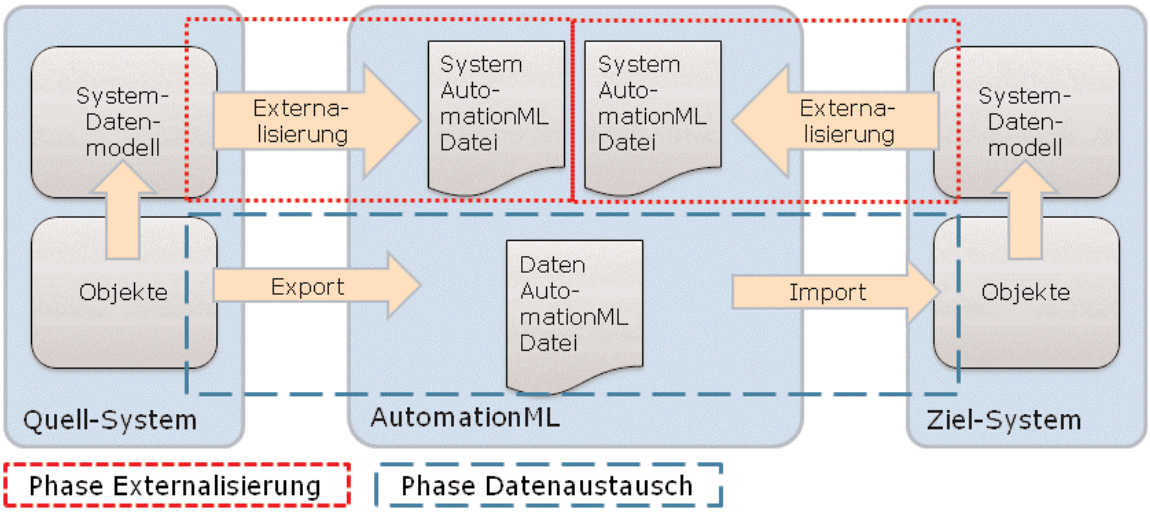

Bild 1: Phasen der Zuordnung

AutomationML – Datenaustausch

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

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.

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-System-Unit-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. So-

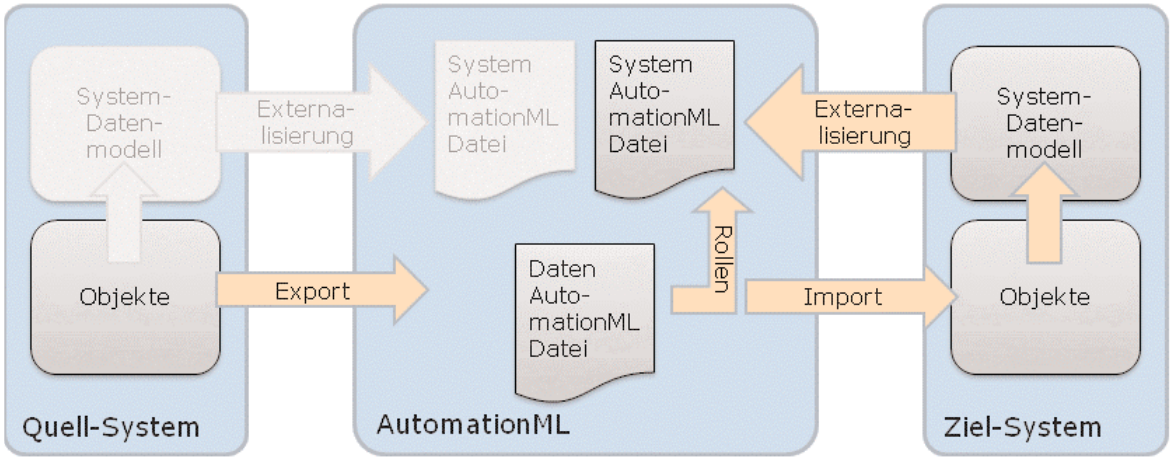

Bild 2: Zuordnung über Rollen

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

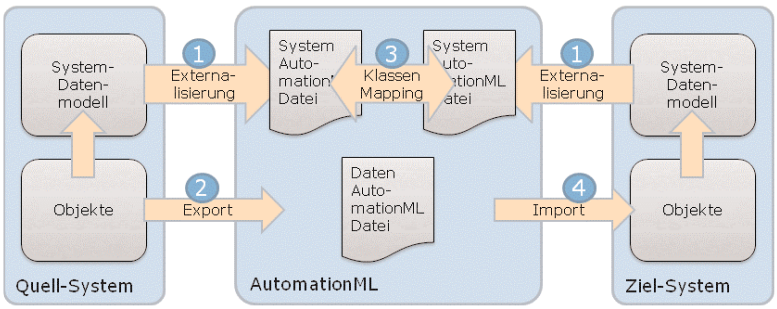

Bild 3: Zuordnung über Klassen

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

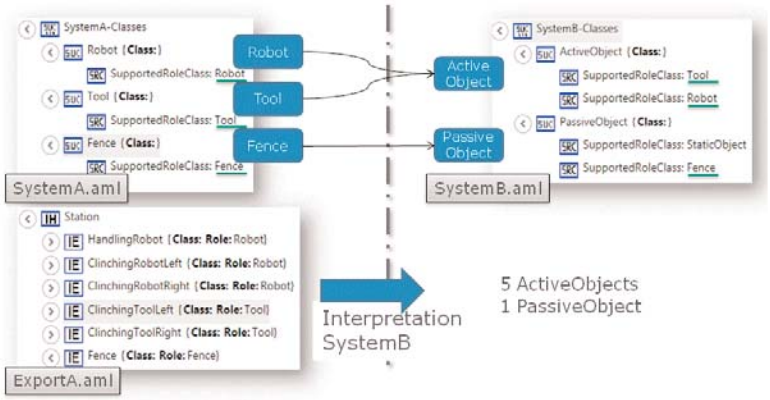

Bild 4: Beispiel Zuordnung über Rollen

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 diesem 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. ■

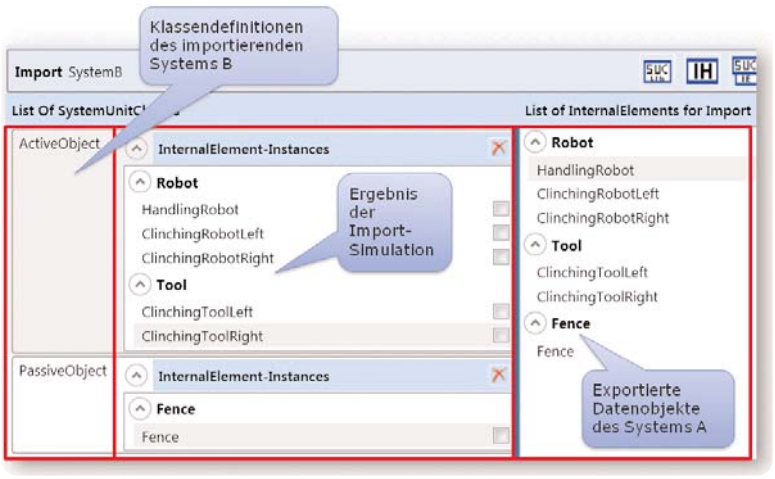

Bild 5: Oberfläche 'AutomationML Mapper'

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