z.B. gemäß der Ortskennzeichen der Implementierung, weiterzugeben. Zudem kann für jedes Element innerhalb der Ortsstruktur die vollständige Dokumentation sowie die wichtigsten konstruktiven Daten als Parameter in der hierarchischen Struktur repräsentiert werden. Dadurch ist eine lückenlose und gut strukturierte Dokumentation der aktuellen Anlagenstruktur, z.B. für den Wartungstechniker, vorhanden. Ebenso ist ein gezielter Austausch von entsprechenden Informationen zwischen verschiedensten Werkzeugen bis hin zu ERP-Systemen möglich. Damit können z.B. Geräteinformationen zwischen Herstellern und Anwendern ausgetauscht und einfach in die Dokumentation integriert werden. In den folgenden SPS-Magazinen werden einige Anwendungsfälle, Einsatzmöglichkeiten und Technologien zur Nutzung von AutomationML beschrieben. ■

www.automationml.org

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

Autor: Lorenz Hundt, Wissenschaftlicher Mitarbeiter CVS, Otto-von-Guericke Universität Magdeburg

Autor: Nicole Schmidt, Wissenschaftliche Mitarbeiterin CVS, Otto-von-Guericke Universität Magdeburg

Autor: Miriam Schleipen, Fraunhofer Institut IOSB, Gruppenleiterin Leitsysteme und Anlagenmodellierung

AutomationML-Programmierung

Serie AutomationML Teil 3: Effizientes Programmieren mit der AutomationML-Engine

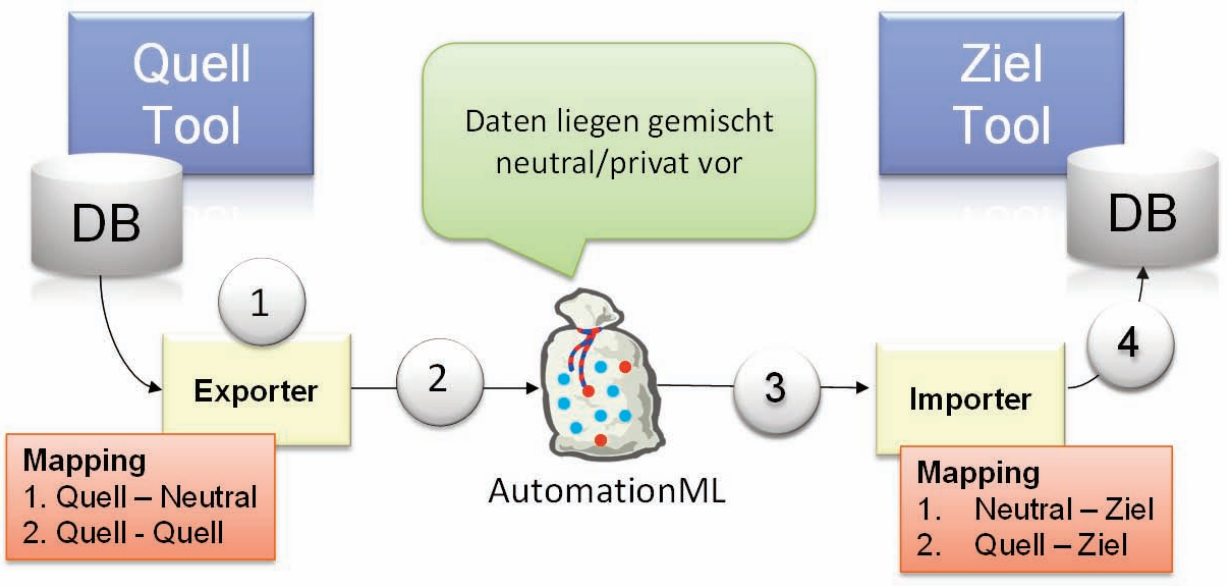


Bild 1: Realer Datenaustausch mit CAEX

Ziel von AutomationML [1] ist der Datenaustausch zwischen Werkzeugen der Anlagenplanung. Als herstellerneutrales, standardisiertes und XML-basiertes Datenformat kann es derzeit Planungsinformationen bezüglich Anlagentopologie/Struktur, Geometrie/Kinematik sowie Logik/Verhalten abbilden. Dabei bildet CAEX [2] das Objektmodell der Engineeringdaten ab. Die Integration von AutomationML in die Schnittstellen von Werkzeugen wirft jedoch immer wieder die Frage auf: Wie aufwändig ist die Programmierung von Ex- und Importern? Dieser Beitrag beleuchtet dies anhand verschiedener Programmierszenarien.

Parallel zu AutomationML entwickelte der AutomationML e.V. eine freie Software für Entwickler – die AutomationML-Engine [3]. Sie spiegelt das CAEX-Datenmodell in Form einer C#-Klassenstruktur exakt wider und enthält alle Klassen und Methoden, um CAEX-Objekte (Klassen und Instanzen) manipulieren zu können. Der Softwareentwickler wird durch diese Software befreit, die Einhaltung des CAEX-Schemas zu überwachen. Stattdessen operiert er auf Klassenebene, während die Engine im Hintergrund konforme CAEX-Dateien erzeugt oder liest. Entwicklungsvoraussetzung zur Nutzung der AutomationML-Engine ist das Betriebssystem WindowsXP oder höher sowie Visual Studio Version 2008 oder höher. Die Einbindung der Engine in eigene Projekte erfolgt über die Referenzierung der Engine-dlls.

AutomationML-Programmierung

Bild 1 zeigt das Basisszenario: Der Datenaustausch zwischen zwei Werkzeugen benötigt einen Exporter und einen Importer. Sowohl Exporter als auch Importer müssen AutomationML-Dateien erzeugen, lesen und manipulieren können. Daraus lassen sich Basisfunktionen und erweiterte Funktionen ableiten, die im Folgenden schrittweise beleuchtet werden.

```
CAEXDocument myDoc = CAEXDocument.New_CAEXDocument();
```

Bild 2: Beispielcode zum Erzeugen eines leeren CAEX-Dokumentes

```
myDoc.SaveToFile(@"c:\temp\test.aml", true);
```

Bild 3: Beispielcode zum Speichern eines CAEX-Dokumentes

```
myDoc = CAEXDocument.LoadFromFile(@"c:\temp\test.aml");
```

Bild 4: Beispielcode zum Laden eines CAEX-Dokumentes

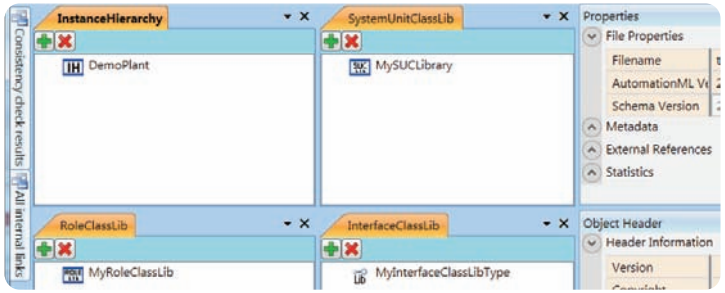


Bild 5: Hierarchien im AutomationML-Editor

```
//variabe declaration
CAEXFileType myCAEXFile = myDoc.CAEXFile;
InstanceHierarchyType IH;
SystemUnitClassLibType SUCL;
RoleClassLibType RCL;
InterfaceClassLibType ICL;

//create instance hierarchy
IH = myCAEXFile.New_InstanceHierarchy("DemoPlant");
SUCL = myCAEXFile.New_SystemUnitClassLibHierarchy("MySUCLibrary");
RCL = myCAEXFile.New_RoleClassLibHierarchy("MyRoleClassLib");
ICL = myCAEXFile.New_InterfaceClassLibHierarchy("MyInterfaceClassLibType");
```

Bild 6: Beispielcode zum Erzeugen der Hierarchien

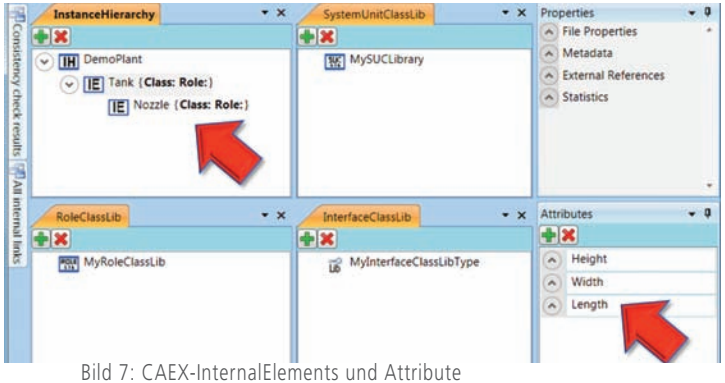


Bild 7: CAEX-InternalElements und Attribute

CAEX-Basisprogrammierung

1. Erzeugen eines CAEX-Dokumentes

Für das Erzeugen eines leeren CAEX-Dokumentes genügt eine Zeile Quelltext (siehe Bild 2). Im Hintergrund, für den Anwender verborgen, erzeugt die AutomationML-Engine alle benötigten XML-Knoten [4].

2. Öffnen/Speichern einer CAEX-Datei

Das Öffnen und Speichern eines CAEX-Dokumentes erfordert ebenfalls jeweils nur eine einzige Zeile Quelltext (siehe Bild 3 und Bild 4). Dies funktioniert mit beliebig komplexen CAEX-Dateien.

3. Erzeugen von CAEX-Hierarchien

CAEX unterstützt vier Arten von Hierarchien: eine Instanz-Hierarchie für die eigentlichen Projektdaten sowie drei Bibliotheksarten. Im AutomationML-Editor [3] kann jede dieser Hierarchien mit einem einzigen Mausklick erzeugt werden (siehe Bild 5). Bild 6 zeigt den zugehörigen C#-Quelltext, um alle vier Hierarchien zu erzeugen. Dieses Beispiel unterstreicht eine Erfahrung des Autors: Das Programmieren eines einfachen CAEX-Browsers gelingt innerhalb von 30min.

4. Erzeugen von Hierarchie-Elementen

Im nächsten Schritt wird die Instanz-Hierarchie mit Objekten gefüllt. Ihre Architektur ermöglicht das Modellieren beliebig tiefer Objektstrukturen. Bild 7 zeigt dies beispielhaft anhand von zwei Objekten Tank und Nozzle im AutomationML-Editor sowie drei Attributen des Tanks. Bild 8 zeigt den zugehörigen C#-Code: Zunächst werden Variablen IE1 und IE2 deklariert. Anschließend werden beide Objekte Tank und Nozzle erzeugt, der Tank wird mit Attributen versehen.

```
//create internal element
InternalElementType IE1, IE2;
IE1 = IH.New_InternalElement("Tank");
IE1.New_Attribute("Length");
IE1.New_Attribute("Width");
IE1.New_Attribute("Height");
//create child internal element
IE2 = IE1.New_InternalElement("Nozzle");
```

Bild 8: Beispielcode zum Erzeugen von CAEX-Objekten

```
MetaInformation m = new MetaInformation();
m.WriterName = "a Source Tile";
m.WriterID = "a Source Tool";
m.WriterVendor = "a Company";
m.WriterVendorURL = "www.acompany.com";
m.WriterVersion = "1.0";
m.WriterRelease = "1.0";
m.LastWritingDateTime = "2012.09.21";
m.WriterProjectTitle = "TestProject";
m.WriterProjectID = "TestProject";
myDoc.CAEXFile.SetMetaInformation(m);
```

Bild 9: Beispielcode zum Etikettieren von CAEX-Dateien

Zwischenfazit

Der Aufwand für die grundlegenden Schritte beim Programmieren von AutomationML/CAEX ist tatsächlich gering. Dies ist nicht auf die beschriebenen Basisanforderungen beschränkt, auch komplexere Funktionen wie das Löschen von Objekten, das Ändern von Attributen, das Referenzieren von Objekten, Funktionen zum Kopieren und Klonen ganzer Teilhierarchien zwischen verschiedenen CAEX-Dokumenten gelingen ähnlich einfach.

Erweiterte Techniken

1. Umgang mit semantischer Viel-falt

AutomationML führte 2011 ein neues Konzept zur Beherrschung semantischer Vielfalt ein. Dies ermöglicht die Realisierung von Datenaustausch zwischen Planungswerkzeugen, ohne dass hierzu ein standardisiertes Datenmodell existieren muss. Das

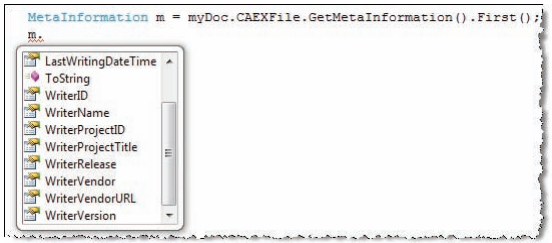


Bild 10: Beispielcode zum Lesen von Etiketten

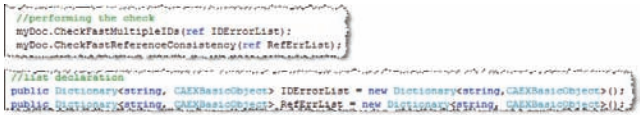


Bild 11: Beispielcode zum Validieren von CAEX-Dateien

Konzept basiert auf der Idee, proprietäre Daten-Teilmodelle in ihrer ursprünglichen Semantik in CAEX abzubilden. Dazu wird die AutomationML-Datei mit einem Etikett versehen, das Auskunft über den Urspung der Datei gibt. Importer können diese Informationen nutzen, um quelltoolspezifische Sub-Routinen zum Import aufzurufen. Das Konzept wird in [5] im Detail beschrieben, im Folgenden soll die Programmierung erläutert werden. Einzige Voraussetzung für die Anwendung dieses Konzeptes ist die Offenheit der adressierten Planungswerkzeuge [6].

2. Etikettierung von CAEX-Dateien

Bild 9 zeigt, wie das Etikettieren erfolgt: Zunächst wird exemplarisch ein Objekt m vom Typ Metainformation erzeugt. Dann werden die von AutomationML definierten Informationen festgelegt und abschließend wird das Objekt m im CAEX-Dokument eingebunden.

3. Lesen von CAEX-Etiketten

Bild 10 illustriert, wie das vorhandene Etikett einer CAEX-Datei ausgewertet werden kann. Die Intellisense-Funktion von Visual Studio hilft, die relevanten Properties zu sichten.

Fehlersuche in CAEX-Dateien

AutomationML-Dokumente sollen fehlerfrei sein. Fehler sind stets auf das Quellwerkzeug oder den Exporter zurückzuführen. Beim Entwickeln von Exportern treten typische Fehler auf: ID's werden doppelt vergeben, referenzierte Klassen oder externe Dateien existieren nicht. Die AutomationML-Engine bietet Funktionen, um solche Fehler leicht zu finden. Bild 11 zeigt, wie diese Fehlerprüfroutinen aufgerufen werden können. Die Fehler werden in Listen gespeichert, deren Einträge jeweils eine menschenlesbare Fehlerbeschreibung sowie einen Pointer auf das fehlerhafte Objekt enthalten.

Weitere Funktionen

Basierend auf der praktischen Nutzung der AutomationML-Engine wurde diese in 2011 um eine Vielzahl weiterer Funktionen erweitert. Dazu gehören Funktionen zum Importieren von CAEX-Daten aus anderen CAEX-Dateien, Funktionen für das Klonen, Ersetzen und Kopieren von CAEX-Daten, Festlegen von Meta-Daten sowie erweiterte Funktionen zum Splitten und Zusammenführen von CAEX-Dateien.

Zusammenfassung

Die AutomationML-Engine vereinfacht die Arbeit mit CAEX-Dateien erheblich und wird beispielsweise vom AutomationML-Editor bereits verwendet [7]. Der realistische Aufwand zum Programmieren eines AutomationML-Exporters wird auf wenige Tage reduziert, der Hauptaufwand liegt in der Kommunikation mit dem Engineeringwerkzeug. Das Programmieren von Importern ist aufwändiger und erfordert das Mappen der empfangenen Daten mit dem Datenmodell des Zielwerkzeuges. Diese Funktionalität liegt jedoch außerhalb der AutomationML-Programmierung und wird in einem weiteren Beitrag dieser AutomationML-Serie beleuchtet.

Literaturhinweise

[1] IEC 62714-1 CD norm draft AutomationML Architecture, www.iec.ch, 2011.
[2] IEC 62424:2008, Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools
[3] www.automationml.org
[4] W3C: Extensible Markup Language (XML), URL: http://www.w3.org/XML/ (last access: 19.04.2010).
[5] Drath R., Barth M: Beherrschung von Semantikvielfalt mit AutomationML, Vorgehensmodell für die semantische Standardisierung heterogener Datenmodelle – wie der Umgang mit unterschiedlichen Datenmodellen beim Datenaustausch im heterogenen Werkzeugumfeld gelingt. In atp 12/2012. Oldenbourgverlag 2012.
[6] Drath R., Barth M., Fay A.: Offenheitsmetrik für Engineering-Werkzeuge. In: atp 09/2012, S 46-55. Oldenbourgverlag, 2012.
[7] www.youtube.com/AutomationML

www.automationml.org

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

AutomationML – Datenaustausch

Serie AutomationML Teil 4: Implementierung von Zuordnungsstrategien für den Datenaustausch mit AutomationML

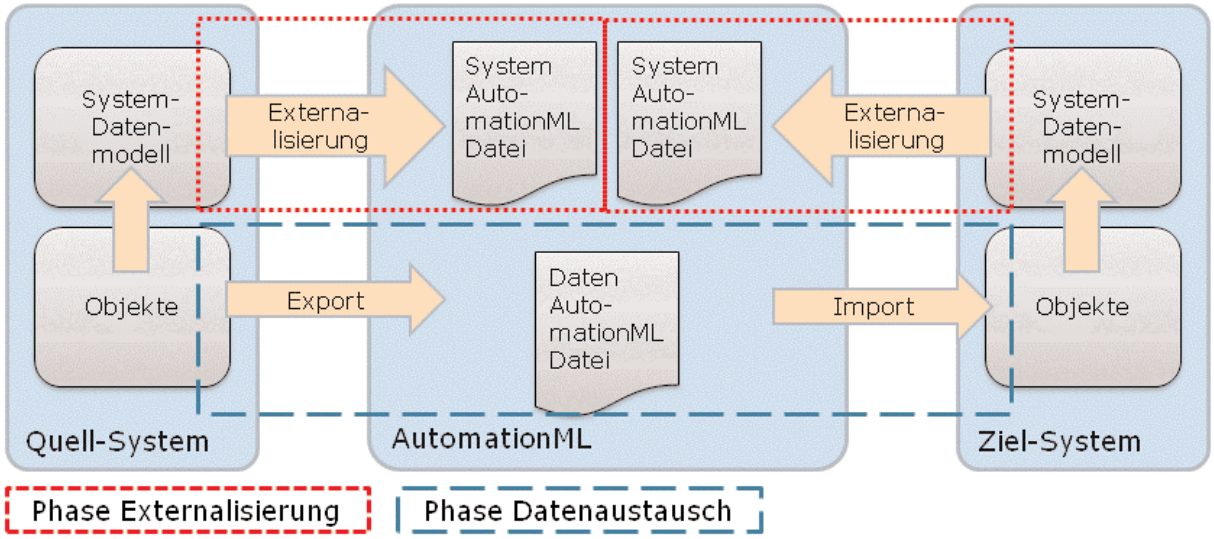


Bild 1: Phasen der Zuordnung

Ein wichtiges Problem beim Datenaustausch zwischen zwei Entwurfswerkzeugen ist die korrekte Übertragung der Datenstrukturen des sendenden Werkzeuges auf die Datenstrukturen des empfangenden Werkzeuges. Es muss möglich sein, für jedes zu übertragende Datenobjekt ein passendes Datenobjekt im empfangenden Tool zu finden und zu nutzen. In diesem Beitrag wird ein Implementierungskonzept für den Datenaustausch mit dem standardisierten Format AutomationML vorgestellt. Das Konzept definiert verschiedene Zuordnungsstrategien, mit deren Hilfe AutomationML-Objekte auf spezifische Datenobjekte der gekoppelten Systeme abgebildet werden können, die damit helfen, das obige Problem zu lösen.

Wenn Systeme über AutomationML gekoppelt werden, muss jedes System einen entsprechenden Exporter bzw. Importer besitzen, der dessen spezifische Datenobjekte auf die standardisierten AutomationML-Objekte abbildet. Dieser Zuordnungs- und Transformationsprozess wird von AutomationML durch Rollenbibliotheken unterstützt. Rollen sind semantische Definitionen für AutomationML-Objekte und sorgen beim Datenaustausch dafür, dass die Objektdefinition von einem importierenden System korrekt verarbeitet wird. Hierzu definiert AutomationML Bibliotheken mit abstrakten Rollen. Spezifische System-Datenobjekte können aber oftmals auf der angebotenen Abstraktionsebene nicht ausreichend genau beschrieben werden. Aus diesem Grund werden für bestimmte Anwendungsdomänen detailliertere Rollenbibliotheken entwickelt. Eine eindeutige Zuordnung zu einem Datenobjekt des Zielsystems kann dadurch verbessert, aber nicht garantiert werden. In einigen Fällen benötigt der Importer daher eine Zuordnungsstrategie und eine Abbildungsvorschrift für die semantisch eindeutige Transformation des AutomationML-Objektes in das Zielsystem, um eine verlustfreie Datenübertragung zu gewährleisten.

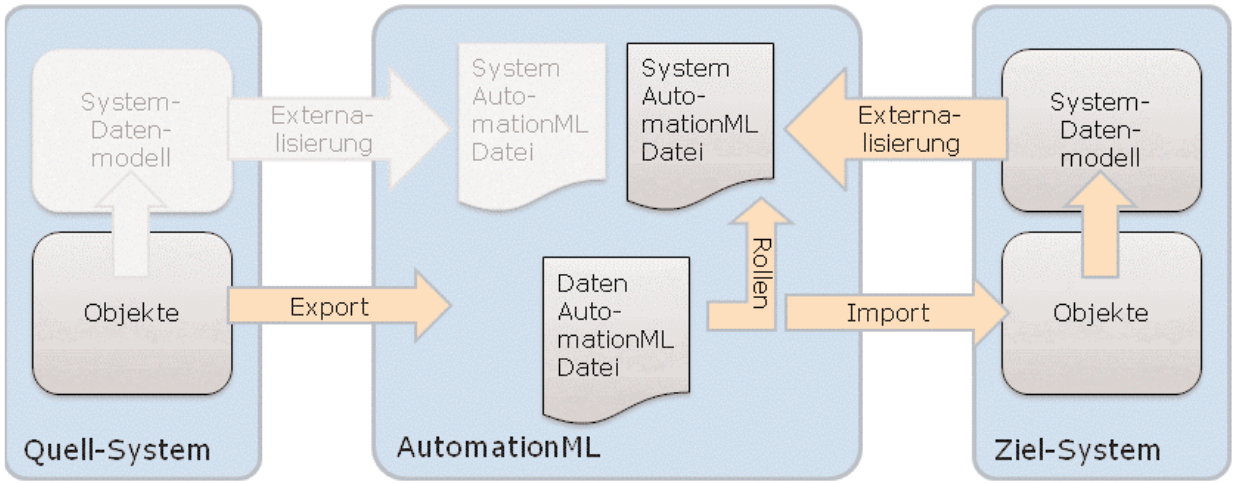


Bild 2: Zuordnung über Rollen

Externalisierung der Systemdatenmodelle

Wie wird ein Datenobjekt eines spezifischen Systems zu einem AutomationML-Objekt und umgekehrt? Eine Möglichkeit besteht in der Externalisierung der systemeigenen Klassen in Form einer AutomationML-SystemUnit-Klassenbibliothek. Von diesen systemeigenen Klassen leiten sich in der Regel die exportierbaren Datenobjekte, welche ausgetauscht werden sollen, ab. Ein Export- bzw. Importprozess müsste mit einer entsprechenden Externalisierung starten (Phase 1, vgl. Bild 1). Das Ergebnis ist eine systemspezifische AutomationML-Datei, die den austauschrelevanten Ausschnitt des Systemdatenmodells abbildet. Anschließend kann diese systembeschreibende AutomationML-Datei zukünftig bei jedem Datenimport und Datenexport (Phase 2) wieder Verwendung finden.

Datentransformation und Datenaustausch

Die Transformation eines System-Datenobjektes in ein AutomationML-Objekt beim Import und Export erfolgt nach unterschiedlichen Strategien, wobei die Zuordnungsstrategien eines Importers in der Regel komplexer sind als die eines Exporters, da der Importer gegebenfalls auf Objekte, für die es keine eins zu eins Entsprechungen gibt, interpretieren und auf das Systemobjektmodell abbilden muss. Sowohl Exporter als auch Importer nutzen das für ihr System externalisierte Systemdatenmodell und stellen Zuordnungen zwischen den Datenobjekten und den SystemUnit-Klassen her. Der Exporter realisiert dies für die zu exportierenden Datenobjekte und erzeugt anhand der zugeordneten SystemUnit-Klasse ein AutomationML-Objekt (ein CAEX – 'InternalElement'). Beim Importer verhalten sich die Dinge genau umgekehrt. Er stellt eine Zuordnung zwischen den vom Quellsystem gelieferten AutomationML-Objekten und den eigenen SystemUnit-Klassen her und erzeugt anhand der zugeordneten SystemUnit-Klasse ein Datenobjekt im Zielsystem.

Zuordnungsstrategien – Zuordnungen anhand von Rollen

Rollen werden dazu verwendet, die Semantik von AutomationML-Objekten zu definieren. Im konkreten Fall muss ein Exporter jedem erzeugten AutomationML-Objekt mindestens eine Rolle zuordnen. Die SystemUnit-Klassen des Quellsystems, die vom Exporter für die Generierung der AutomationML-Objekte verwendet werden, definieren die Menge der zulässigen Rollen für das jeweils generierte AutomationML-Objekt. Wenn ein Importer ein AutomationML-Objekt von einem Quellsystem erhält, interpretiert er die zugeordneten Rollen und versucht seinerseits nun unter den SystemUnit-Klassen des Zielsystems diejenigen zu finden, die diese Rollen unterstützen (Bild 2). Er kann diesen Schritt nur ausführen, wenn sowohl die SystemUnit-Klassen des Quellsystems als auch die SystemUnit-Klassen des Zielsystems dieselben Rollenbibliotheken benutzen bzw. Rollenbibliotheken mit demselben 'Stammbaum' (Rollen können in Vererbungsbeziehungen stehen). Findet der Importer mehr als eine passende SystemUnit-Klasse, benötigt er eine Strategie, um den Konflikt zu lösen. Eine Möglichkeit besteht darin, dass der Exporter bereits im Vorfeld bei der Zuordnung einer Rolle zu einem AutomationML-Objekt zusätzliche Bedingungen definiert. In diesen können bestimmte Werte oder Wertebereiche für spezielle Objekteigenschaften gefordert sein, die der Importer dann zur Interpretation nutzen kann. Wenn

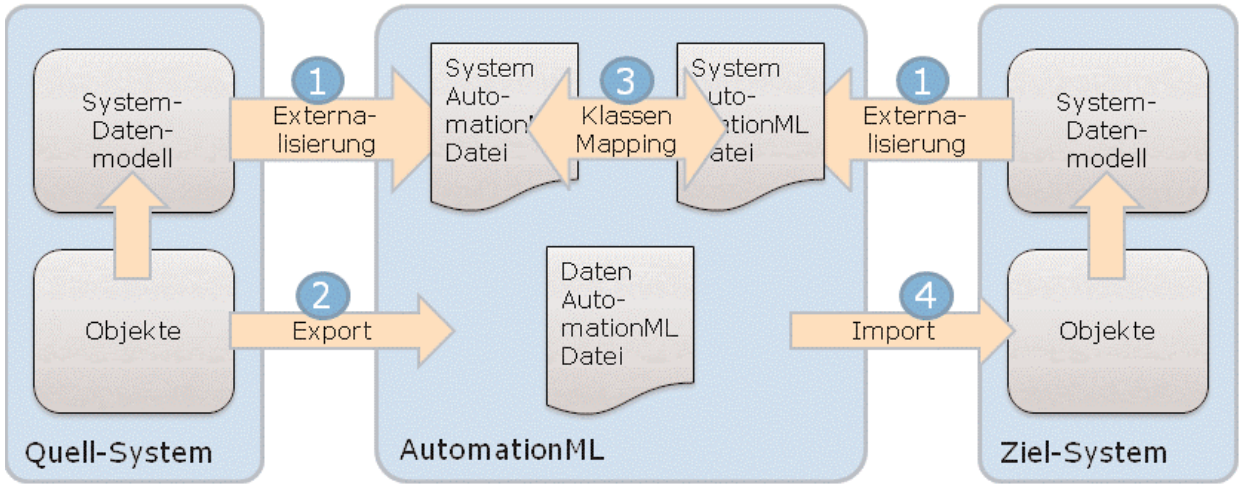


Bild 3: Zuordnung über Klassen

die Interpretation auch damit noch nicht möglich ist, müssen zusätzliche Regeln und Strategien definiert werden.

Zuordnung mithilfe von Klassen-Relationen

Bei der Zuordnung von Daten-Objekten des Quell- und Zielsystems mithilfe von Klassen-Relationen ist der erste Schritt wiederum die Externalisierung der AutomationML-System-Unit-Klassenbibliotheken aus beiden Systemen (1), vergleiche Bild 3. Im Anschluss erfolgt der Export der Daten aus dem Quellsystem in die Daten der AutomationML-Datei (2). Diese kann durch den Importer des Zielsystems eingelesen werden. Dabei sind dem Importer sowohl beide System-Unit-Klassenbibliotheken, als auch die Vorschriften, wie die einzelnen Klassen aufeinander zu mappen sind, bekannt. So kann zuerst ein Klassen-Mapping erfolgen (3) und in einem weiteren Schritt der Import der AutomationML-Daten inklusive der semantisch eindeutigen Zuordnung auf das eigene Objektmodell (4). Eine Erweiterung des Klassen-Mappings stellte die Zuordnung über Regeln dar. Dabei werden nicht nur die Klassen einander zugeordnet, sondern zusätzlich Regeln für z.B. Attributwerttransformationen erzeugt. Solche Regeln können in einer Transformationssprache wie beispielsweise XSLT formuliert werden.

Beispiel und Mappingwerkzeug

Ein typisches Anwendungsbeispiel für die Notwendigkeit eines Mappings ist die unterschiedliche Struktur in der Klassendefinition auf Basis unterschiedlicher Rollen in den Systemen, die Daten austauschen. In Bild 4 ist dies exemplarisch dargestellt. Hier baut sich im System A die Klassenstruktur entsprechend der konkreten Automatisierungsfunktionalität bzw. der Geräteklassifizierung, die in den Rollen abgebildet ist, auf. System B hingegen unterscheidet zwischen passiven und aktiven Objekten. Entsprechend des oben erläuterten Mappings von Rollen, müssen für einen Datenaustausch die Rollen 'Robot', 'Tool' und 'Fence' auf die in System B verwendeten Klassen abgebildet werden. Für die im Beispiel dargestellte Exportdatei 'ExportA.aml' heißt dies, dass die sechs exportierten Objekte auf fünf 'ActiveObjects' und ein 'PassiveObject' abgebildet werden. Zur Definition und Validierung solcher Zuordnungsstrategien wurde bei inpro ein prototypisches Werkzeug 'AutomationML Mapper' entwickelt. Mit die-

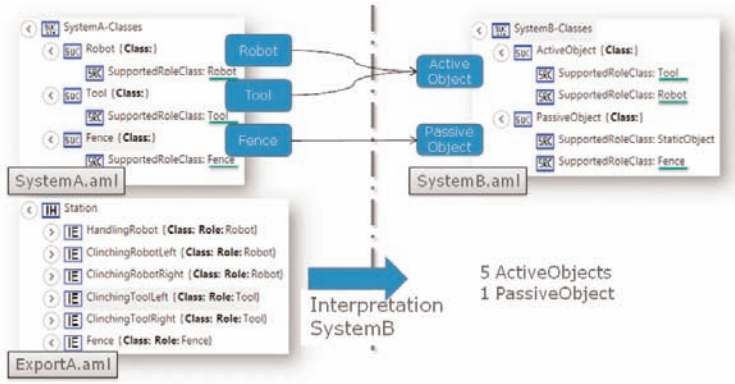


Bild 4: Beispiel Zuordnung über Rollen

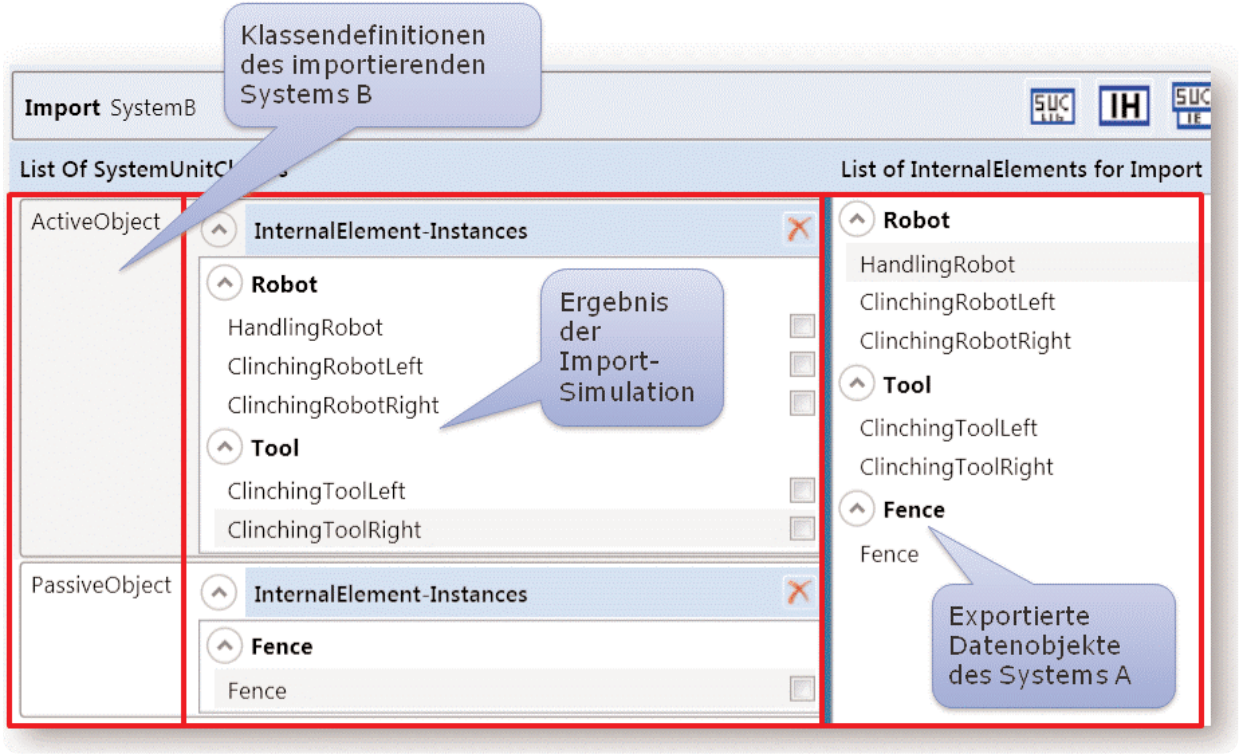


Bild 5: Oberfläche 'AutomationML Mapper'

sem ist es möglich, unterschiedliche Objektmodelle via AutomationML miteinander zu verknüpfen und so eine semantisch eindeutige Schnittstelle zwischen unterschiedlichen Werkzeugen zu erstellen. Bild 5 zeigt einen Ausschnitt der Werkzeugoberfläche. In diesem können die benötigten Zuordnungen bequem definiert und getestet werden.

Fazit

Der Austausch von Daten über eine syntaktisch eindeutige Schnittstelle ist ein erster bedeutender Schritt zur vollständigen Interoperabilität zwischen am Planungsprozess beteiligten Werkzeugen. Um das volle Potenzial dieser Schnittstellen auszuschöpfen, ist ein weiter Schritt notwendig, der die historisch gewachsenen Objektmodelle auch semantisch verknüpft. Um diese Herausforderung zu lösen, gab es in der Vergangenheit eine Reihe von Ansätzen, welche ein 'alles beschreibendes Weltmodell' nutzten, was sich in der Praxis oft als nicht tauglich erwiesen hat. In diesem Beitrag wurde eine Möglichkeit vorgestellt, die dieses Problem löst, indem es Vorteile des Austauschformates AutomationML mit einfach zu definierenden Zuordnungsstrategien kombiniert und somit einen sowohl syntaktisch als auch semantisch eindeutigen Datenaustausch ermöglicht. ■

www.automationml.org

Autor: Dr.-Ing. Lorenz Hundt, Projektingenieur Produktionssysteme und Informationsprozesse, inpro GmbH

Autor: Josef Prinz, Senior-Experte Digitale Fabrik, inpro Innovationsgesellschaft für fortgeschrittene Produktionssysteme mbH

AutomationML – Effizienz für den Engineeringprozess

Serie AutomationML Teil 5: Engineering-Effizienz für Antriebs- und Automatisierungslösungen

Bild 1: Zwölf typische Maschinenaufgaben

Bei der Entwicklung von Maschinen ist eine interdisziplinäre Zusammenarbeit gefordert: sowohl zwischen den Konstrukteuren, Elektronikern und Software-Entwicklern des Maschinen- oder Anlagenbauers als auch zwischen dem Auftraggeber und den Applikationsingenieuren der Komponentenlieferanten. Bisher mussten viele Informationen manuell zwischen den einzelnen Domänen ausgetauscht werden. Dabei besteht immer das Risiko, dass Wissen und Daten verloren gehen, zudem hat dieses Vorgehen einen enormen Einfluss auf die Entwicklungszeiten von Maschinen und Anlagen. Für ein durchgängiges Engineering von der Planungsphase über die Programmierung bis zur Betriebsphase kann AutomationML als Verbindung zwischen individuellen, anwender- und aufgabenorientierten SW-Werkzeugen für mehr Transparenz und Effizienz beim Austausch von Daten sorgen.

Als Systemlieferant für Antriebs- und Automatisierungslösungen hat sich Lenze zum Ziel gesetzt, das Engineering einer Maschine deutlich zu vereinfachen, um die Kosten und den Zeitaufwand zu reduzieren. Es gilt, einen optimalen Support über die gesamte Wertschöpfungskette herzustellen. Hierfür ist einerseits ein breites, auf die Bedürfnisse des Maschinenbaus abgestimmtes Produktportfolio erforderlich, das auch unterschiedliche Automatisierungstopologien unterstützt. Andererseits ist ein methodisches Vorgehen im Entwicklungsprozess nötig, das neben den richtigen Werkzeugen auch Standards beinhaltet. Diese Standards ermöglichen es, Know-how zur Verfügung zu stellen, die Komplexität zu verringern und In-

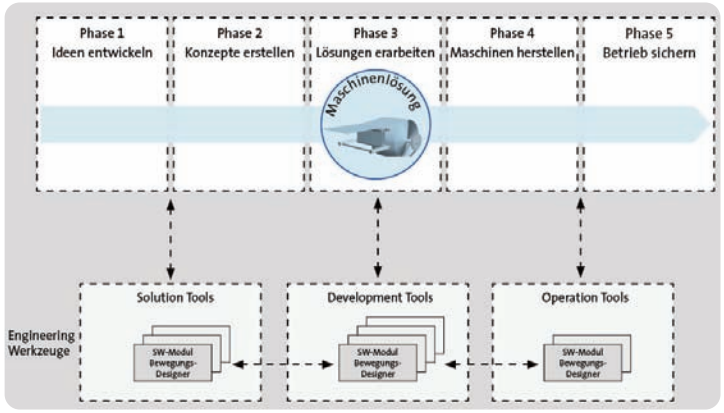


Bild 2: Fünf Phasen bei der Erstellung einer Maschinenlösung mit erforderlicher Tool-Unterstützung.

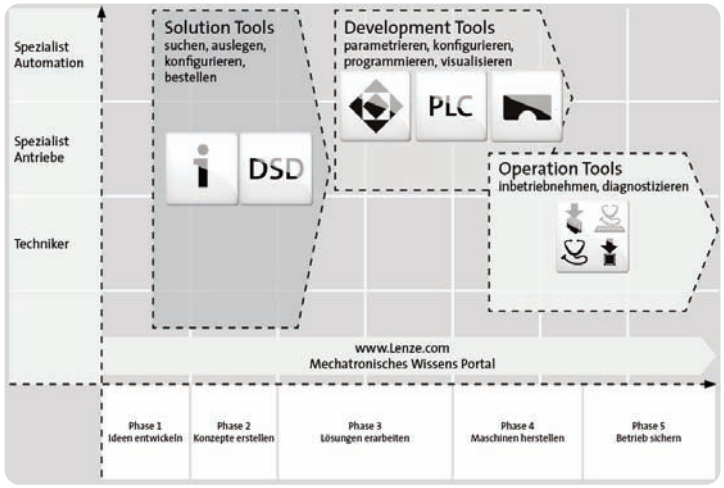


Bild 3: Engineering-Werkzeuge für die Erstellung von Antriebs- und Automatisierungslösungen

formationen wiederzuverwenden. Diesem Anforderungsprofil wird mit der Modularisierung von Maschinen und der damit verbundenen Bildung von mechatronischen Einheiten begegnet. Bild 1 zeigt zwölf typische Maschinenaufgaben, für die Standards definiert werden können, bestehend aus geeigneter Automatisierungstopologie, erforderlichen Antriebskomponenten und vor allem aus vorgedachter Applikations-Software. Zusammengefasst ergibt dies eine Maschinenlösung, an der sich der Maschinenbauer und der Systemlieferant innerhalb des Entstehungsprozesses orientieren können. Bei Betrachtung des Produktentstehungsprozesses im Maschinenbau können folgende fünf Phasen herausgebildet werden (Bild 2, oberer Teil):

1. Idee: Formulierung der Idee und Erstellung eines Grobkonzepts.
2. Konzepterstellung: Definition der Maschinenmodule und deren Funktion sowie Festlegung der grundlegenden Automatisierungsarchitektur.
3. Entwicklung: Ausarbeitung der Lösung, inklusive der Auslegung des Antriebsstrangs und der Auswahl der Antriebs- und Steuerungskomponenten. Auch die Softwaremodule werden vorbereitet.
4. Produktion: Bau der Maschine. Parallele bzw. zeitversetzte Fertigstellung der Maschinen-Software.
5. Betrieb: Inbetriebnahme und Betrieb der Maschine beim Endkunden. Die Maschinenverfügbarkeit muss gesichert werden. Wichtige Aufgaben sind hierbei die Zustandsüberwachung oder auch die (vorbeugende) Wartung.

Effizienz durch Kette von Engineering-Werkzeugen

Die Firma Lenze kann die Kunden über alle fünf Phasen des Entwicklungsprozesses begleiten. Wichtig dabei sind immer wieder neue und intelligente Engineering-Werkzeuge, die bei der Entwicklung unterstützen. Ein entscheidender Ansatzpunkt zur Verbesserung der Engineering-Produktivität ist die Beseitigung der noch in weiten Teilen existierenden Brüche bzw. Lücken in der gesamten Werkzeugkette. Eine Werkzeugkette soll den zuvor beschriebenen Lebenszyklus komplett abdecken. Allerdings ist es nicht zielführend, für den gesamten Prozess nur ein einziges Werkzeug vorzusehen. Wesentlich effektiver und sicherer ist es, wenn für die verschiedenen Projektierungsphasen maßgeschneiderte Tools zur Verfügung stehen. Tools, die auf die Aufgabenstellung, das Wissen und die Arbeitsweise des jeweiligen Projektbeteiligten (Konstrukteur, Steuerungsprogrammierer, Visualisierungsprogrammierer, Wartungstechniker) zugeschnitten sind und ihn damit optimal und effizient in seiner Arbeit unterstützen. Ist der Engineering-Prozess auf mehrere Werkzeuge aufgeteilt, ist eine durchgängige Datenhaltung si-

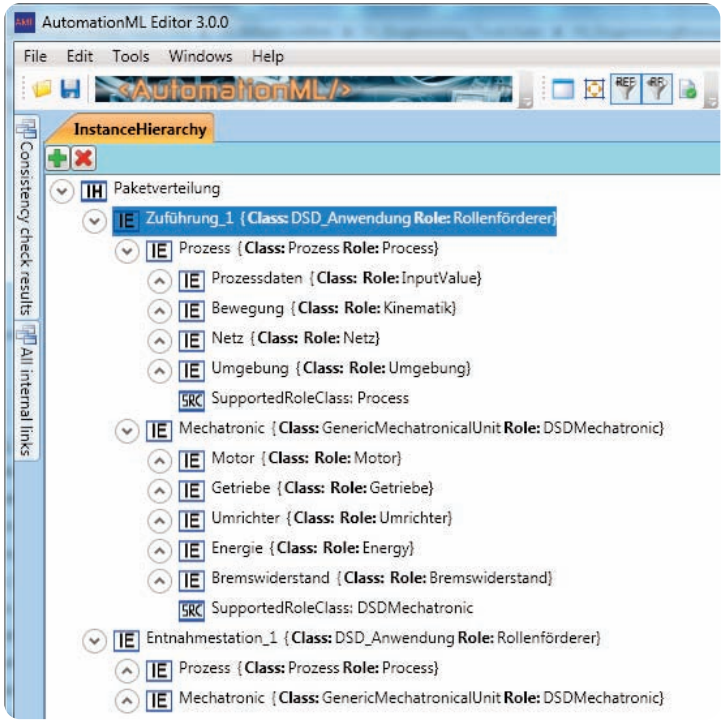


Bild 4: Beispielhafte AML-Struktur für einen Rollenförderer

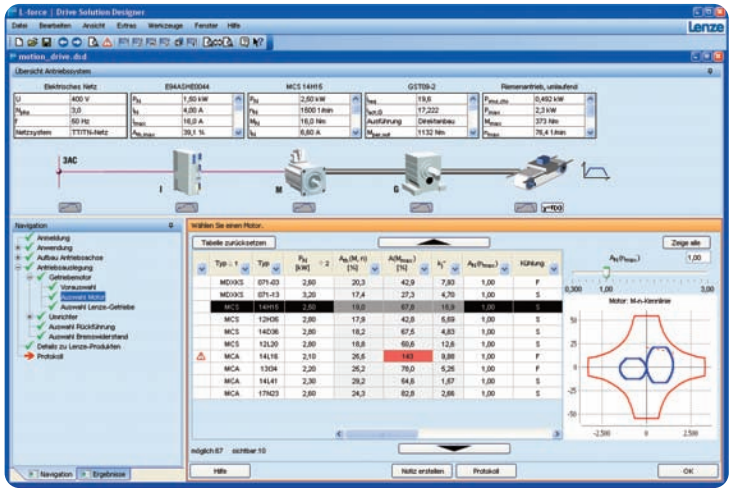


Bild 5: Antriebsauslegung mit DSD

cherzustellen. AutomationML bietet hierzu die Möglichkeit, die Übergänge von einem Werkzeug zum nächsten nahtlos zu gestalten. Die Mehrfacheingabe von Daten entfällt und die Zusammenarbeit zwischen den Konstrukteuren (Mechanik und Elektronik) und den Softwareentwicklern vereinfacht sich. Gerade hier ist der Informationsaustausch besonders wichtig, da diese Domänen in der Regel nicht nur organisatorisch getrennt sind, sondern auch eine individuelle Sichtweise auf das Projekt haben. Bild 3 zeigt in einer Übersicht, wie eine solche durchgängige Werkzeug-Landschaft aussehen kann und welche Zielgruppen diese Werkzeuge einsetzen.

Anwendungsbeispiel DSD

Ein beispielhaftes Element einer solchen Werkzeuglandschaft bildet das Antriebsauslegungswerkzeug Drive Solution Designer (DSD). Durch ihn werden dem Konstrukteur die physikalischen Zusammenhänge und das Lösungswissen für die vordefinierten Maschinenmodule zur Verfügung gestellt. Anhand dieser Module kann der Anwender deren Bewegung beschreiben und anschließend den kompletten Antriebsstrang auslegen, die richtigen Antriebskomponenten auswählen und die Lösung optimieren. Dabei ist es möglich, zahlreiche physikalische Parameter zu ermitteln bzw. deren Einhaltung zu prüfen. Dazu gehören u.a. die Kombinierbarkeit der einzelnen Produkte, deren thermische und mechanische Auslastung, die Lebensdauer und der Energieverbrauch der Antriebskomponenten. Der DSD wurde um eine AutomationML-Schnittstelle ergänzt, das heißt, er kann strukturierte Daten zu einer Anwendung, z.B. einem Rollenförderer, importieren. Es handelt sich dabei sowohl um Prozessdaten, wie Durchmesser, Trägheitsmomente, Massen, Reibung, etc. als auch um Bewegungsinformationen, Netz- und Umgebungsbedingungen. Das sind Informationen, die in der Planungsphase beispielsweise in CAD-Werkzeugen bereits zuvor eingegeben bzw. ermittelt wurden. Nach dem Import werden diese Werte im DSD direkt verwendet, sodass der Anwender weniger Zeit bei der Ermittlung und Eingabe der Daten benötigt und direkt die Auslegung durchführen kann. Das Ergebnis der Auslegung (benötigte Antriebsprodukte, Einstellungen der Umrichter, Energiebedarf, etc.) kann er dann durch die Export-Funktion in das AML-Format zurückschreiben und somit seinen Beitrag am gesamten Engineeringprozess leisten. Ist die Auslegung abgeschlossen, liegen die erforderlichen konkreten Antriebskomponenten vor und es kann mit der eigentlichen Projektierung, also der Ausarbeitung der Bewegungs- und Ablaufprogramme sowie der Visualisierung, fortgefahren werden. Hier kommt unter anderem der PLC Designer zum Einsatz, mit dem die Programmerstellung und Inbetriebnahme nach IEC61131-3 erfolgt. Er beinhaltet sechs Editoren, Debugger und Monitoringfunktionen und basiert auf Codesys3. Der Clou: Mit Hilfe von Standard-Software-Bausteinen für die unterschiedlichen Maschinenmodule kann die Produktivität der Programmierer gesteigert werden. Eine zukünftige AutomationML- Schnittstelle bietet die Möglichkeit, die bereits zuvor mit dem DSD erstellten Informationen für das Maschinenmodul im PLC Designer noch schneller und vollständiger zur Verfügung zu stellen.

Fazit

Mit AutomationML lässt sich Durchgängigkeit und Effizienz bei gleichzeitiger Modularisierung für den Entwicklungsprozess einer Maschine oder Anlage erreichen. Mechanik-, Elektronik- und Software-Informationen können für eine Antriebs- und Automatisierungslösung gebündelt werden und stehen der individuellen Toolkette des Maschinenbauers und des Systemlieferanten zur Verfügung. Dies verbessert neben der Effizienz und Qualität des Engineerings auch die Wissenssicherung im Maschinen- und Anlagenbau. ■

www.automationml.org

Autor: Olaf Götz, Produktmanager Engineering Tools, Lenze Hameln

AutomationML: Umgebung für das Multi-User-Engineering

Serie AutomationML Teil 6: Digitaler Engineering-Tisch (DigET) kombiniert OPC-UA und AutomationML

Bild 1: Logo des Projekts Digitaler Engineering-Tisch (DigET)

Der Digitale Engineering-Tisch (Bild 1) bietet Planern von Produktionsanlagen die Möglichkeit, Planungsszenarien für alle Disziplinen konsistent zu beleuchten und zudem intuitiv mit den IT-Werkzeugen zu interagieren (beispielsweise per Gesten). Er kombiniert hierfür die Standards AutomationML und OPC-UA mit Assistenzmechanismen und einer interaktiven Umgebung.

Der Prozess der Fabrik- und Anlagenplanung ist komplex und besteht aus vielen, meist sequenziellen Planungsschritten mit vielen verschiedenen Beteiligten, z.B. Fabrik- und Gebäudeplaner, Prozess- und Anlagenplaner. Diese Disziplinen arbeiten heute mit vielen, oft nicht integrierten IT-Werkzeugen, deren Interoperabilität verbessert werden muss. Die Ingenieure und Planer arbeiten größtenteils räumlich und technisch voneinander getrennt und setzen konventionelle Medien ein, z.B. Pläne auf Papier. Obwohl die Disziplinen das gleiche Planungsobjekt, das Produktionssystem, bearbeiten, werden zwischen den Tools Informationen nur teilweise oder gar nicht ausgetauscht. Die Personen stimmen sich meist – falls und wenn nötig – mündlich ab. Dieser Bruch macht den interdisziplinären Abstimmungsprozess ineffizient, aufwändig und fehleranfällig.

Ziele

Um Kommunikation und Planung an einem gemeinsamen Medium zu verbessern, hat das Fraunhofer IOSB den Digitalen Engineering-Tisch (DigET) [1] entwickelt (Bild 2). Am DigET können die Planer mithilfe einer interaktiven Umgebung Planungsszenarien für alle Disziplinen konsistent beleuchten und zudem intuitiv mit den IT-Werkzeugen interagieren (siehe [2]). Die Interaktion mit dem DigET über Gesten ist dabei so einfach, dass sich die Planer voll auf den Planungsgegenstand konzen-

Bild 2: DigET – interaktive Arbeitsumgebung für das Multi-User-Engineering

trieren können und die Interaktion mit den IT-Werkzeugen in den Hintergrund tritt. Er kombiniert hierfür das durchgängige und standardisierte Datenformat AutomationML mit Assistenzmechanismen und einer interaktiven Umgebung. So wird die domänenübergreifende Zusammenarbeit gefördert und der elektronische Informationsaustausch- und Änderungsprozess wird mithilfe der verbesserten menschlichen Kollaboration unterstützt. Verbesserungen beim Einsatz des DigET sind:

- Konsistente Daten für alle an einer Fabrik- und Anlagenplanung beteiligten Ingenieure,
- Verbesserter Datenaustausch zwischen verschiedenen IT-Tools bei Anlagenherstellern und -betreibern,
- Verbesserte Zusammenarbeit der einzelnen Disziplinen / Gewerke,
- Benutzer-Assistenz im Engineering,
- Erleichterung von Produktionsumstellungen und -inbetriebnahmen,
- Durchgängige computerunterstützte Planung von Fertigungszellen oder -linien und
- Lückenschluss zwischen individuellen domänenspezifischen Planungsschritten auf Datenebene und der interpersonellen Kommunikation.

Angewendet werden kann der DigET beispielsweise zur Team-basierten Anlagen-Umplanung. Die Umplanung ist ein besonders herausforderndes Szenario, weil hier neue Anlagen in bestehende Linien oder Hallen eingefügt werden müssen. Bei Veränderungen in den Planungsständen aufgrund von Umplanungen zeigt sich die Stärke der neuartigen Umgebung. Der DigET dient dabei als Arbeitsplatz zur Diskussion von Änderungen und unterstützt bei der Lösung der dabei entstehenden/auftretenden Konflikte. Er ist aber nicht der Arbeitsplatz der Disziplinen zur Planung, sondern stellt einen gemeinsamen Abstimmungsraum dar. Jede Disziplin arbeitet dabei weiterhin mit dem Tool ihrer Wahl, der DigET unterstützt bei der effektiven Koordination und Fusionierung aller Änderungen auf dem Datenbestand.

Hardware

Der DigET (Bild 3) ist ein Multi-Display-System mit verteilten Displays (horizontal und vertikal angeordnet) sowie verschiedenen mobilen Endgeräten, die in die Umgebung eingebunden werden. Die komplette Umgebung kann mithilfe Display-übergreifender Handgesten-Interaktion im 3D-Arbeitsraum bedient werden. Dabei werden Tablet-PCs und SmartPhones als mobile Arbeitsplätze eingesetzt und durch persönliche Identifikationsmöglichkeiten über Marken-Tracking mit dem Tisch gekoppelt.

Software

Das Multi-User-System DigET setzt auf Standards. Als integriertes und durchgängiges Standard-Datenaustauschformat zur Anlagenbeschreibung wird AutomationML eingesetzt, wobei die in der Umgebung verwendeten Tools lediglich eine AutomationML-Schnittstelle bereitstellen müssen. AutomationML befindet sich aktuell auf dem Weg der IEC-Normierung und wird bereits von Toolherstellern als Austauschformat unterstützt. Als Standard zur Kommunikation und dem Datenaustausch zwischen allen Komponenten wird OPC-UA verwendet. An diese Kommunikation können sich potenzielle Nutzer mithilfe eines OPC-UA-Clients einfach 'andocken'. Dabei wird AutomationML in den Kommunikationsstandard OPC-UA als Informationsmodell integriert. Dies schafft die notwendige Transparenz. Der DigET setzt keine einheitliche Software voraus, sondern ermöglicht es den Nutzern, die für sie beste Software einzusetzen und diese mit dem Tisch zu koppeln. Jeder Nutzer behält seinen eigenen Datenbestand und seine gewerkespezifische Sicht. So können bspw. Elektronik- und Mechanikplanung sich über die gemeinsame Umgebung verständigen, behalten aber weiterhin ihre proprietären Datenbestände. Durch das gemeinsame Modell im Format AutomationML können die Nutzer Änderungsvorschläge an den Tisch melden, die dann verarbeitet werden. Die Änderungspropagation erfolgt von der dafür entwickelten Middleware (siehe [3]) in Form eines OPC-UA-Servers mit speziellen Methoden. Änderungen werden also an die entsprechenden Beteiligten weitergeleitet. Die Bedienung der Tools durch die Gesten-Interaktion

Bild 3: Hardware-Komponenten der DigET-Umgebung und deren Anordnung

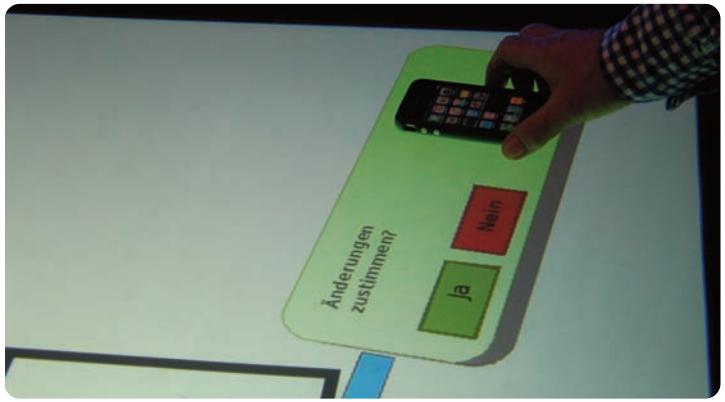


Bild 4: Interaktion der Nutzer mit dem DigET, z.B. über Smartphone

(Bild 4) erfolgt am DigET über ein transparentes Overlay, das zur Kommunikation mit der entsprechenden Standard-Software dient. Daher sind für die Gesten-Bedienung keine zusätzlichen Schnittstellen in den Tools nötig.

Interaktion/Assistenz

In unserem täglichen Leben werden wir durch entsprechende technische Systeme und intelligente Umgebungen bereits unterstützt. Warum also nicht auch bei komplexen Prozessen wie der Produktionsplanung? Der DigET unterstützt als IT-System die Kooperation der Beteiligten und führt und begleitet die zugehörigen Entscheidungsprozesse. Für die direkte Multi-Display-Interaktion wurden verschiedene Handgesten definiert, die die Multi-User-Interaktion im Team unterstützen. Änderungen werden als Vorschlag einer Disziplin eingebracht, andere Disziplinen können Konflikt-behafteten Veränderungen zustimmen oder diese ablehnen. Mithilfe ergebnisorientierter Konfliktlösungsunterstützung in Form entsprechender Assistenzfunktionalitäten in der Umgebung werden konfliktfreie Änderungen übernommen, Konflikte aufgelöst und Abstimmungsprozesse begleitet. Änderungsprozesse können so überwacht und dokumentiert werden. So können gemeinsame, koinzidente Modelländerungen vorgenommen werden. Darüber hinaus wurden verschiedene Assistenzmechanismen zur Verbesserung der interpersonellen und interdisziplinären Zusammenarbeit in die Umgebung integriert.

Vorteile

Die durchgängige Kommunikation und Datenhaltung über Nutzer und Tools hinweg, unter Verwendung und Kombination der Standards AutomationML und OPC-UA, ermöglicht die Adaption und Integration weiterer Tools und nutzt die Vorteile der einzelnen Standards optimal aus. Durch den DigET können daher verschiedene Vorteile geltend gemacht werden: Reibungsverluste, Missverständnisse und der Abstimmungsaufwand zwischen den Beteiligten werden reduziert. Ebenso wird die Eigendynamik in Gruppen implizit genutzt und durch die Systeme entsprechend unterstützt. Die enge Zusammenarbeit und verbesserten Kommunikationsmöglichkeiten erhöhen das gegenseitige Verständnis für die anderen Disziplinen, verbessern den Systemüberblick für die einzelnen Beteiligten und die Arbeit im Team steigert die Motivation. Durch mögliche Dokumentations- und Abstimmungsmechanismen sowie die Archivierbarkeit werden Ergebnisse besser nachvollziehbar und der Entwicklungszyklus transparenter. Darüber hinaus wird im besten Fall die Planungsqualität verbessert. ■

Literatur

[1] Fraunhofer IOSB: DigET, www.dig-et.de, Stand 15.05.2012.
[2] Miriam Schleipen, Thomas Bader: A concept for interactive assistant systems for multi-user engineering based on AutomationML. Proceedings of CAPE Conference 2010, Edinburgh, 13.-14.4.2010, Paper 014.
[3] Miriam Schleipen, Manfred Schenk: Intelligent environment for mechatronic, cross-discipline plant engineering. IEEE conference on Emerging Technologies and Factory Automation ETFA 2011, September 5-9, 2011, Toulouse, France, 2011.

www.automationml.org


Autorin: Dr.-Ing. Miriam Schleipen, Gruppenleiterin Leitsysteme und Anlagenmodellierung, Fraunhofer IOSB


Autor: Manfred Schenk, Wissenschaftlicher Mitarbeiter, Fraunhofer IOSB


Autor: Sebastian Maier, Wissenschaftlicher Mitarbeiter, Fraunhofer IOSB


Autorin: Dr.-Ing. Elisabeth Peinsipp-Byma, Abteilungsleiterin Interaktive Analyse und Diagnose, Fraunhofer IOSB


Autor: Jan Hendrik Hammer, Wissenschaftlicher Mitarbeiter, Karlsruher Institut für Technologie (KIT)

AutomationML – Kommunikation

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

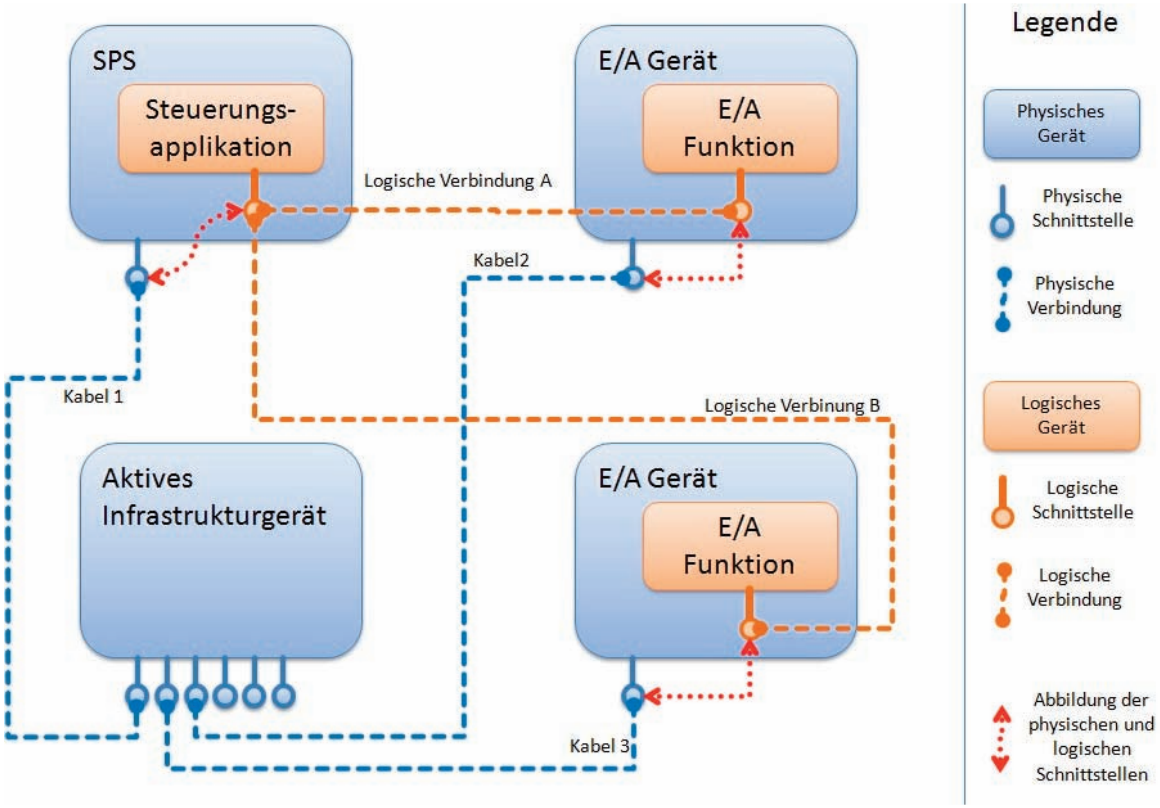


Bild: AutomationML e.V. c/o IAF

Bild 1: Kommunikationsnetzwerk mit logischen und physischen Geräten und Verbindungen

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 ausgetausch-

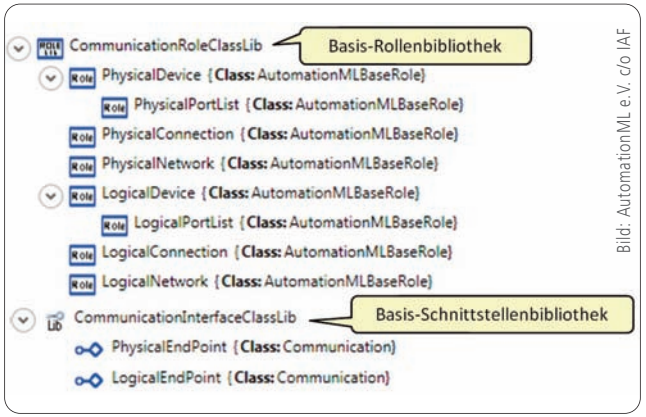


Bild 2: Rollen- und Interface-Klassen für die Kommunikationsmodellierung

ten 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 Eigenschaften (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.

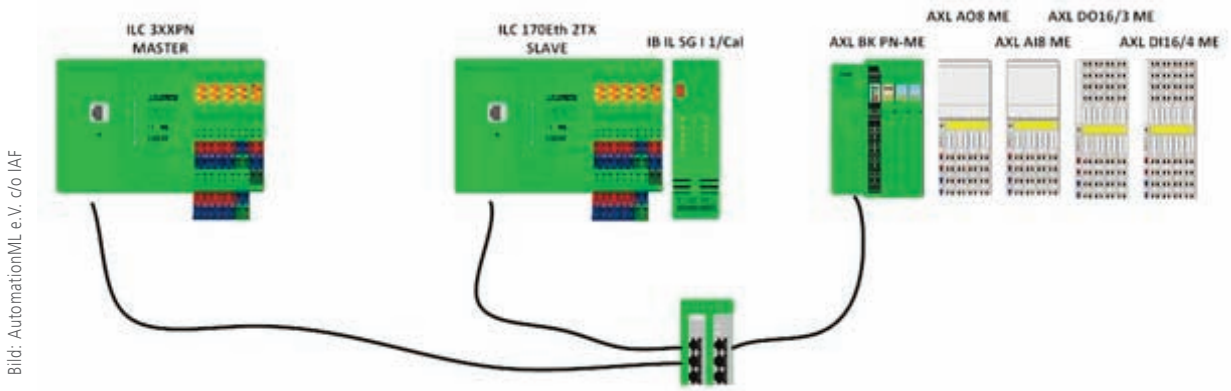


Bild 3: Kommunikationssystembeispiel

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 Commu-

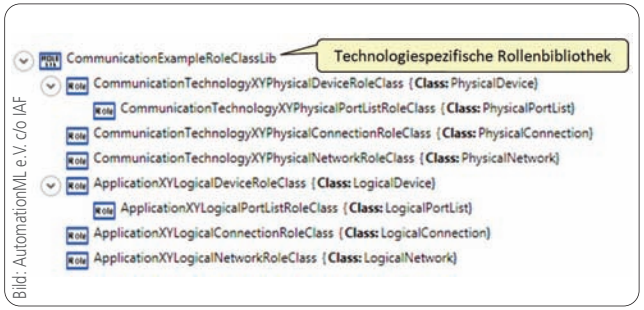


Bild 4: Technologieabhängige Rollenklassen für das Anwendungsbeispiel

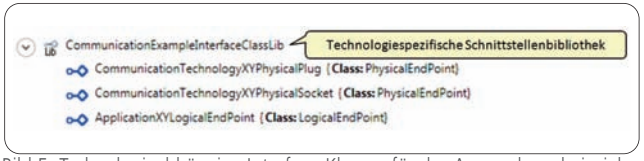


Bild 5: Technologieabhängige Interface-Klassen für das Anwendungsbeispiel

nicationTechnologyXY 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 <SystemUnitClassLib>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 eingela-

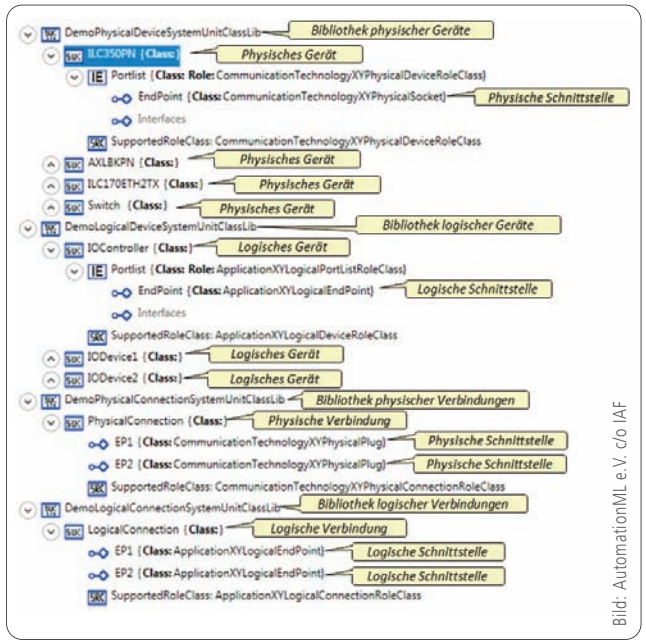


Bild: AutomationML e.V. c/o IAF

Bild 6: <SystemUnitClassLib>s für das Anwendungsbeispiel

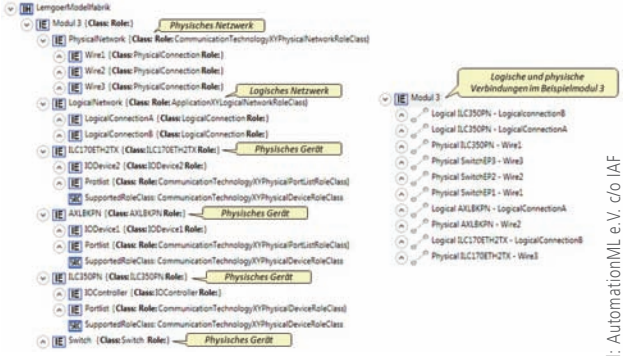


Bild: AutomationML e.V. c/o IAF

Bild 7: Netzwerkbeschreibung für das Anwendungsbeispiel

den, 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

Literatur

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

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

AutomationML – Integration MES-Wissen

Serie AutomationML Teil 8: Integration domänenspezifischen MES-Wissens in AutomationML

Bild 1: MES-Systeme sind fester Bestandteil moderner Produktionen. AutomationML bietet die Möglichkeit, MES-Wissen zu integrieren.

In der Produktionsanlagenplanung sind das Wissen und die damit erzeugten Informationen das wertvollste Gut. Dabei kann zwischen domänenunabhängigem Wissen und domänenspezifischem Wissen unterschieden werden. Domänenspezifisches Wissen ist beispielsweise MES-spezifisches Wissen. Domänenunabhängig kann Wissen z.B. im XML-basierten Datenaustauschformat AutomationML abgelegt werden. Beide Ansätze zur Modellierung und Formalisierung von Wissen können verbunden werden, indem etwa MES-spezifisches Wissen mithilfe des Rollenkonzepts in AutomationML integriert wird.

MES (Manufacturing Execution Systems) [VDI5600-1] sind IT-/Software-Systeme für das Fertigungsmanagement mit direktem Zugang zur Steuerungsebene, z.B. zu den SPSen. Sie überwachen den aktuellen Produktionszustand und aggregieren Informationen über diesen beispielsweise in sogenannten Kennzahlen (engl. KPI in sogenannte Kennzahlen (engl. KPI (Key Performance Indicator), siehe [ISO22400-2]). Für MES als überwachendes und steuerndes System ist es wichtig, Informationen aus verschiedenen heterogenen Quellen zu sammeln und zu fusionieren. Dabei wäre ein einheitliches Datenaustauschformat statt der in der Praxis anzutreffenden großen Anzahl an unterschiedlichen Schnittstellen ideal. Ontologien [Onto] versuchen die Verwendung von domänenspezifischen Terminologien zu vereinfachen und zu vereinheitlichen. Sie integrieren Semantik in die modellierten Daten und formalisieren Zusammenhänge zwischen den definierten Konzepten, sodass diese auch maschinenbearbeitbar und -auswertbar werden. Die MES-Ontologie [VDI5600-3] beschreibt eine einheitliche Terminologie in einer Domäne, nämlich spezifisch für Daten und Informationen, die zwischen MES und Maschinen-/Anlagensteuerungsebene ausgetauscht werden. Firmenspezifische Daten und Informationen werden auf die einheitliche Terminologie abgebildet (siehe [ETFASch]). Die MES-Ontologie wurde zwischen 2008 und 2010 vom VDI GPP-Fachausschuss 'Manufacturing Execution Systems (MES) – Arbeitsgruppe MES-Logische Schnittstellen' erarbeitet. Innerhalb der

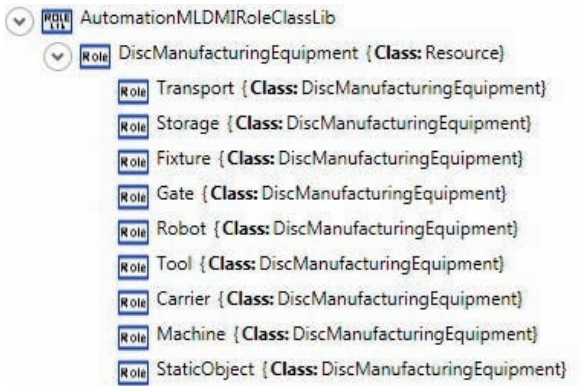


Bild: IEC 62714-2

Bild 2: AutomationMLDMIRoleClassLib

Ontologie, die sowohl in Textform als auch in der Web Ontology Language (OWL) vorliegt, wird unterschieden zwischen Datenpunkten und Ordnern. Für jeden Datenpunkt wird festgelegt, ob es sich um einen mandatorischen oder einen optionalen Datenpunkt handelt. Ebenso werden die Datenformate vorgegeben.

Domänenunabhängiges Wissen in AutomationML

AutomationML bietet als offenes und in der Standardisierung befindliches Datenaustauschformat (IEC62714) die Möglichkeit, semantische Informationen in den Anlagenmodellen zu hinterlegen. Dies kann genutzt werden, um einheitliche Terminologien und Standards aus spezifischen Domänen, wie z.B. MES, zu integrieren. Jedes Objekt innerhalb des AutomationML-Modells [AML2] übernimmt innerhalb seiner Umgebung eine Funktion und spielt dabei eine oder mehrere 'Rollen'. Diese Rollen werden mithilfe des AutomationML-Rollenkonzepts im Dachdatenformat CAEX [CAEX] umgesetzt. Das CAEX-Bibliothekskonzept unterscheidet drei Bibliotheksarten: die System-UnitClassLibraries, die InterfaceClassLibraries und die RoleClassLibraries. Darüber hinaus existiert die konkrete Anlagenhierarchie, die in der InstanceHier-archy repräsentiert wird. Eine Rolle beschreibt unabhängig von einer konkreten technischen Realisierung die Funktionen der physikalischen oder logischen Anlagenobjekte. Die konkrete technische Implementierung wird in der SystemUnitClass oder in InternalElements beschrieben. Die Rolle/RoleClass stellt also eine Möglichkeit dar, die Bedeutung eines Objekts abstrakt und herstellerunabhängig zu beschreiben. Die Definition bzw. Typisierung der Rollen erfolgt hierarchisch in RoleClassLibraries. Durch die Zuordnung einer Rollenklasse zu einem Objekt in der InstanceHierarchy des CAEX-Datenmodells werden diesem Objekt seine grundlegenden Funktionen und Anforderungen zugeteilt. Für AutomationML werden im Part 2 (IEC62714-2) der Spezifikation verschiedene Rollenklassenbibliotheken definiert. Diese können unterschieden werden in normative Rollenklassenbibliotheken, informative Rollenklassenbibliotheken sowie nutzerdefinierte Rollenklassenbibliotheken. Normativ existieren eine Basis-Rollenklassenbibliothek (AutomationMLBaseRoleClassLib), eine Rollenklassenbibliothek für die diskrete Fertigungsindustrie (discrete manufacturing industry – AutomationMLDMIRoleClassLib, siehe Bild 2), eine Rollenklassenbibliothek für die kontinuierliche Fertigungsindustrie (continuous manufacturing industry – AutomationMLCMIRoleClassLib), eine Rollenklassenbibliothek für die Chargen-verarbeitende Fertigungsindustrie (batch manufacturing industry – AutomationMLBMIRoleClassLib) sowie eine Rollenklassenbibliothek für Steuerungssysteme (control systems – AutomationMLCSRoleClassLib). Informativ ist die erweiterte Rollenklassenbibliothek (AutomationMLExtendedRoleClassLibrary) gegeben. Sie beschreibt mögliche Erweiterungen der normativen Rollenklassenbibliotheken. Beispielsweise werden für hierarchische Strukturen Rollen (z.B. Site, Area, ProductionLine) definiert, deren Definition an die

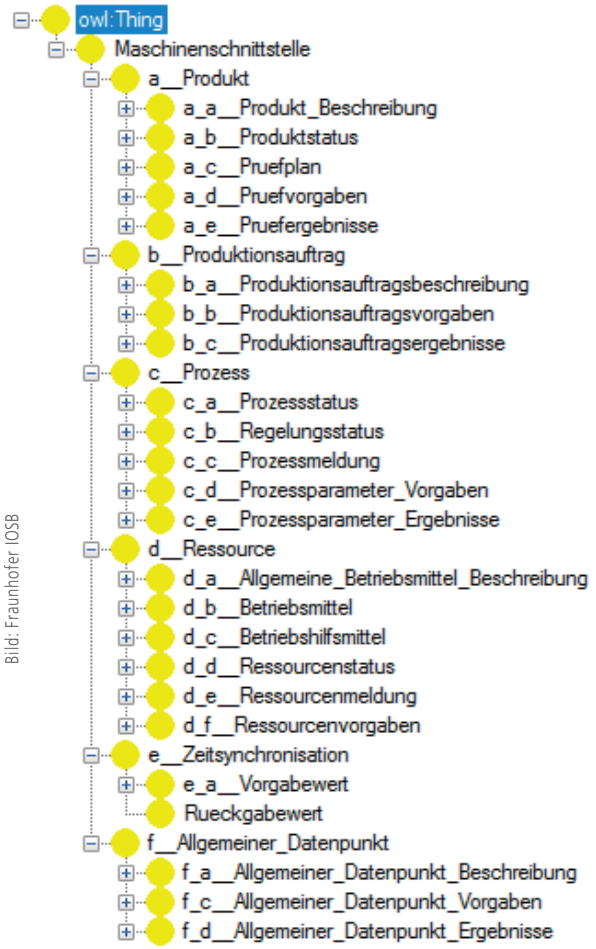


Bild: Fraunhofer IOSB

Bild 3: MES-Ontologie (oberste Hierarchieebene) [VDI5600-3]

Literatur

[AML] Miriam Schleipen, Rainer Drath: Three-View-Concept for modeling process or manufacturing plants with AutomationML. 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). 22.-25.9.2009, Palma de Mallorca, Spain.

[AML2] AutomationML. http://www.automationml.org/, Juni 2013.

[Automation] Miriam Schleipen, Olaf Sauer, Lidmilla Fuskova: MES-Ontologie – Semantische Schnittstelle zwischen Maschine und MES. Automation 2010, Baden-Baden.

[CAEX] IEC 62424. Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools, 2008.

[CIRPSch] Miriam Schleipen, Olaf Sauer, Lidmilla Fuskova, 'Logical interface between MES and machine – semantic integration by means of ontologies', Proceedings of: CIRP ICME '10 – 7th CIRP International Conference on INTELLIGENT COMPUTATION IN MANUFACTURING ENGINEERING, Innovative and Cognitive Production Technology and Systems, 978-88-95028-65-1. 23 – 25 June 2010, Capri (Gulf of Naples), Italy, 2010.]

[ETFASch] Miriam Schleipen, Dirk Gutting, Franziska Sauerwein: Domain dependant matching of MES knowledge and domain independent mapping of AutomationML models. IEEE conference on Emerging Technologies and Factory Automation ETFA 2012, September 17-21, Krakow, Poland, 2012.

[ISO22400-2] DRAFT INTERNATIONAL STANDARD ISO/DIS 22400-2, ISO/TC 184/SC 5 Secretariat: ANSI, Automation systems and integration – Key performance indicators for manufacturing operations management – Part 2: Definitions and descriptions.

[Onto] Mike Uschold, Michael Grüninger, 'Ontologies: Principles, Methods and Applications', Knowledge Engineering Review 11(2) (1996), pp. 93-155, 1996.

[SemMES] Christoph Thomalla, Miriam Schleipen, Olaf Sauer: Standardisierte semantische Schnittstelle MES - Maschinenebene (SemMES). Abschlussbericht. Förderkennzeichen: 01FS10013. Standardized semantic interface MES – machine level (SemMES). Report. Elektronische Publikation, Fraunhofer Publica, http://publica.fraunhofer.de/dokumente/N-226473.html, 2013.

[VDI5600-1] VDI 5600-1, Manufacturing Execution Systems (MES), Beuth Verlag, 2007.

[VDI5600-3] VDI 5600-3, Manufacturing Execution Systems (MES) – Logische Schnittstelle Maschinen- und Anlagensteuerung, Beuth Verlag, 2013.

Spezifikation der ISA95 (IEC62264-1:2003) angelehnt ist. Darüber hinaus gibt es die Möglichkeit der Integration nutzerspezifischer Inhalte, indem neue Bibliotheken von Nutzern definiert werden, die die bestehenden Basis-Bibliotheken erweitern. Diese nutzerspezifischen Rollen müssen von den Basisrollen abgeleitet werden. So wird sichergestellt, dass diese auf einen definierten Ursprung zurückgeführt werden können. Beispiele für nutzerspezifische Rollenklassenbibliotheken könnten sein: branchenspezifische Bibliotheken (z.B. Lebensmittel), domänenspezifische Bibliotheken (z.B. MES), organisationsspezifische Bibliotheken (z.B. rotes Buch des VDMA und VDW) oder auch firmenspezifische oder toolspezifische Bibliotheken.

Integration domänenspezifischen Wissens in AutomationML

Die Möglichkeit der Integration domänenspezifischen Wissens in AutomationML [AML] soll nun anhand der VDI 5600-Blatt 3 [Automation] gezeigt werden. Diese Integration birgt den Vorteil, dass die dort bereitgestellten Terminologien und das Vokabular, inklusive deren Definition und Vererbungsrelationen, in AutomationML genutzt werden können. Dabei werden die in der Ontologie hinterlegten Begrifflichkeiten und Definitionen als Rollenbibliothek in AutomationML verwendet. So kann die Semantik der Domäne MES direkt innerhalb von AutomationML eingesetzt werden. Eine solche Integration erhöht die Akzeptanz und baut die Anfangsbarrieren ab. Die Integration kann manuell oder basierend auf den XML-Datenformaten und Strukturen von OWL und AutomationML automatisch (siehe beispielsweise Projekt SemMES [SemMES]) mit manueller Nachbearbeitung erfolgen. Wichtig hierbei sind vor allem die Möglichkeiten, die durch die Integration der Definitionen für die einzelnen Rollen entstehen. Auf oberster Organisationsebene (siehe Bild 3) der MES-Ontologie [CIRPSch] unterteilen sich die Informationen in Produkt, Prozess, Ressource, Produktionsauftrag, Zeitsynchronisation und einem allgemeinen Datenpunkt. Dies passt ideal zu den erweiterten Konzepten in AutomationML, die ebenfalls zwischen Prozess, Produkt und Ressource unterscheiden. Daher bietet sich eine Integration der VDI 5600-3 in AutomationML als eigene Rollenklassenbibliothek an. ■

www.automationml.org

Autorin: Dr.-Ing. Miriam Schleipen, Gruppenleiterin Leitsysteme und Anlagenmodellierung, Fraunhofer IOSB

AutomationML: Verhältnismäßig gut!

Serie AutomationML Teil 9: Eignet sich AutomationML als neutrale Sprache zur Beschreibung von Verhalten?

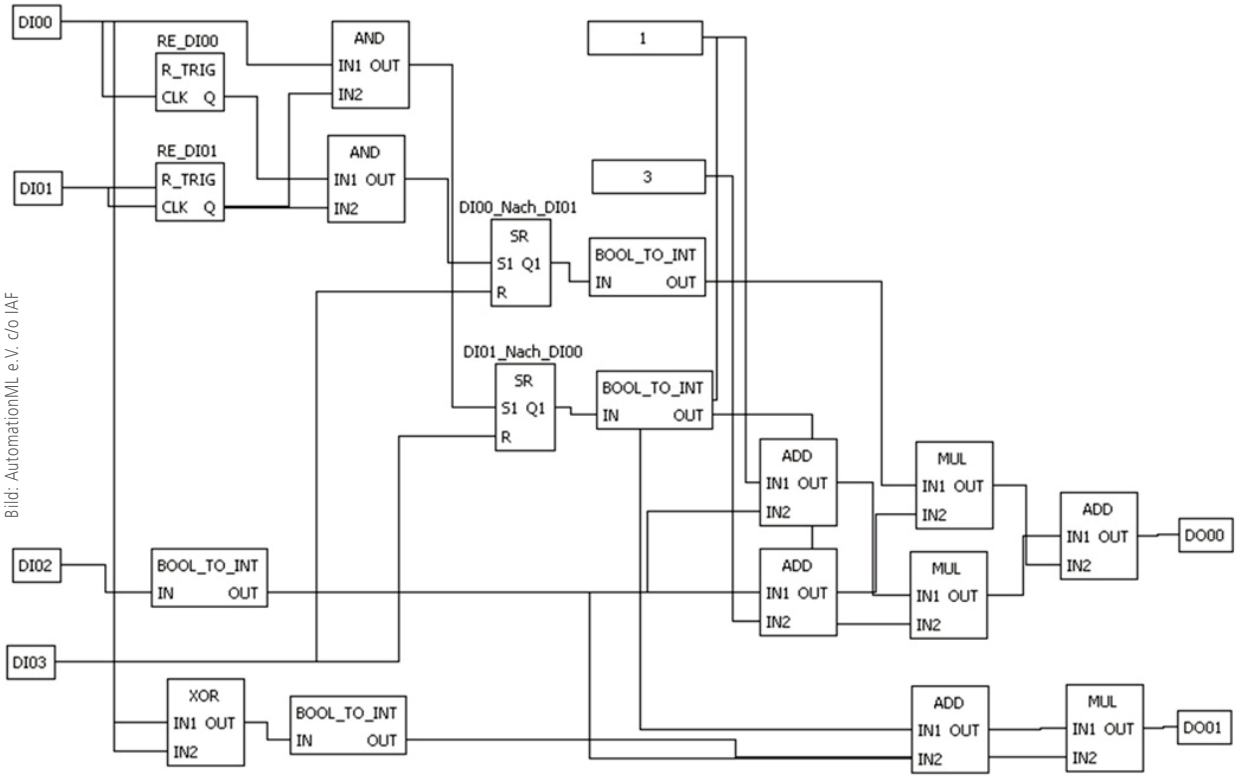


Bild: AutomationML e.V. c/o IAF

Bild 1: Steuerung des Antriebs im Funktionsblockdiagramm nach IEC61131-3

Die virtuelle Planung von Fertigungsanlagen gewinnt immer mehr an Bedeutung. Die Logik einzelner Komponenten sowie vollständige Anlagenabläufe inklusive aller möglichen Sonderfunktionen sollen im Vorfeld ausgiebig getestet werden können, noch vor der Installation auf der Baustelle. Dazu gehört beispielsweise auch das Verhalten von Antrieben für Förderer, welches dann mithilfe von gängigen Werkzeugen, für die WinMOD von Mewes & Partner oder Tecnomatix Process Simulate von Siemens PLM als Beispiele genannt werden könnten, simuliert werden kann.

Voraussetzung für die Simulation ist das Vorhandensein entsprechender simulierbarer Modelle, die neben der Geometrie und Kinematik auch das Verhalten der Systemkomponenten beschreiben. Die Anbieter von Systemkomponenten geben derzeit das Verhalten ihrer Komponenten in einem Handbuch mit oder sie bieten spezifische Module für die einzelnen Simulationswerkzeuge an. Die Möglichkeit, das Verhalten in einer neutralen Sprache zu beschreiben, würde sie unabhängig von dem vom Auftraggeber genutzten Simulationswerkzeug machen, vorausgesetzt eine Überführung in dieses Werkzeug ist möglich. Zudem können Bibliotheken von Systemkomponenten entstehen, die für verschiedene Simulationsziele unterschiedliche Modelle der Systemkomponenten bereitstellen, die der Wissenssicherung dienen und die Wiederverwendbarkeit dieser Systemkomponenten vereinfachen. Derartige Bibliotheken könnten aus unterschiedlichen Quellen gespeist werden. Kandidaten dafür sind neben den Systemin-

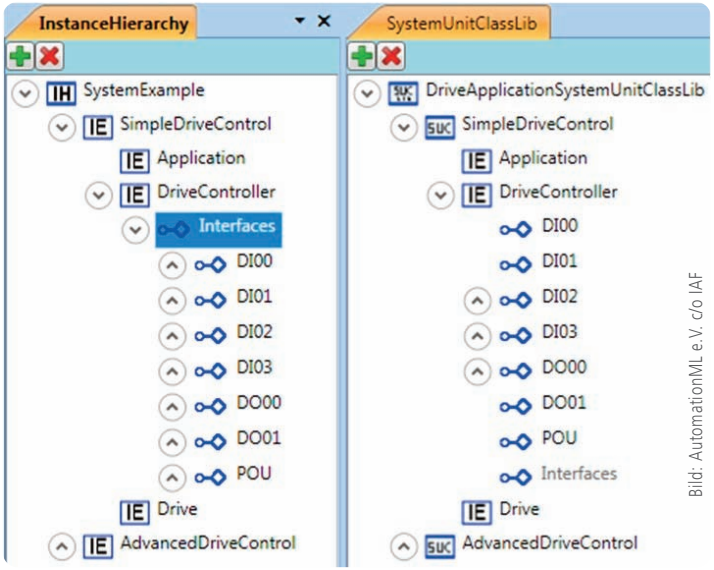


Bild: AutomationML e.V. c/o IAF

Bild 2: Komponentenbeispiel

```
<InternalElement Name="DriveController" ID="{397d601e-3300-473a-a6b0-3d8b6a28d5c7}">
  <ExternalInterface Name="DI00" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="{291
    <Attribute Name="refURI" AttributeDataType="xs:anyURI">
      <Value>//simpledrivecontrol.xml#DI00</Value>
    </Attribute>
  </ExternalInterface>
  <ExternalInterface Name="DI01" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="{915
    <Attribute Name="refURI" AttributeDataType="xs:anyURI">
      <Value>//simpledrivecontrol.xml#DI01</Value>
    </Attribute>
  </ExternalInterface>
  <ExternalInterface Name="DI02" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="{162
  <ExternalInterface Name="DI03" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="{3d4
  <ExternalInterface Name="DO00" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="{0b
  <ExternalInterface Name="DO01" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="{b8
  <ExternalInterface Name="POU" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface" ID="{2bf24
    <Attribute Name="refURI" AttributeDataType="xs:anyURI">
      <Value>//simpledrivecontrol.xml</Value>
    </Attribute>
  </ExternalInterface>
</InternalElement>
```

Bild: AutomationML e.V. c/o IAF

Bild 3: Beispielreferenzen auf PLCopen XML-Dateien

tegratoren auch Komponenten- und Gerätelieferanten. Dieser Artikel soll zeigen, dass AutomationML sowohl zum neutralen Austausch von Verhaltensinformationen geeignet ist als auch die Voraussetzung für eine Bibliotheksbildung erfüllt.

Ein Beispiel

Für ein besseres Verständnis soll an dieser Stelle ein Antriebsstrang betrachtet werden, wie er in Teil 5 dieser Artikelserie bereits vorgestellt wurde. Dieser soll einen Antrieb mit einer vereinfachten Antriebssteuerung enthalten. Dieser Antrieb kann sowohl im Rechts- als auch im Linkslauf betrieben werden und unterstützt zwei Geschwindigkeiten. Dabei darf er aus dem Stillstand in den Links- oder Rechtslauf gestartet werden. Zwischen den beiden Geschwindigkeiten darf innerhalb derselben Laufrichtung jederzeit gewechselt werden. Um den Antrieb allerdings aus der einen Laufrichtung in die andere zu schalten, muss er zunächst angehalten werden. Wird versucht den Antrieb vom Rechts- direkt in den Linkslauf zu schalten (oder umgekehrt), wird dies ignoriert und ein Fehlerstatus gesetzt, der zurückgesetzt werden kann. Hieraus ergeben sich die Eingänge der Steuerung, wie sie in Bild 6 zusammengefasst sind. Während des Betriebs sollen von der Antriebssteuerung zwei Informationen abgefragt werden können. Zum einen soll die aktuelle Geschwindigkeit und zum anderen der aktuelle Fehlerstatus ausgelesen werden können. Die sich daraus ergebenden Ausgänge der Steuerung sind in Bild 7 zusammengefasst. Das Verhalten dieser Steuerung lässt sich nach IEC61131-3 als Funktionsblockdiagramm (FBD) beschreiben und ist in Bild 1 zu sehen.

Herstellerunabhängige Komponentenbeschreibung

Für eine hersteller- und technologieunabhängige Beschreibung des Antriebstrangs kann AutomationML eingesetzt werden. Dazu werden CAEX zur Beschreibung der Komponentenstruktur und dazugehörige Schnittstellen und PLCopen XML zur Beschreibung des Komponentenverhaltens verwendet. In CAEX werden, wie in Bild 2 zu sehen, die einzelnen Geräte zu einer Gerätestruktur als Hierarchie von InternalElements zusammengefasst und bilden damit eine Komponente. In Bild 2 sind beispielhaft die Antriebsstränge SimpleDriveControl und AdvancedDriveControl gezeigt, die jeweils eine Komponente bilden und einen Antrieb, eine Antriebssteuerung und die Applikationsbeschreibung enthalten. Die einzelnen InternalElements der Hierarchie können notwendige Eigenschaften als Attribute enthalten. Zudem wird aus CAEX heraus auf PLCopen XML Dateien verwiesen, um

```
<body>
  <FBD>
    <inVariable localId="1" height="23" width="33">
    <inVariable localId="2" height="23" width="33">
    <inVariable localId="3" height="23" width="33">
    <inVariable localId="4" height="23" width="33">
    <outVariable localId="5" height="23" width="37">
    <outVariable localId="6" height="23" width="37">
    <block localId="7" width="47" height="40" typeName="R_TRIG" instanceName="RE_DI01">
    <block localId="8" width="53" height="60" typeName="AND">
      <position x="270" y="38"/>
      <inputVariables>
        <variable formalParameter="IN1">
        <variable formalParameter="IN2">
      </inputVariables>
      <inOutVariables/>
      <outputVariables>
        <variable formalParameter="OUT">
      </outputVariables>
    </block>
    <block localId="9" width="47" height="40" typeName="R_TRIG" instanceName="RE_DI00">
    <block localId="10" width="53" height="60" typeName="AND">
    <block localId="11" width="41" height="60" typeName="SR" instanceName="DI01_Nach_DI00">
    <block localId="12" width="41" height="60" typeName="SR" instanceName="DI00_Nach_DI01">
    <block localId="13" width="53" height="60" typeName="MUL">
    <inVariable localId="15" height="23" width="79">
    <inVariable localId="16" height="23" width="78">
    <block localId="17" width="53" height="60" typeName="ADD">
    <block localId="18" width="80" height="40" typeName="BOOL_TO_INT">
    <block localId="19" width="80" height="40" typeName="BOOL_TO_INT">
    <block localId="20" width="53" height="60" typeName="ADD">
    <block localId="21" width="53" height="60" typeName="MUL">
    <block localId="14" width="80" height="40" typeName="BOOL_TO_INT">
    <block localId="22" width="53" height="60" typeName="ADD">
    <block localId="23" width="53" height="60" typeName="XOR">
    <block localId="24" width="80" height="40" typeName="BOOL_TO_INT">
    <block localId="25" width="53" height="60" typeName="MUL">
    <block localId="27" width="53" height="60" typeName="ADD">
  </FBD>
</body>
```

Bild 4: FDB in PLCopen XML

Bild: AutomationML e.V. c/o IAF

Bild 5: Analogie zwischen PLCopen XML und Process Simulate

Bild: AutomationML e.V. c/o IAF

darüber die Verhaltensbeschreibungen einzubinden. Wie Bild 3 verdeutlicht, dienen dazu entsprechende ExternalInterface Elemente, die auf Objekte (ganze POEs oder einzelne Variablen) in den PLCopen XML Dateien verweisen.

PLCopen XML

Das FBD ist eine der fünf in der IEC61131-3 definierten Sprachen zur Programmierung von Speicherprogrammierbaren Steuerungen (SPS). PLCopen XML wurde entwickelt, um die Steuerungslogik gemäß IEC61131-3 in einem herstellerunabhängigen Format abbilden zu können. Ähnlich wie in der grafischen Repräsentation in Bild 1 wird in PLCopen XML jeder einzelne Funktionsbaustein als eigenes Element beschrieben. Die Eingangs- bzw. Ausgangsvariablen werden durch die Elemente <inVariable> und <outVariable> modelliert und die einzelnen Funktionsblöcke, die nach IEC61131-3 standardisiert sind, durch die Elemente <block> dargestellt. Die einzelnen Funktionsblöcke und Variablen werden nun über sogenannte Konnektoren miteinander verbunden, wodurch sich die Antriebssteuerung vollständig in der neutralen Sprache PLCopen XML modellieren lässt. Ein Ausschnitt aus der resultierenden Beschreibung ist in Bild 4 zu sehen. Um festzustellen, ob diese Darstellung auch für den herstellerunabhängigen Datenaustausch geeignet ist, muss untersucht werden, wie die gleiche Steuerung in einem Simulationssystem zu modellieren und ob eine automatische Überführung der Darstellungen möglich ist.

Process Simulate

Für diese Untersuchungen soll das Simulationssystem Process Simulate beispielhaft herangezogen werden. Es ist ein Repräsentant einer Werkzeugklasse, für die nach Ansicht der Autoren die nachfolgenden Aussagen ebenso zutreffen. Für die Beschreibung einer Steuerung steht im Simulationssystem Process Simulate ein Editor zur Verfügung, mit dessen Hilfe sich Eingänge, Ausgänge und Parameter definieren lassen. Eingängen und Ausgängen können Datentypen zugewiesen werden. Parameter definieren Funktionen, die aus n Variablen ein Ergebnis ermitteln. Als Variablen dienen entweder die Eingänge oder die Ergebnisse von anderen Parametern. Als Funktionen stehen neben arithmetischen und booleschen Operationen

Bild: AutomationML e.V. c/o IAF

Name des Eingangs	Typ	Bedeutung
DI00	Boolean	True: Rechtslauf starten False: Rechtslauf anhalten
DI01	Boolean	True: Linkslauf starten False: Linkslauf anhalten
DI02	Boolean	False: Betrieb mit Geschwindigkeit 1 True: Betrieb mit Geschwindigkeit 2
DI03	Boolean	Fehlerstatus zurücksetzen

Bild 6: Eingänge der Antriebssteuerung

Bild: AutomationML e.V. c/o IAF

Name des Ausgangs	Typ	Bedeutung
DO00	Integer	Abfrage Fehlercode: 0: kein Fehler 1: versuchte Änderung vom langsamen Rechtslauf in Linkslauf 2: versuchte Änderung vom schnellen Rechtslauf in Linkslauf 3: versuchte Änderung vom langsamen Linkslauf in Rechtslauf 4: versuchte Änderung vom schnellen Linkslauf in Rechtslauf
DO01	Integer	Abfrage Geschwindigkeit: 0: Motor angehalten 1: Antrieb im Betrieb Geschwindigkeit 1 2: Antrieb im Betrieb Geschwindigkeit 2

Bild 7: Ausgänge der Antriebssteuerung

auch einige vordefinierte Funktionen wie z.B. 'RisingEdge', 'FallingEdge' oder 'SetReset' zur Verfügung. Diese stellen ein Subset der in der IEC61131-3 standardisierten FBD Bausteine zur Beschreibung von Steuerungsverhalten dar. Gespeichert wird das Verhalten in einer XML Datei. Ein Blick in die erzeugte Datei (siehe Bild 5) zeigt, dass hier die Eingänge und Ausgänge sowie die einzelnen Funktionsblöcke in Listen gespeichert und über ihre IDs miteinander verknüpft werden.

Beliebig austauschbar!

Im direkten Vergleich ist ein analoger Aufbau der beiden XML Formate zu erkennen.

- Die Daten werden in Eingänge, Ausgänge und Funktionen bzw. FBD Bausteinen gruppiert.
- Eingänge sind die Variablen der Funktionen.
- Das Ergebnis einer Funktion ist entweder eine Variable für eine folgende Funktion oder der Wert eines Ausgangs.
- Alle Funktionen entsprechen denen in der IEC61131-3 beschriebenen.

Der Informationsgehalt in beiden Formaten ist somit identisch. Da es sich bei beiden Formaten um XML Dokumente handelt, wäre eine Überführung im einfachsten Fall durch eine XML Transformation mittels XSLT möglich. Process Simulate bietet zwar nur ein Subset der in der IEC61131-3 beschriebenen Funktionen an, allerdings steht es dem Anwender frei über eine Plugin-Schnittstelle weitere Funktionen im Editor zur Verfügung zu stellen.

Aber sicher!

Dem Bestreben Daten herstellerneutral abzubilden steht auch immer der Wunsch gegenüber, den Inhalt vor dem Zugriff Un-

befugter und vor nicht intendierter Veränderung abzusichern. In diesem Zusammenhang stellen sich im Wesentlichen zwei Fragen:

- Von wem sind die Daten und wurden sie auch nicht verändert?
- Wer darf die Daten sehen?

Die erste Frage beschäftigt sich also mit der Authentizität und der Integrität der Daten und die zweite mit dem Schutz vor Missbrauch. Beide Fragen sind so alt wie ihre Lösungsansätze. Authentizität und Integrität kann durch eine digitale Signatur gewährleistet und der Schutz vor Missbrauch kann durch Verschlüsselung erreicht werden. Welcher Standard dafür genutzt wird, sei es PKCS oder XML Signature/Encryption, ist reine Geschmackssache. Vielmehr sollte klar sein, welche Folgen die Nutzung von Verschlüsslungen bzw. Signaturen haben könnten. Eine digitale Signatur ist in jedem Fall sinnvoll. Die Daten sind lesbar, die Integrität kann jederzeit überprüft werden und die Entscheidung, ob der Signatur vertraut wird, kann frei getroffen werden. Sind die Daten hingegen (auch) verschlüsselt, so kann nur der vorgesehene Empfänger die Daten sehen. Wird eine Verschlüsselung als Transportsicherung für eine Übermittlung beispielsweise per Mail verwendet oder dient sie dem Wissensschutz bei einer Langzeitarchivierung, so ist dies mit Sicherheit sinnvoll. Hier muss jedoch für jeden Anwendungsfall genau überprüft werden, wer wann welche der verschlüsselten Informationen lesen bzw. verändern darf und wie ihm die dafür notwendigen Schlüssel zur Verfügung gestellt werden können. Insbesondere gilt dies, wenn der Empfänger ein Programm ist, das die Daten weiterverarbeitet. Hier muss durch alle Beteiligten sichergestellt sein, dass der Ansatz eines herstellerneutralen Datenformates nicht ad absurdum geführt wird. Für beide Anwendungsfälle, Verschlüsslungen und Signatur, sind für die Verarbeitung von XML Dateien entsprechende frei verfügbare Softwarekomponenten vorhanden, so dass ihre Anwendung nur vor organisatorischen, nicht jedoch vor technischen Herausforderungen steht.

Und nun?

Fasst man die beschriebenen Strukturen zusammen, so können herstellerunabhängige Bibliotheken wiederverwendbarer Systemkomponenten entstehen, die passend zu unterschiedlichen Anwendungsfällen Verhaltensmodelle enthalten. Diese Modelle können, wie erste Entwicklungsansätze zeigen, automatisch zur Erstellung von simulierbaren Anlagenmodellen kombiniert werden. Für deren Simulation könnte dann entweder auf bestehende Werkzeuge zurückgegriffen oder neue Simulationswerkzeuge unter Anwendung von IEC61131-3 basierten Soft-SPSen entwickelt werden. In beiden Fällen können insbesondere Komponentenlieferanten und Systemintegratoren profitieren. Erstere können sich durch die Bereitstellung von Simulationsmodellen einen Marktvorteil sichern und letztere können Wissen sichern und wiederverwenden und damit ihre Entwurfsqualität deutlich erhöhen. ■

www.automationml.org

Autor: Steffen Lips, Projektleiter, NetAllied Systems GmbH

Warum ein Engineering-Weltmodell bisher nicht gelang

Serie AutomationML Teil 10: Austausch von Daten zwischen unterschiedlichen Datenmodellen im heterogenen Werkzeugumfeld

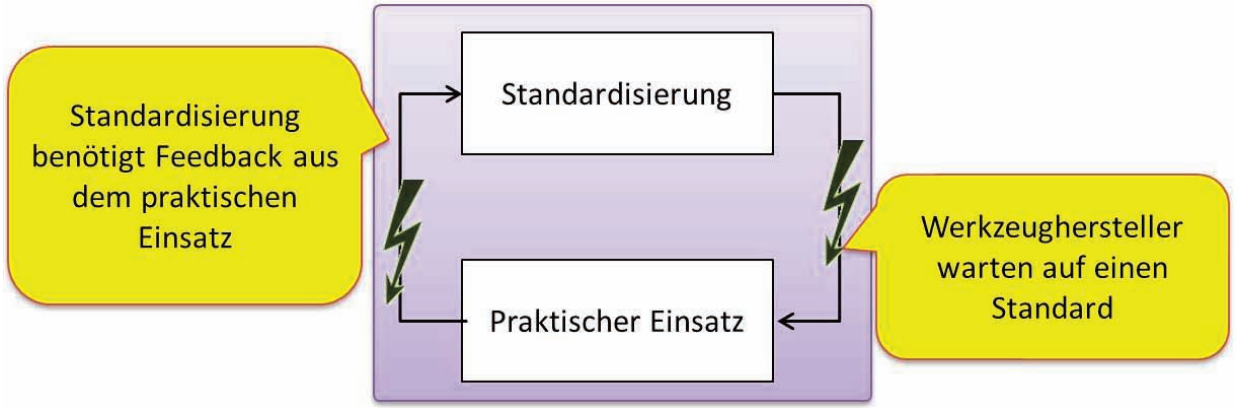


Bild: ABB AG Forschungszentrum Deutschland

Bild 1: Die Rückführungen im Standardisierungsregelkreis sind gestört.

Es klingt plausibel: Für ein neutrales Datenaustauschformat für das Engineering benötigt man eine standardisierte Syntax und Semantik. Es müsste ein 'Super-Datenmodell' entwickelt werden, das die wichtigsten Datenmodelle des Engineering vereint und eine eindeutige und verlustlose Abbildung aller Daten ermöglicht.

Die Idee des 'Super-Datenmodells' ist die treibende Kraft aller entsprechenden Standardisierungsaktivitäten, z.B. NE100 [1], STEP [2], ISO15926 [3] u.v.m. Dazu müssen sie stets folgende Schritte durchlaufen:

- a) Experten sammeln für beteiligte Gewerke bzw. Werkzeuge sinnvolle Datenmodelle und Konzepte.
- b) Experten einigen sich auf die gemeinsam benötigten Grundkonzepte.
- c) Experten einigen sich auf ein neutrales, semantisches Datenmodell, das alle Grundkonzepte berücksichtigt, z.B. mit UML.
- d) Experten einigen sich auf eine neutrale Syntax, z.B. basierend auf XML.
- e) Alle gesammelten Konzepte, das semantische Datenmodell und die Umsetzung in einer gemeinsamen Syntax, werden dokumentiert.
- f) Die Industrie und Akademia schafft die Voraussetzungen dafür, dass genügend Experten zur Verfügung stehen, die den Standard beurteilen und implementieren können.
- g) Anwender und Werkzeughersteller implementieren und nutzen den Standardentwurf in ihren Werkzeugen.
- h) Erfahrungen aus der praktischen Nutzung des Standards fließen in (a) zurück und tragen zur Reifung bei.

Bis heute ist dies nicht gelungen. Bild 1 erklärt den Konflikt: Standardisierung reift nur durch Rückfluss von Erfahrungen aus der praktischen Anwendung. Werkzeughersteller warten jedoch aus Kostengründen typischerweise erst auf einen gereiften Standard, bevor sie ihre Software mit Ex- und Importern ausstatten. Die Rückführungen der praktischen Erfahrungen in die Standardisierung (h) sowie die notwendigen Implementierungen der Hersteller (g) sind gestört, die Standardisierung befindet sich in einem Deadlock.

Die Idee: Ein gemischtes Datenmodell

Ein Weltmodell des Engineering ist offensichtlich nicht in einem Schritt erreichbar. AutomationML [4] verfolgt daher

```
<AdditionalInformation>
  <WriterHeader>
    <WriterName>ToolX AutomationML Exporter</WriterName>
    <ToolWriterID>ToolXToAutomationML123</ToolWriterID>
    <WriterVendor>ToolX Vendor</WriterVendor>
    <WriterVendorURL>http://www.ToolX-Vendor.org</WriterVendorURL>
    <WriterVersion>0.1</WriterVersion>
    <WriterRelease>123 prealpha</WriterRelease>
    <LastWritingDateTime>2011-05-25T09:30:47</LastWritingDateTime>
    <WriterProjectTitle>eCarproduction</WriterProjectTitle>
    <WriterProjectID>eCarproduction_LinePLC.prj</WriterProjectID>
  </WriterHeader>
</AdditionalInformation>
```

Bild: ABB AG Forschungszentrum Deutschland

Bild 2: AutomationML-Beispiel-Etikett

einen anderen Ansatz. Anstatt auf eine Standardisierung zu warten, lässt es die Speicherung proprietärer Semantiken explizit zu [5]. Beim Export aus einem Engineeringwerkzeug werden die Daten syntaktisch neutralisiert, die Semantik bleibt jedoch proprietär ('privat'). Ein Signal würde beispielsweise alle Eigenschaften inklusive seines Namens aus dem Ursprungswerkzeug behalten. Auch wenn das Zielwerkzeug diese Daten zunächst nicht interpretieren kann (gerade zu Beginn), stellt dies bereits einen erheblichen Gewinn dar: Erstmalig werden proprietäre Datenmodelle aus ihrem Werkzeug herausgelöst, syntaktisch neutralisiert und untersuchbar. Dies erleichtert die beiden Schritte (a) und (b) der beschriebenen Standardisierungskette erheblich. Ziel-Engineering-Werkzeuge können die Datei öffnen, durchsuchen und darstellen. Funktionen wie Navigation, Änderungsberechnungen und Versions-Management sind unabhängig vom Datenmodell und lassen sich generisch entwickeln und wiederverwenden. Es fehlen nur noch die Transformationsroutinen, die für die proprietären Datenmodelle des Quellwerkzeuges zugeschnitten sind. Doch woher kommen sie und wie kann ein Importer erkennen, welche davon aufgerufen werden müssen?

Die Idee: Identifikation des Quellwerkzeuges

AutomationML führt dazu erstmalig einen standardisierten Etikettiermechanismus ein. Die CAEX-Datei bekommt ein Etikett (siehe Bild 2), welches während des Exports in die AutomationML-Datei eingebettet wird und Informationen über das Quellwerkzeug enthält. Ein Importer kann dieses Etikett auswerten und private Transformationsroutinen anstoßen. Dieses Konzept wird von der frei verfügbaren Software AutomationML Engine [6] unterstützt. Diese Vorgehensweise bedeutet keineswegs, dass jedes Werkzeug seine Daten dann auch gleich in seinem eigenen proprietären Datenformat exportieren könnte. Bereits die syntaktische Neutralisierung bringt entscheidende Vorteile:

- Der Hauptaufwand eines Importers liegt in Operationen wie Versionsmanagement oder Änderungsberechnungen. Diese Operationen sind unabhängig vom Datenmodell. Derartige Funktionalität müsste für jedes proprietäre Datenformat erneut entwickelt werden. Basierend auf einem neutralen Datenformat genügt eine einzige Implementierung. Ein Großteil des Importers kann daher – unabhängig vom jeweiligen Quellwerkzeug – wiederverwendet werden.
- Das Etikettierkonzept erlaubt die automatische Absendererkennung. Proprietäre Datenformate wie z.B. Excel-Sheets etc. verfügen über keine standardisierte Methodik dafür.

Und dann?

Die quellwerkzeugspezifischen Subroutinen für die Interpretation und den Import müssen erst entwickelt werden. Die Erkennung des Quellwerkzeuges ist der Schlüssel. Der 'Trick' besteht darin, dass der Importer 'unbekannte private Datenmodelle' erkennen und in einer separaten CAEX-Bibliothek speichern kann. Mit anderen Worten: Der Importer weiß, was er nicht weiß. Eine Bibliothek unbekannter Datentypen bzw. Klassen enthält alle Datenmodell-Kandidaten, sozusagen ein Lastenheft für die Softwareentwicklung. Anhand von Häufigkeitsuntersuchungen lassen sie die Kandidaten sogar priorisieren. Private Transformationsroutinen können damit ohne Warten auf einen Standard in lokaler Verantwortung entwickelt werden. Das geht

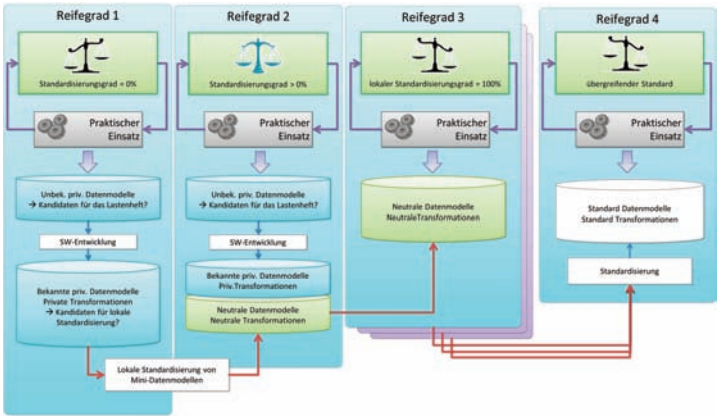


Bild: ABB AG Forschungszentrum Deutschland

Bild 3: Reifegradmodell

schneller als jede Standardisierung. Das Ziel ist zunächst der gelebte praktische Einsatz und die Reifung unter realen Bedingungen. Nach der Entwicklung erster Transformationsroutinen werden aus 'unbekannten' somit 'bekannte' Datenmodelle. Sie lassen sich für jeden weiteren Datenaustausch mit demselben Quellwerkzeug wiederverwenden. Das beschriebene Szenario ist unmittelbar einführbar. Zwei Werkzeughersteller oder -nutzer können beliebig Export-Import-Partnerschaften eingehen. Ein dritter Teilnehmer kann sich (auch später) einbinden. Dieses Szenario stellt zugleich Reifegrad 1 eines evolutionären Reifegradmodells dar: (siehe Bild 3). Dieser ist ein erster Meilenstein in der Evolution eines Standardisierungsprozesses: Er erleichtert die Schritte (a) und (b) und belebt die Rückführung von Erfahrungen (h) aus der praktischen Anwendung. Der Deadlock aus Bild 1 wird durchbrochen. Dieser Reifegrad ist für Werkzeuge geeignet und möglicherweise sogar ausreichend, die nur wenige Datenaustauschpartner besitzen. Die Entwicklung privater Transformationsroutinen benötigt deutlich weniger Aufwand als eine Standardisierung. Reifegrad 1 ist weiterhin ideal für Werkzeuge, die lediglich ihre eigenen Daten archivieren bzw. extern manipulieren und anschließend wieder einlesen möchten. Mit einer wachsender Zahl von Teilnehmern wird eine Bibliothek aus Transformationsroutinen entstehen. Dies ist ein

geeigneter Indikator für den Start einer Konsolidierung des Entwicklungsaufwandes und somit ein geeigneter Übergangspunkt zu Reifegrad 2, denn ähnliche Datenmodelle sind Kandidaten für eine erste Standardisierung. Reifegrad 2 entsteht aus der Konsolidierung der Standardisierungskandidaten aus Reifegrad 1. Ähnliche Datenmodelle führen zu Mehrfachentwicklung von ähnlichen Transformationsroutinen und erhöhen unnötig die Entwicklungskosten. Dies fördert die Standardisierungsbereitschaft und führt zu ersten neutralen 'Mini'-Datenmodellen. Für jedes neutralisierte 'Mini'-Datenmodell kann in jedem Importer die Zahl n der privaten Transformationsroutinen für n Quellwerkzeuge auf eine einzige Transformationsroutine reduziert werden. Auf diese Weise entsteht ein Pool an neutralen Datenmodellen, während der Pool privater Transformationsroutinen abnimmt. Der Evolutionsprozess hin zu Reifegrad 2 wird somit aus praktischen Projektbedürfnissen getrieben. Dies stellt einen natürlichen und evolutionären Prozess dar. Unter Verwendung erster neutraler Daten-Teilmodelle erzeugt ein Exporter in Reifegrad 2 ein gemischt standardisiert/privates CAEX-Datenmodell. Reifegrad 3 wird erreicht, wenn alle benötigten Daten-Teilmodelle durch neutrale Datenmodelle abgebildet werden. Er bildet das Gegenstück zu Reifegrad 1, da hierbei der Anteil an privaten Datenmodellen 0% beträgt. Seine Reife erlangt er durch Erfahrungen im praktischen Einsatz. Reifegrad 3 ist nicht an internationale Standards gebunden. Es könnte ebenso ein Standard eines einzelnen Projektes oder eines Konsortiums sein. Dieser Reifegrad kommt der eigentlichen Idee eines neutralen Datenaustauschformates bereits nahe: Private Transformationsroutinen sind nicht mehr notwendig – das Datenmodell ist aus Erfahrungen der Reifegrade 1-2 gewachsen und gereift. Reifegrad 4 wird erreicht, wenn mehrere 'Reifegrad 3 Standards' zu einem übergeordneten Standard kombiniert werden. Die Datenmodelle aus Reifegrad 3 bilden aufgrund praktischer Erfahrungen eine belastbare Basis für die Schritte (a) und (b) eines übergreifenden Standardisierungsvorhabens, z.B. für einen internationalen Standard. Dieser Reifegrad ist als langfristiges Ziel zu verstehen und ein evolutionäres Ergebnis des Durchlaufens der vorigen Reifegrade.

Diskussion und Zusammenfassung

Standardisierungsgremien und Softwarehersteller haben frühzeitig den Wert eines gemeinsamen Datenmodelles erkannt, verfolgen jedoch mehrheitlich Reifegrad 3 bzw. 4. Für die Praxis macht es jedoch keinen Sinn, sofort mit Reifegrad 3 oder 4 zu beginnen. Eine Evolution beginnt stets mit dem ersten Schritt (Reifegrad 1). Mit dem vorgestellten Konzept wird semantische Vielfalt beherrschbar. Die Mischung privater und neutraler Daten eliminiert den Bedarf nach einem umfassenden und gereiften Engineering-Weltmodell: Der Schwerpunkt verlagert sich stattdessen auf kleine und flexible 'Mini'-Datenmodelle. Diese 'Mikro-Standardisierung' zwischen zwei Werkzeugen ist ein agiler und pragmatischer Weg mit iterativen Standardisierungs-Sprints. Der praktische Einsatz steht in jedem Reifegrad im Vordergrund, die Standardisierung folgt der Nutzung, nicht umgekehrt. Erstmalig lässt sich Standardisierungsbedarf inklusive Priorisierung automatisch ableiten. Mit diesem Ansatz versucht AutomationML nicht, die bestehende Heterogenität von Engineering-Werkzeugen und Datenmodellen durch Vereinheitlichung zu überwinden; Das Konzept verfolgt vielmehr das Anliegen, den Datenaustausch in einem heterogenen Werkzeugumfeld auch ohne bzw. nur mit teilweiser Standardisierung durchführen zu können. Anwender können Datenaustauschlösungen deutlich leichter und schneller als bisher entwickeln. Gerade für das Umsetzen von Werkzeugketten mit nur wenigen Datenaustauschpartnern ist der Entwicklungsaufwand deutlich geringer als das Standardisieren eines Weltmodells. Das Warten auf einen Standard ist nicht mehr nötig. Zugleich ebnet diese Vorgehensweise den Weg zu einer evolutionären Standardisierung, entkoppelt jedoch dessen Zeitbedarf vom Projektdruck in der Industrie. Erste Verbände (z.B. VDA und VDMA) haben dies erkannt und verfolgen statt breiter Standardisierung die zügige Entwicklung leichtgewichtiger Schnittstellen für begrenzte Bereiche. ■

www.automationml.org

Literatur

[1] Namur Empfehlung 100: Merkmalleisten zur Erstellung von PLT-Gerätespezifikationen. 2003.
[2] ISO10303: Automation systems and integration - Product data representation and exchange, formerly known as STEP, Standard for the Exchange of Product model data.
[3] ISO15926. Industrial automation systems and integration - Integration of life-cycle data for process plants in-cluding oil and gas production facilities.
[4] IEC62714-1 CD norm draft AutomationML Architectu-re, www.iec.ch, 2011.
[5] Drath R., Barth M.: Wie der Umgang mit unterschiedlichen Datenmodellen beim Datenaustausch im heterogenen Werkzeugumfeld gelingt. In: Tagungsband Automation 2013, Baden-Baden, VDI-Berichte 2209, ISBN 978-3-18-092209-6, 2013.
[6] www.automationml.org

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

AutomationML – Engineering Workflow

Serie AutomationML Teil 11: Unterstützung von Engineering Workflows

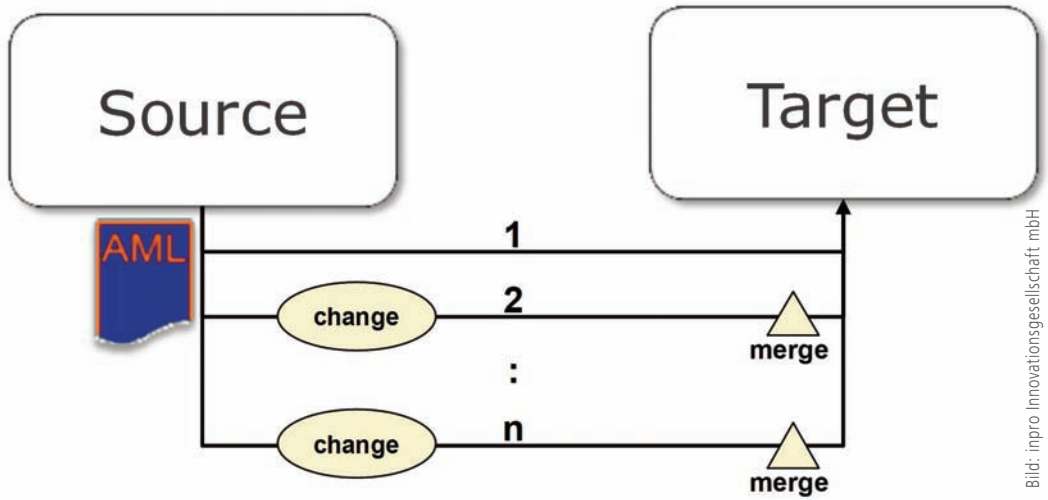


Bild: inpro Innovationsgesellschaft mbH

Bild 1: Iteratives Engineering

Der Datenaustausch zwischen zwei Entwurfswerkzeugen während eines Entwicklungsprojektes ist oft keine einmalige Angelegenheit, sondern findet wiederholt statt. Objekte werden auch nach einem Datenaustausch beim Datenversender und beim Datenempfänger geändert, spezifiziert oder möglicherweise neu definiert. Bei einem wiederholten Datenaustausch in einem Entwicklungsprojekt sollten möglichst alle Änderungen als solche kenntlich gemacht werden, sodass das Zielwerkzeug einen selektiven Import der Daten durchführen kann und Änderungen mit dem eigenen Datenbestand vergleichen und in diesen sinnvoll integrieren kann. Ein Änderungs- und Versionsmanagement ist bei lose gekoppelten Werkzeugen, die lediglich über eine Datei-Schnittstelle und nicht über eine gemeinsame Datenbank verknüpft sind, in der Regel nicht möglich. Dieser Beitrag zeigt, wie auch für die dateibasierte Kopplung – mit Hilfe des AutomationML Austauschformates – ein Änderungs- und Versionsmanagement implementiert und damit komplexere Workflows unterstützt werden können, die über den einmaligen Datenaustausch weit hinausgehen.

Beim iterativen Engineering (Bild 1) werden Daten zwischen zwei Engineering-Werkzeugen während eines Entwicklungsprojektes wiederholt ausgetauscht. Der Datenaustausch ist unidirektional vom Quell- zum Zielsystem organisiert. Bei einem wiederholten Datenexport werden dabei vom Engineeringsystem, das die Datenquelle repräsentiert, zu jedem geänderten AutomationML-Objekt die entsprechenden Änderungsinformationen (mit) in die AutomationML-Datei geschrieben. Diese Informationen werden auf der Empfängerseite interpretiert, sodass eine Selektion und Integration der Daten in das Zielsystem erfolgen kann. Zur Kennzeichnung von Änderungen der Daten innerhalb eines Projektes bietet AutomationML folgende Konzepte an:

- ChangeMode – Attribute für jedes CAEX-Objekt
- Revision/Version – Information für jedes CAEX-Objekt
- Globally Unique Identifier (GUID) für alle AutomationML-Objekte
- Projekt Identifikation im 'Writer-Header'

Das 'ChangeMode'-Attribut erlaubt dem Versender, ein Objekt als neu ('create'), geändert ('change'), unverändert ('state') oder gelöscht ('delete') zu kennzeichnen. Soll das ursprüngliche Objekt auch bei Änderungen noch in der AutomationML Datei erhalten bleiben, besteht die Möglichkeit, das geänderte Objekt neu anzulegen und bei diesem und bei dem ursprünglichen Originalobjekt entsprechende Versionsinformationen zu hinterlegen. Der Versender der Daten muss bei einem wiederholten Datenexport auf die ursprünglich erzeugten AutomationML-Dokumente (bzw. auf die zuletzt versendete Datei) zugreifen kön-

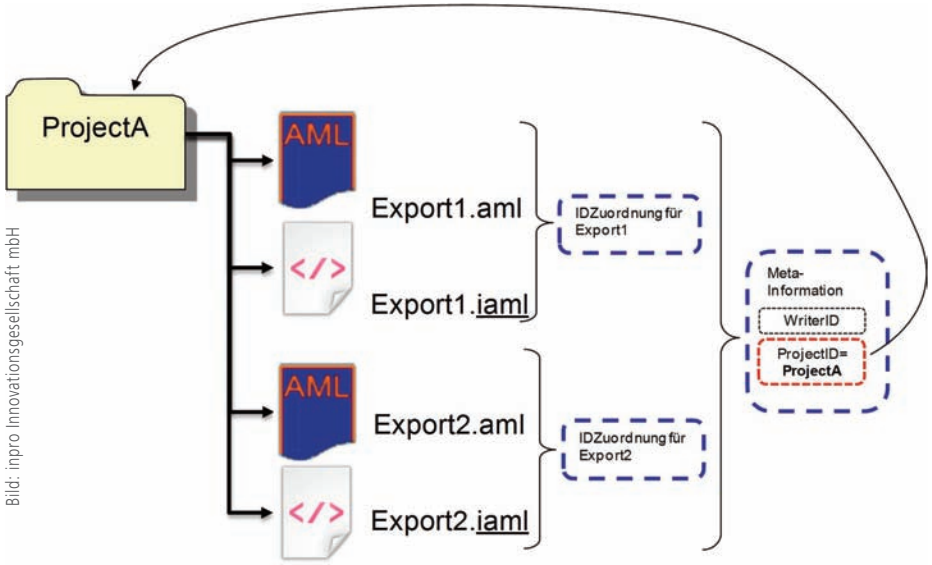


Bild: inpro Innovationsgesellschaft mbH

Bild 2: Organisation der Dateiverwaltung

nen, um Änderungsinformationen generieren zu können. Eine einfache Möglichkeit dies zu realisieren, ist die Verwendung eines projektspezifischen Dateiablageorts für die Versionskontrolle der versendeten AutomationML-Dateien durch den Versender. In diesem Verzeichnis können auch die nötigen Zuordnungsdateien (Endung .iaml in Bild 2) abgelegt werden, über die die Zuordnung der AutomationML-Objekte zu den ursprünglichen Systemobjekten erfolgen kann (Bild 2). Eine Voraussetzung zur effektiven Nutzung der beschriebenen Konzepte besteht darin, dass jedes AutomationML-Objekt eine eindeutige globale ID (GUID) erhält. Diese ID bleibt über den gesamten Lebenszyklus des Objektes erhalten. So behält ein AutomationML-Objekt, das bei einem wiederholten Datenaustausch als geändertes Objekt gekennzeichnet wird, die ursprüngliche GUID, die es bei seiner 'Geburt' erhalten hat. In einer Zuordnungstabelle kann diese GUID mit einem systemeigenen Identifizierungsschlüssel assoziiert werden, sodass ein Exporter die Änderungsinformationen für jedes Objekt anhand der Zuordnungstabelle und einem Vergleich mit der zuletzt erzeugten AutomationML-Datei generieren kann. Das Empfängerwerkzeug der Daten kann erkennen, ob ein AutomationML-Objekt zum ersten Mal gesendet wird, da es in diesem Fall den ChangeMode 'create' besitzt. In allen nachfolgenden Sendungen besitzt dasselbe Objekt einen anderen der möglichen ChangeMode-Werte ('change', 'state', 'delete'). Bei einem selektiven Import müssen Objekte mit einem ChangeMode-Wert von 'change' oder 'delete' besonders behandelt werden, da in der Regel die eingelesenen Daten seit dem letzten Import im Zielsystem weiter bearbeitet wurden. Bei geänderten Objekten (ChangeMode='change') muss darauf geachtet werden, dass keine Änderungen überschrieben werden und dadurch Planungsergebnisse verloren gehen. Auch bei gelöschten Objekten (ChangeMode='delete') ist Vorsicht geboten, wenn diese inzwischen mit Planungsdaten verknüpft wurden, die nicht automatisch mit gelöscht werden dürfen. Der Empfänger der Daten muss für die Wiedererkennung von importierten Objekten aus vorherigen Iterationen auf jeden Fall die ID-Zuordnungstabelle aufheben, welche alle Zuordnungen zwischen AutomationML-Objekt IDs (GUID) und System-Objekt IDs enthält.

Es kann vorteilhaft sein, für den Vergleich von Änderungen das schon existierende Systemobjekt zunächst in ein AutomationML-Objekt zu transformieren, um den Vergleich auf Basis eines einheitlichen Formats durchführen zu können. Bild 3 zeigt einen Lebenszyklus eines AutomationML-Objektes über mehrere Iterationsstufen. Im Beitrag zum Thema Datenkonsistenz [1] skizzieren die Autoren ein Systemkonzept für ein systemunabhängiges Werkzeug, das auf Grundlage der AutomationML-Dokumente und der enthaltenen Absender- und Änderungsinformationen die Datenkonsistenz in solchen Engineering Workflows überwachen und unterstützen kann.

Roundtrip Engineering

Der Workflow des Roundtrip Engineer-ing stellt eine Erweiterung des Iterativen-Engineerings dar und erlaubt das Ändern und Zurücksenden von importierten Daten an das Ursprungswerkzeug. Im Engineering tritt der Fall häufig auf, beispielsweise wenn ein generisches Objekt oder generische Merkmale

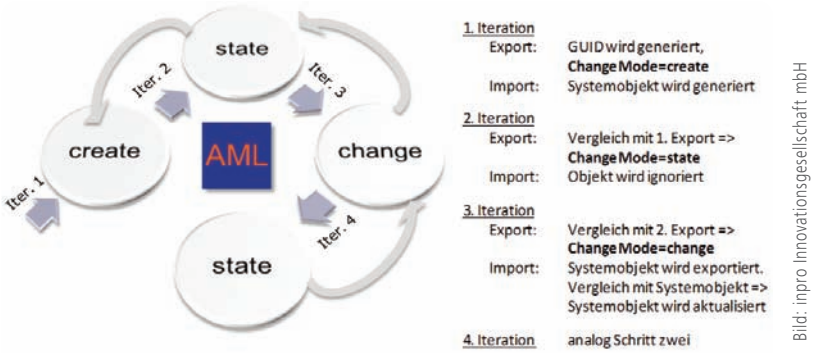


Bild: inpro Innovationsgesellschaft mbH

Bild 3: Lebenszyklus eines AutomationML-Objektes im iterativen Engineering

Bild: inpro Innovationsgesellschaft mbH

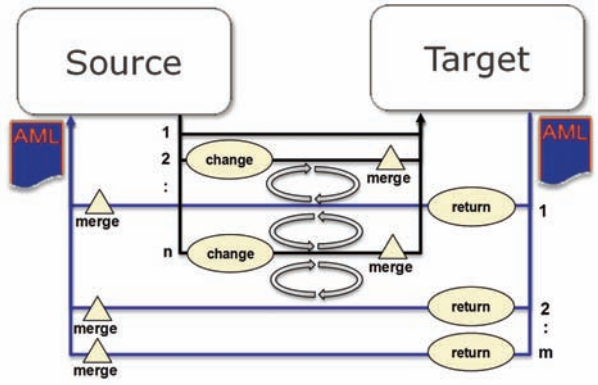


Bild 4: Roundtrip Engineer-ing

eines Objektes spezifiziert werden, um den spezifizierten Wert danach in allen weiteren Planungsschritten benutzen zu können. Ein Beispiel hierfür wäre die Kalkulation eines Zeit- oder Kostendatums durch ein spezifisches Kalkulationstool. Auch in diesem Fall kann der Datenaustausch zwischen den Werkzeugen wiederholt stattfinden (Bild 4). In diesem Fall besitzt jedes der beteiligten Systeme sowohl Export- als auch Import-Funktionen und eine Möglichkeit zur Verwaltung der AutomationML-Dokumente und Zuordnungstabellen. Der Datenexport, den das Zielsystem durchführt, ist aber restriktiver als der Datenexport des Quellsystems. Das Zielsystem generiert beim Export AutomationML-Objekte, die an vorher eingelesene AutomationML-Objekte angefügt werden. Das generierte AutomationML-Dokument enthält daher immer AutomationML-Objekte, die auch in der zuerst importierten AutomationML-Datei enthalten waren. Idealerweise wird dafür dasselbe AutomationML-Dokument, das vorher empfangen wurde, genutzt. Nur so kann der ursprüngliche Sender erkennen, welche Objekte in der zurückkommenden Datei geändert/erweitert wurden. Zur Gewährleistung der Datenkonsistenz kann es besser sein, die Originalobjekte nicht direkt zu verändern, sondern Kopien zu erzeugen, zu verändern und als Revisionen bzw. Versionen des Originalobjektes zu versenden. AutomationML enthält mehrere Konzepte für das Roundtrip Engineering. Der Datenerzeuger und Versender kann einem Empfänger Vorgaben für die Spezifikation mitgeben, indem er AutomationML-Objekte mit 'RoleRequirements' und 'Attribute-Constraints' assoziiert. Der Empfänger kann Spezifikationen durch 'Verfeinerungen' und 'Klassen-Instanz-Relationen' mit entsprechenden spezifischen Klassenbibliotheken definieren. Für die Änderung von Objekten durch den Empfänger sind das 'ChangeMode'-Attribut und das 'Versions- und Revisionskonzept' nutzbar. Das Integrieren der zurückkommenden Daten erfolgt in ähnlicher Weise, wie im unidirektionalen iterativen Engineering. Für die Integration wird die Änderungsinformation, die entweder direkt durch das Setzen des 'ChangeMode'-Attributes oder durch 'Versions-' und 'Revisionsobjekte' angegeben ist, ausgewertet, mit den vorhandenen Daten verglichen und in den vorhandenen Datenbestand eingefügt.

Fazit

Mit den ausdrucksstarken Beschreibungskonzepten von AutomationML ist es möglich, auch für lose gekoppelte Systeme ein Änderungs- und Versionsmanagement für definierte Engineering-Workflows zu implementieren. Die in diesem Beitrag dargestellten Konzepte und Workflows können auch für einen Verbund von mehreren Werkzeugen einzeln oder in Kombination verwendet werden. Weiterhin hat der Beitrag verdeutlicht, dass mit AutomationML-Dokumenten die Realisierung eines zentralen Datenpools für die redundanzfreie Verwaltung von Engineering-Daten möglich ist. Für diesen Fall müssen die beschriebenen Modellierungskonzepte des Formates zur Änderungs- und Versionskontrolle genutzt und darüber hinaus muss lediglich eine wechselseitige Zugriffskontrolle für die genutzten AutomationML-Dokumente implementiert werden. ■

Literatur

[1] R. Drath, B. Schröter, M. Hoernicke, Datenkonsistenz im Umfeld heterogener Engineering-Werkzeuge in VDI Automation 2011.

www.automationml.org

Autor: Josef Prinz, Senior-Experte Digitale Fabrik, inpro Innovationsgesellschaft mbH

Autor: Dr.-Ing. Lorenz Hundt, Projektingenieur Produktionssyst. und Informationsprozesse, inpro Innovationsgesellschaft mbH

AutomationML verbindet Software-Werkzeuge

Serie AutomationML Teil 12: Validierung von Verhalten in unterschiedlichen Phasen des Anlagenentwurfs

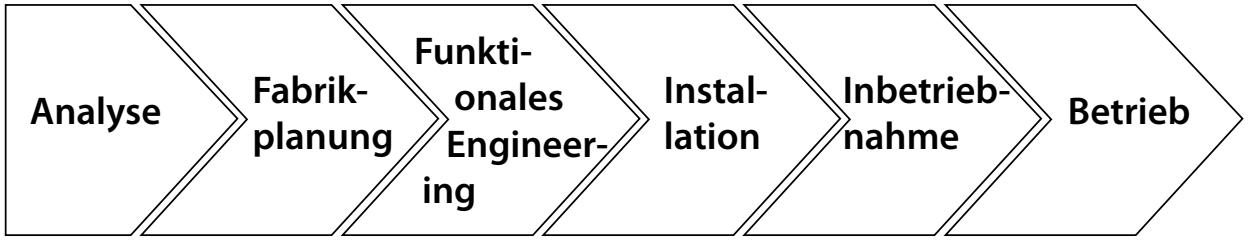


Bild: Otto-von-Guericke-Universität Magdeburg

Bild 1: Allgemeiner Anlagenentwurfsprozess

Die Planung, das 'Engineeren', einer Anlage für die Fertigungsindustrie oder die Prozessindustrie ist ein ressourcenintensiver und komplexer Prozess, an dem unterschiedliche ingenieurstechnische Disziplinen beteiligt sind. Dabei erfolgt die Planung anfangs recht grob und wird mit Fortschritt im Entwurfsprozess stetig detaillierter. Das Anlagenverhalten zum Beispiel wird anfangs mittels einfacher Aktivitäten modelliert, welche final in den Steuerungscode gipfeln. Bei diesem Prozess der kontinuierlichen Informationsanreicherung setzen Ingenieure ein gewisses Vertrauen in die Korrektheit der an sie übergebenen Daten – diese werden nicht noch einmal bis ins Detail überprüft. Irren ist jedoch menschlich.

Fehler sollten früh gefunden werden. Es gilt: Je früher ein Fehler gefunden wird, desto geringer ist der Aufwand für dessen Behebung. Ist es da nicht sinnvoll, die Daten zu überprüfen, bevor sie weitergegeben werden? Bezogen auf das Anlagenverhalten: ob das modellierte Verhalten dem Beabsichtigten entspricht?

Welche Informationen werden benötigt?

Um das Anlagenverhalten simulativ überprüfen zu können, bedarf es grundsätzlich folgender Informationen:

- Die Anlagenstruktur: Welche Produktionsressourcen werden verwendet und wie sind diese untereinander angeordnet?
- Die Geometrie und Mechanik der Anlage: Wie sind deren Gestalt und die Bewegungsmöglichkeiten?
- Das Anlagenverhalten: In welcher Reihenfolge werden welche Schritte bzw. Aktivitäten auf den Produktionsressourcen ausgeführt?

Dabei setzt sich die Geometrie und Mechanik der Anlage aus der Geometrie und Mechanik der einzelnen Produktionsressourcen zusammen, deren Anordnung mit der Anlagenstruktur beschrieben wird. Um nun das Verhalten dieser Komposition der Produktionsressourcen simulieren zu können, muss jede Produktionsressource selbst mit modulspezifischem Verhalten versehen werden.

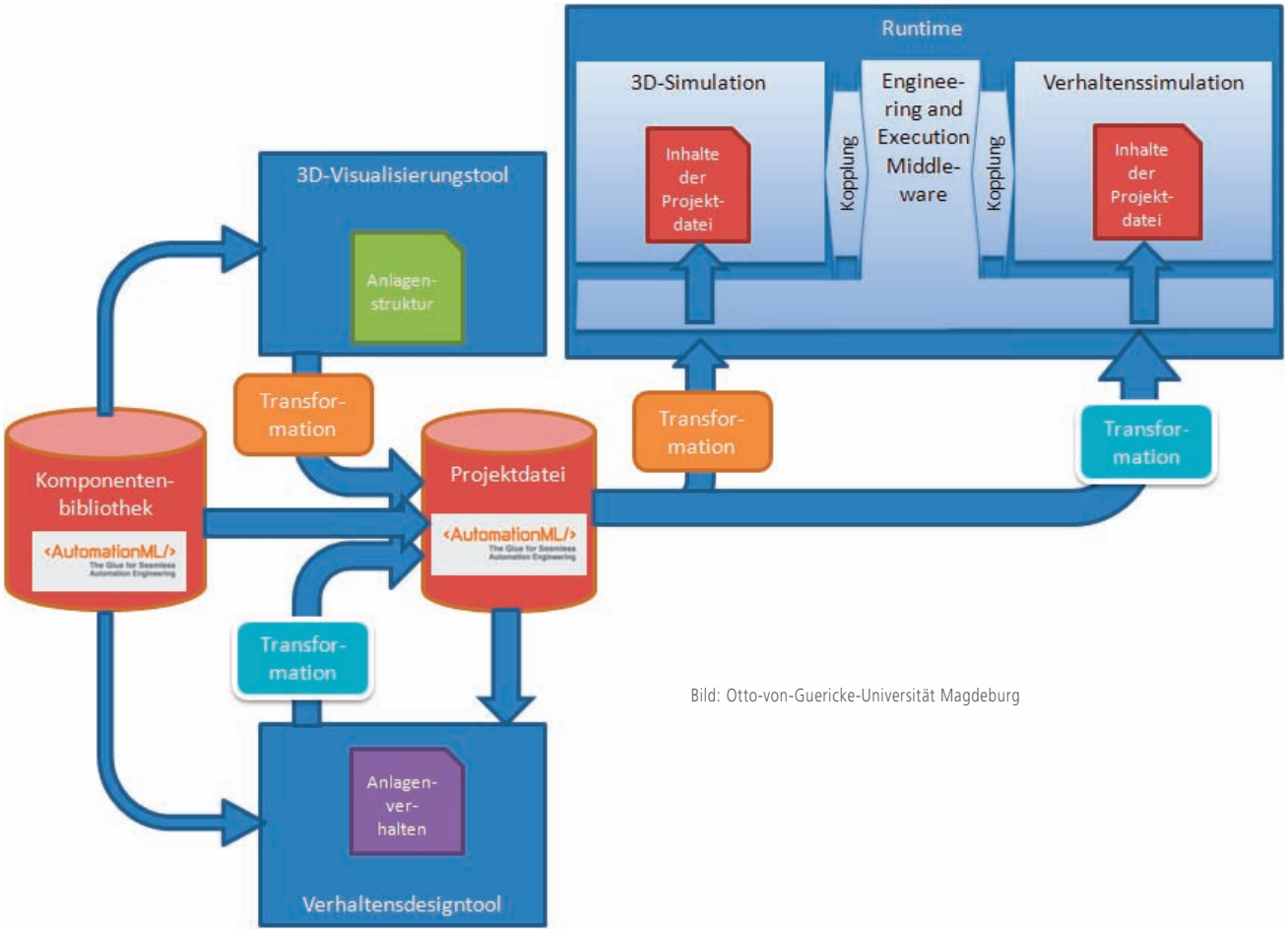


Bild: Otto-von-Guericke-Universität Magdeburg

Bild 2: Schematische Darstellung des Simulations-Frameworks

Wo ist das einsetzbar?

Die Planung einer Anlage kann im Allgemeinen in sechs Phasen eingeteilt werden (siehe Bild 1). In welcher Phase bietet sich eine simulative Überprüfung des Verhaltens an? Und was kann da konkret überprüft werden? In der Fabrikplanungsphase wird das Anlagenverhalten grob beschrieben – meist mithilfe einfacher Modelle, wie Gantt oder PERT Charts. An dieser Stelle kann überprüft werden, ob sich das Produkt mit den festgelegten Fertigungsschritten mit den gewählten Produktionsressourcen fertigen lässt. Innerhalb des funktionalen Engineerings liegen bereits detailliertere Verhaltensbeschreibungen vor, z.B. in Form von Impulsdiagrammen: Werden bei Eingang bestimmter Sensorsignale die richtigen Aktoren angesteuert? Es lässt sich damit überprüfen, ob die Signalabfolgen überhaupt realisierbar sind. Während der virtuellen Inbetriebnahme (Teil des funktionalen Engineerings), die sich kurz vor dem eigentlichen Aufbau der Anlage befindet, also zu einem Zeitpunkt, in dem die Anlage komplett spezifiziert vorliegt, kann der Steuerungscode auf Fehlerfreiheit geprüft werden.

Was ist dazu nötig?

Innerhalb des Anlagenentwurfsprozesses sind verschiedene ingenieurstechnische Disziplinen beteiligt, die jeweils ihre eigenen, an den Bereich optimierten Software-Werkzeuge verwenden. Aus diesem Grund macht es wenig Sinn, ein neues Werkzeug zu entwickeln, das eine Verhaltensvalidierung zu unterschiedlichen Zeitpunkten der Planung zulässt. Innerhalb der Forschungsanstrengungen der Otto-von-Guericke-Universität Magdeburg fiel deshalb die Wahl auf die Entwicklung eines Simulations-Frameworks, das die Einbindung dieser bereichsspezifischen Werkzeuge erlaubt. Um die zuvor genannten Informationsmengen

erzeugen bzw. verarbeiten zu können, werden folgende Werkzeuge bzw. Werkzeugfunktionen benötigt:

- 3D-Visualisierung
- Verhaltensdesign
- 3D-Simulation
- Verhaltenssimulation

Mittels einer zuvor erstellten mechatronisch orientierten Komponentenbibliothek, in der zu jeder Produktionsressource sowohl die Geometrie und Mechanik als auch das Verhalten des Moduls hinterlegt ist, kann in der Software zur 3D-Visualisierung die Anlagenstruktur durch Komposition der Produktionsressourcen modelliert werden. Das modulspezifische Verhalten (modelliert in dem Werkzeug zum Verhaltensdesign) ist jedoch so erstellt, dass es entsprechend der drei identifizierten Anwendungsfälle Einstiegs- bzw. Ansteuermöglichkeiten für das Anlagenverhalten gibt. Es ergeben sich drei Ebenen, auf denen die Produktionsressourcen angesteuert werden können. Somit können sie je nach Detaillierungsgrad des Anlagenverhaltens, das durch den Ingenieur mit der Software zum Verhaltensdesign erstellt wird, angesteuert werden. Das modulspezifische Verhalten, egal auf welcher Ebene, braucht allerdings auch auf der Seite der digitalen 3D-Repräsentation Anknüpfungspunkte (modelliert im Werkzeug zur 3D-Visualisierung), so dass sich die Produktionsressource bei deren Ansteuerung entsprechend 'bewegt'. Damit sind alle notwenigen Informationen vorhanden, um das Anlagenverhalten in den Werkzeugen zur 3D- und Verhaltenssimulation simulieren zu können. Bild 2 zeigt schematisch das entwickelte Simulations-Framework. Um für die simulative Verhaltensvalidierung eine konsistente Informationsweitergabe zwischen den beteiligten Werkzeugen sicherzustellen, ist AutomationML als Datenaustauschformat sehr gut geeignet. Neben der mechatronisch orientierten Komponentenbibliothek kann in AutomationML auch die Projektdatei mit dem Anlagenverhalten und der Anlagenstruktur, die die verwendeten mechatronischen Produktionsressourcen beinhaltet, konsistent geführt werden. Dabei werden die Strukturen mittels CAEX abgebildet: die Komponentenbibliothek als 'SystemUniClassLib' und die Projektdatei in der 'InstanceHierarchy'. Das Anlagenverhalten sowie das jeweilige modulspezifische Verhalten werden mithilfe von PLCopen XML abgebildet und die Geometrie- und Mechanikinformationen in Collada abgelegt, wie es in [1] beschrieben ist. PLCopen XML und Collada werden dabei als externe Dateien aus dem AutomationML Dachformat CAEX heraus referenziert. Besitzen die Werkzeuge jedoch selbst keine AutomationML- Schnittstelle, müssen entsprechende Konverter zur Verfügung stehen. Innerhalb der Forschungsarbeiten wurde das Simulations-Framework prototypisch mit Software von KW-Software (Multiprog für das Verhaltensdesign, Proconos für die Verhaltenssimulation) und tarakos (taraVRbuilder für die 3D-Visualisierung, taraVRcontrol für die 3D-Simulation) umgesetzt.

Fazit

Das Datenaustauschformat AutomationML ist gut dafür geeignet, das modulare, erweiter- und anpassbare Simulations-Framework zu realisieren, mit dem Anlagenverhalten in den verschiedenen Phasen des Planungsprozesses von Anlagen auf seine Fehlerfreiheit überprüft werden kann. So kann der Ingenieur vor Weitergabe seiner Daten noch einmal überprüfen, ob das modellierte Verhalten dem beabsichtigten Verhalten entspricht oder auch dazu nutzen, anderen Mitarbeitern das intendierte Anlagenverhalten zu veranschaulichen. Das einmalige Anlegen oder Aufbereiten der mechatronischen Produktionsressourcen ist erstmal ein Mehraufwand, der sich jedoch durch die ermöglichte Wiederverwendung der Komponenten schnell amortisieren kann. Unternehmen können sich 'hausinterne' Simulations-Frameworks erstellen und somit ihre Engineering-Qualität und auch ihre Engineering-Effizienz steigern. ■

Literatur

[1] R. Drath (Editor): Datenaustausch in der Anlagenplanung mit AutomationML, Springer Verlag, 2010.

www.automationml.org

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

Autorin: Nicole Schmidt, Wissenschaftliche Mitarbeiterin Center Verteilte Systeme (CVS), Otto-von-Guericke Universität Magdeburg

AutomationML – Erreichtes und Zukünftiges

Serie AutomationML Teil 13: Der Stand dessen, was erreicht wurde, und die weiteren Entwicklungsschritte

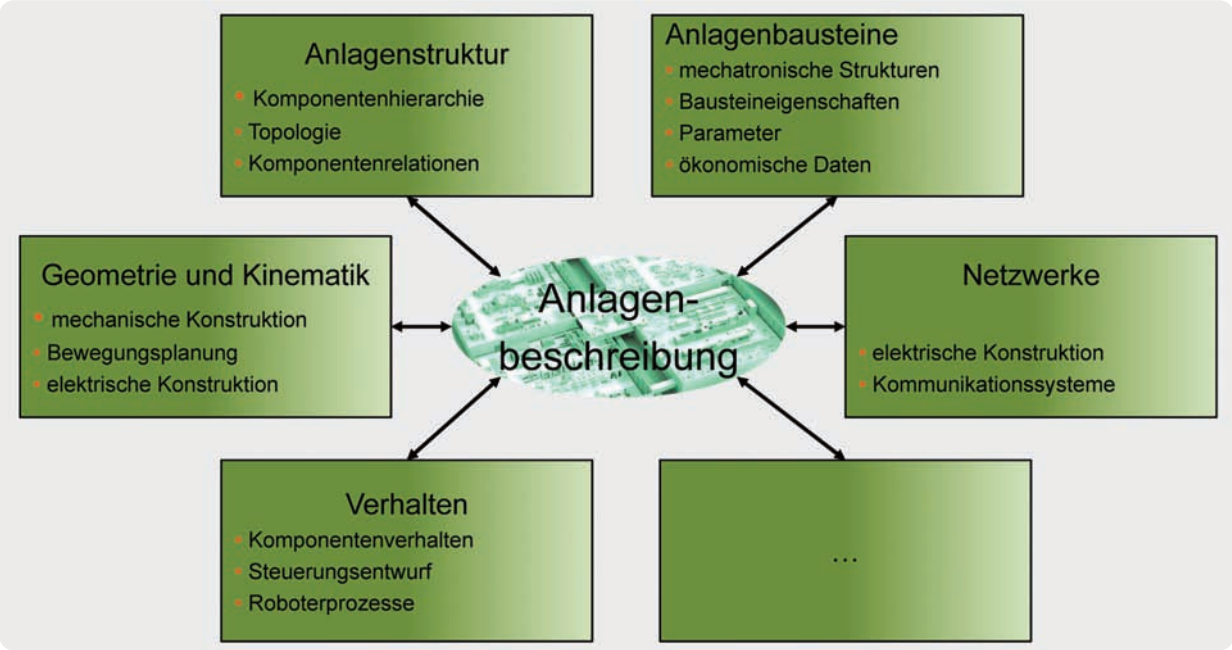


Bild: OvG-Universität Magdeburg

Bild 1: In AutomationML bisher abgebildete Informationsmengen

Im Jahre 2006 startete die AutomationML-Initiative mit dem Ziel, ein Datenaustauschformat zu schaffen, das die Entwurfswerkzeuge des Lebenszyklus eines Produktionssystems verbinden kann. Damit sollte dem Bedarf nach verlustfreiem Datenaustausch und damit der Reduzierung von wiederholten Entwurfsschritten in verschiedenen Werkzeugen und von Fehlern nachgekommen werden. Jetzt, acht Jahre später und durch den 2009 gegründeten AutomationML e.V. gepflegt, hat sich AutomationML als Technologie für den Datenaustauschprozess etabliert und weitere Anwendungen kommen stetig hinzu.

In den letzten zwölf Beiträgen dieser Serie wurden neben technologischen Grundlagen auch Anwendungen und Nutzungshinweise beschrieben. In diesem Artikel sollen nun der Stand des Erreichten zusammengefasst und die nächsten Entwicklungsschritte angedeutet werden.

Abgedeckte Informationsmengen

Grundidee der Automation Markup Language, oder kurz der AutomationML, ist die synergetische Kombinierbarkeit aller Informationsinhalte, die im Entwurfsprozess von Produktionssystemen relevant sind und zwischen Entwurfswerkzeugen ausgetauscht werden sollen. Dabei wurden anfänglich vorrangig Informationen zur Anlagentopologie, zur Geometrie und Kinematik und zu steuerungstechnischem Verhalten untersucht. In der Zwischenzeit ist die abbildbare Informationsmenge stark angestiegen, was Bild 1 verdeutlicht. Neben den klassischen Informationsmengen der Mechanikkonstruktion (Anlagentopologie, Geometrie, Kinematik), der Roboterprogrammierung (Anlagentopologie, Geometrie und Kinematik, steuerungstechnisches Verhalten) und der Steuerungsprogrammierung (Anlagentopologie, steuerungstechnisches Verhalten) kamen in den

Bild: OvG-Universität Magdeburg

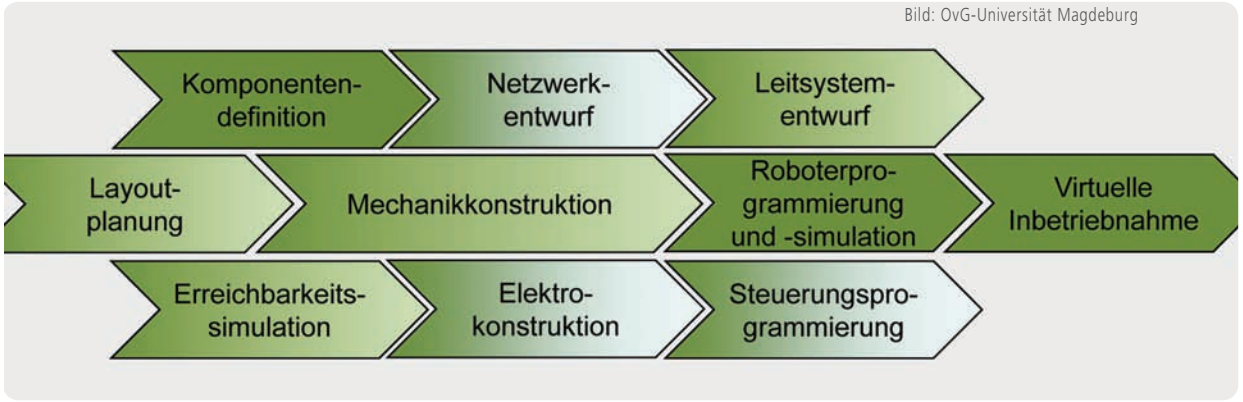


Bild 2: Entwicklungsstand von AutomationML

letzten Jahren Informationen für die Beschreibung von Netzwerken sowohl für die Beschreibung von Kommunikationsnetzen als auch für elektrische, hydraulische oder pneumatische Netze, die konsistente Beschreibung von Anlagenbausteinen für den mechatronischen Entwurf und die virtuelle Inbetriebnahme und die Beschreibbarkeit von grundlegenden betriebswirtschaftlich relevanten Eigenschaften (z.B. für den Einkauf) hinzu. Damit kann AutomationML prinzipiell die gesamte grundlegende Informationsmenge, die im Entwurfsprozess eines Produktionssystems relevant ist, abbilden.

Bestehende Anwendungen

Für die meisten der Entwurfsschritte im Entwurfsprozess konnte der Nachweis der Nutzbarkeit von AutomationML bereits in Pilotanwendungen erbracht werden. In einigen Anwendungsbereichen ist bereits die 'Serienreife' erreicht, das heißt, das Datenformat hat den produktiven Einsatz erreicht. Bild 2 gibt das anschaulich wieder. Je grüner hier der Entwurfsschritt dargestellt ist, desto weiter ist die Anwendung von AutomationML fortgeschritten. Einige der Anwendungshighlights sind die folgenden:

- Bei den Firmen Daimler und Audi wurden im Jahre 2013 umfassende Teile neuer Anlagen zur Produktion von PKW-Karosserien virtuell in Betrieb genommen. Dazu hat z.B. die Firma Rücker EKS eine entsprechende, auf AutomationML basierte Werkzeugkette entwickelt, die produktiv im Einsatz ist.
- Die Firma Lenze nutzt AutomationML zum Datenaustausch beim Entwurf von Antriebssträngen und hat es dazu in den Drive Solution Designer integriert.
- ABB nutzt AutomationML im Rahmen seines Werkzeuges 'RobotStudio'. Hier werden notwendige Geometrie- und Kinematikinformationen für die Roboterprogrammierung und -simulation ausgetauscht.

Neben diesen produktiven Anwendungen sind bereits in Piloten beim Fraunhofer IOSB, bei der tarakos GmbH, der Universität Magdeburg und der Cenit AG (um nur einige zu nennen) nachgewiesen worden, dass die Layoutplanung, die Mechanikkonstruktion, die Erreichbarkeitssimulation und der Leitsystementwurf entsprechend unterstützt werden können. Für den Netzwerkentwurf, die Elektrokonstruktion und die Steuerungsprogrammierung sind entsprechende Pilotanwendungen derzeit in der Umsetzung.

Besondere Vorteile

In den verschiedenen Anwendungen und Piloten kommt eine der wichtigsten Eigenschaften des AutomationML-Datenformates zum Tragen: die semantische Flexibilität und Erweiterbarkeit. Dazu nutzt AutomationML eine Struktur mit drei Bibliothekstypen. Insbesondere die Rollenklassenbibliotheken (RoleClassLib) und die Komponentenbibliotheken (SystemUnitClassLib) ermöglichen eine anwendungsfallspezifische Erweiterung der austauschbaren Semantiken. Hier können standardisierte und noch nicht standardisierte/ private Inhalte koexistieren und gemeinsam eindeutig ausgetauscht werden. Dazu können Werkzeugschnittstellen anwendungsfallspezifisch konfiguriert und der aktuelle Anwendungsfall über globale Projektattribute erkannt werden. Bild 3 zeigt dazu eine übliche Projektstruktur. Damit löst AutomationML eines der wichtigsten Probleme der Standardisierung: das wechselseitige Warten von Werkzeugentwicklern und Anwendern. Es muss nicht mehr die 'große standardisierte Lösung' abgewartet werden. Bereits mit kleinen standardisierten Bausteinen, die wichtige Probleme lösen können, kann produktiv gearbeitet werden. Diese 'kleinen Lösungen' können bei Erfolg erweitert und standardisiert werden.

Standardisierung

Einen wichtigen Teil der Arbeiten des AutomationML e.V. bildet neben der eigentlichen Weiterentwicklung und Verbrei-

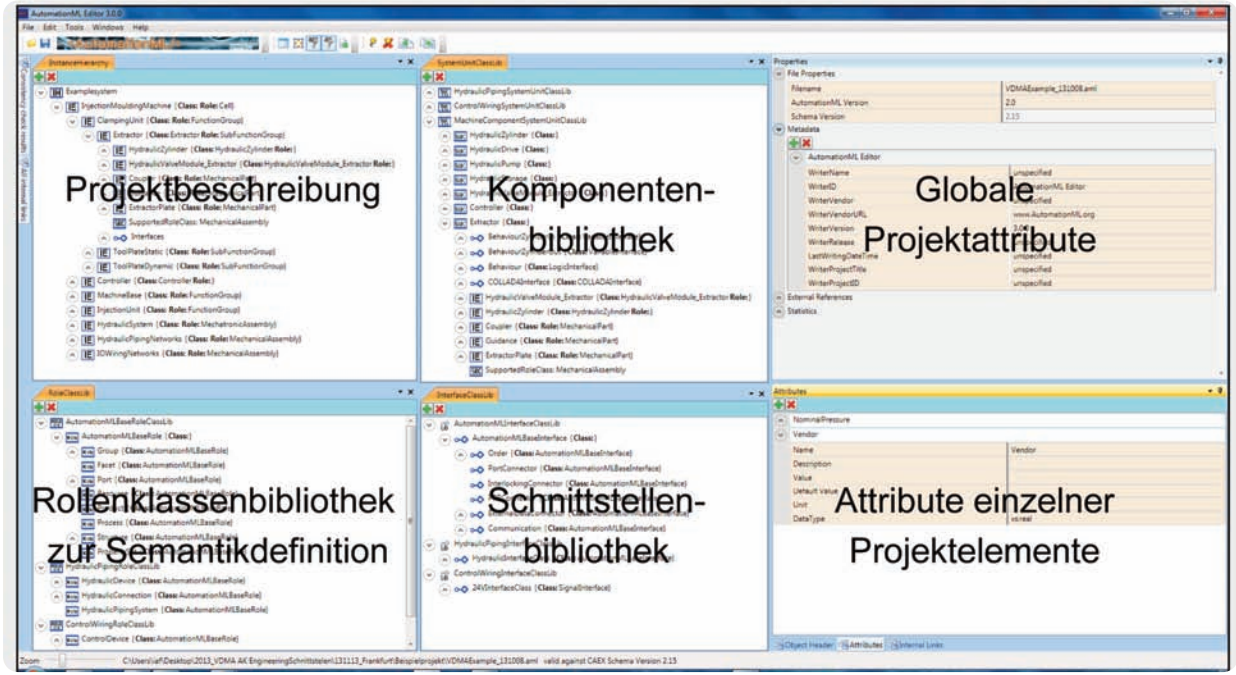


Bild: OvG-Universität Magdeburg

Bild 3: Grundlegende Struktur des AutomationML-Datenformates visualisiert im AutomationML Editor

tung des AutomationML-Datenformates auch dessen internationale Standardisierung. Diese erfolgt in sehr enger Zusammenarbeit mit der Arbeitsgruppe 'Engineering' der Deutsche Kommission Elektrotechnik Elektronik Informationstechnik (DKE) und der Workgroup 9 des SC 65 E der International Electrotechnical Commission (IEC). Um eine schnelle Standardisierung zu erreichen, wird dabei eine Normenreihe entwickelt, die unter dem Titel 'Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language' steht und die Projektnummer IEC62714 besitzt. Bisher sind mit den Teilen für die Beschreibung der grundlegenden Architektur, die Nutzung von Bibliotheken, die Geometrie- und Kinematikbeschreibung, die Verhaltensbeschreibung und die Beschreibung von Kommunikationsnetzwerken fünf Teile geplant. Es wird aber mit Sicherheit weitere Teile geben. Bild 4 zeigt den derzeit erreichten Entwicklungsstand. Dabei kann stolz vermeldet werden, dass nach nur acht Jahren Entwicklungszeit der erste Teil der IEC-Reihe den Status einer internationalen Norm erreicht hat. Die anderen werden in schneller Abfolge folgen.

Kooperationen

Von Anfang an wurde an die Entwicklung des AutomationML-Datenformates der Anspruch gestellt, das Rad nicht noch einmal zu erfinden. Bestehende und erprobte Technologien zu erkennen und passend zu kombinieren machten es möglich, die Entwicklung schnell voranzutreiben. Dazu waren und sind Kooperationen mit anderen Organisationen sinnvoll, die bereits Technologien entwickelt haben, die für AutomationML von Nutzen sein können. Begonnen hat dieses Modell mit den erfolgreichen Kooperationen mit der Khronos Group zur Anpassung und Integration des Collada-Datenformates für die Geometrie und Kinematikbeschreibung sowie mit der PLCopen zur Nutzbarmachung des PLCopen XML- Datenformates für die Verhaltensbeschreibung. Im Jahre 2013 kamen drei weitere Kooperationen hinzu. Gemeinsam mit dem eCl@ss e.V. wird die semantische Eindeutigkeit von beschriebenen Anlagenobjekten verbessert. Dazu werden entsprechende Methoden zur Rollenklassendefinition aus eCl@ss-Katalogen entwickelt. In Kooperation mit der OPC Foundation wird die Übertragung von AutomationML-Projekten mittels OPC UA untersucht. Ein weiterer Blick auf die Beschreibung von Geometrie und Kinematik erfolgt gemeinsam mit dem ProStep iViP e.V.

Aktuelle Arbeiten

Neben den genannten Arbeitsgebieten zur semantischen Eindeutigkeit von beschriebenen Anlagenobjekten, zur Nutzung von OPC UA und zu anderen Mitteln für die Beschreibung von Geometrie und Kinematik werden im AutomationML e.V. derzeit noch weitere wichtige Themenstellungen bearbeitet. Diese betreffen z.B. die Sicherstellung von Datensicherheit beim Datenaustausch (insbesondere über Mittel der Verschlüsselung und Authentifizierung), die Erweiterung der Nutzbarkeit für die virtuelle Inbetriebnahme, die Beschreibung von Produktionsprozessen, die konsistente Modellierung (mechatronischer) Komponenten für den Anlagenbau und die Erweiterung der Möglichkeiten zur Verhaltensbeschreibung

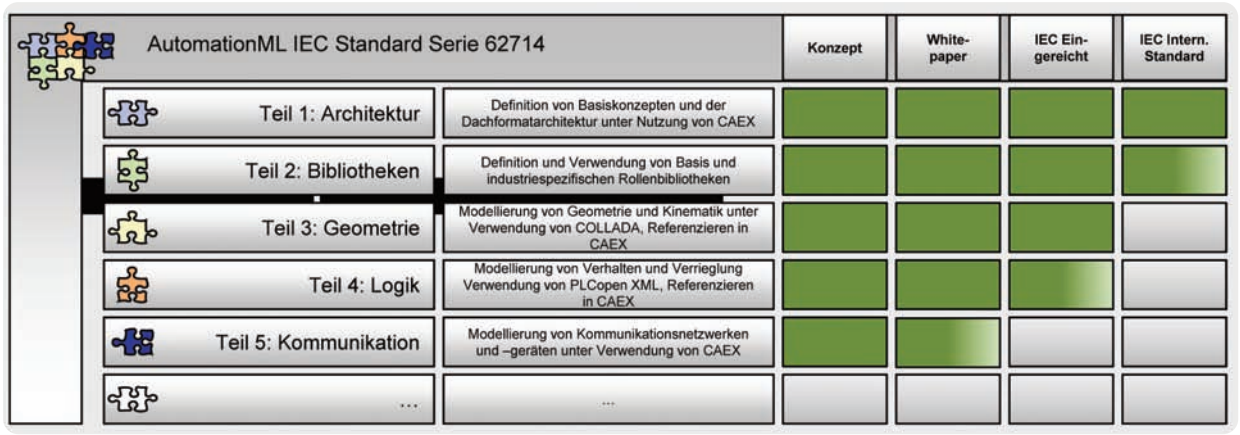


Bild: OvG-Universität Magdeburg

Bild 4: Stand der Standardisierung des AutomationML-Datenformates

über mathematische Formelsätze und Funktionsblocknetzwerke. In diesem Rahmen gibt es auch lose Zusammenarbeiten mit dem VDA, der NAMUR und dem VDMA. Jedoch können interessierte Unternehmen jederzeit neue, für sie wichtige Themenfelder initiieren oder sich in der Arbeit der bestehenden Themenfelder mit ihren speziellen Anwendungsproblemen engagieren. Sie sind sogar dazu aufgerufen, um die Schlagkraft der Entwicklungen und ihre bereits hohe Geschwindigkeit weiter zu erhöhen. Informationen zur Mitarbeit finden sich auf der Homepage des AutomationML e.V. unter www.automationml.org/o.red.c/mitgliedschaft.html.

Nutzersupport

Da die Anwendung des AutomationML-Datenformates stark von der Verbreitung entsprechender Export- und Importschnittstellen in Entwurfswerkzeugen abhängt, bietet der AutomationML e.V. eine breite Palette von fachlicher und technischer Unterstützung für interessierte Anwender an. Neben dem in Bild 3 bereits gezeigten AutomationML-Editor, einem Werkzeug zur einfachen Erstellung von AutomationML-Beispielen, zum Erlernen des Datenformates und zum Testen von Schnittstellen, stehen softwareseitig eine freie Bibliothek für das Erstellen, Verändern und Speichern von AutomationML-Projekten (die AutomationML Engine) und ein Testcenter zur Prüfung der Standardkonformität von AutomationML-Projekten zur Verfügung. Beide können für eine schnelle und fehlerfreie Entwicklung von Schnittstellen verwendet werden. Um entsprechendes Wissen über die Nutzbarkeit von AutomationML in spezifischen Anwendungen und das Aussehen entsprechender Strukturen zu verbreiten, hat der AutomationML e.V. eine Reihe von Veröffentlichungen sowie von Anwendungsbeispielen bereitgestellt. Beides, die verfügbare Software und die Veröffentlichungen und Beispiele stehen unter www.automationml.org/o.red.c/dateien.html zum Download bereit. ■

www.automationml.org

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

Autorin: Nicole Schmidt, Wissenschaftliche Mitarbeiterin Center Verteilte Systeme (CVS) Otto-von-Guericke-Universität Magdeburg