Towards Model-based Product Engineering: Transformation zwischen OPC UA Informationsmodellen und SysML-v2-Systemmodellen

Towards Model-based Product Engineering: Transformation between OPC UA information models and SysML v2 system models

Niklas Bönisch[*,1], Christian Plesker[1], Adrian Reuther[1], Benjamin Schleich[1]

[1] Product Life Cycle Management, Technical University of Darmstadt

** Korrespondierender Autor:*
Niklas Bönisch
Product Life Cycle Management
Otto-Berndt-Straße 2
64287 Darmstadt
☎ *+49 6151 16-21847*
✉ *boenisch@plcm.tu-darmstadt.de*

Abstract

The configuration of modern manufacturing systems is often complicated by the use of vendor-specific tools and the need for specialist knowledge. This work presents an approach that automatically generates SysML v2 system models from OPC UA information models. This approach has the advantage of reducing the amount of manual effort required, thereby facilitating the adoption of model-based design methodologies in the field of manufacturing systems. The demonstration of this approach is illustrated through the utilisation of industrial robots, thereby exemplifying the transformation of structural and functional data from OPC UA into SysML-v2 constructs. Furthermore, adapted system models can be partially reflected back into OPC UA. While this bidirectional transformation supports tighter integration between product development and production, limitations remain regarding process logic and machine-specific constraints.

Keywords

OPC UA, SySML v2, MBSE

 | DOI: 10.35199/dfx2025.20

1. Motivation

In der modernen Fertigung kommen häufig Kombinationen von Maschinen unterschiedlicher Hersteller zum Einsatz. Die Inbetriebnahme und der Betrieb dieser Fertigungssysteme stellen in diesem Zusammenhang jedoch oft noch eine Herausforderung dar, da die Maschinen individuell konfiguriert werden müssen [1]. Die dafür verwendeten proprietären Systeme, Schnittstellen und Software führen zudem nicht selten zu Interoperabilitätsproblemen. Die zunehmende Digitalisierung und Vernetzung der Systeme sorgt außerdem für eine steigende Komplexität, was den Umgang für die Benutzer weiter erschwert. Um dieser Komplexität entgegenzuwirken, ist die Einführung neuer Methoden erforderlich, die den Benutzern im Umgang mit den Systemen unterstützt.

Ein vielversprechender Ansatz ist in diesem Kontext das sogenannte Model-based Systems Engineering. Dieses setzt auf den Einsatz von Modellen, um beispielsweise Anforderungen, Systemarchitekturen, Systemänderungen und mehr zu definieren und zu verwalten. Für die Erstellung dieser Modelle werden Modellierungssprachen wie SysML-v2 eingesetzt. Die Erstellung und Verwaltung dieser Modelle erfordert jedoch ein entsprechendes Fachwissen und ist oft mit einem hohen Zeitaufwand verbunden [2]. Die Modellierung bestehender Systeme wird insbesondere durch unzureichende Dokumentation [3] sowie hohe Abhängigkeiten der Komponenten erschwert. Die automatisierte Erstellung von Systemmodellen würde demnach zu einer signifikanten Reduzierung des Zeit- und Kostenaufwands führen. Ein möglicher Ansatz dafür wäre eine automatisierte Generierung der Systemmodelle auf Basis bereits vorhandener Informationen aus anderen Standards.

In der vorliegenden Arbeit wird dazu der Standard Open Platform Communication Unified Architecture näher untersucht. Dieser Standard wird für die Vernetzung von Maschinen und zur Überwindung von Interoperabilitätsproblemen unterschiedlicher Maschinen eingesetzt und bietet durch seine Modellierungseigenschaften eine geeignete Grundlage für die Generierung von Systemmodellen. Ziel der Arbeit ist es, zu untersuchen, inwiefern eine direkte Generierung von SysML-v2-Systemmodellen aus dem Standard Open Platform Communication Unified Architecture möglich ist und inwiefern Änderungen an den abgeleiteten Systemmodellen wiederum zurückgeführt werden können.

2. Stand der Technik

Der Einsatz von Systemmodellen ermöglicht die Repräsentation von Herstellungsprozessen [4] und deren zentrale Nutzung im Umgang mit den Systemen, wie beispielsweise zur Rekonfiguration. Zur Beschreibung und Erstellung dieser Systemmodelle wird in dieser Arbeit die Modellierungssprach SysML-v2 genutzt.

2.1. SysML-v2

SysML-v2 steht für „System Modeling Language – Version 2” und ist eine von der Object Management Group herausgegebene Modellierungssprache. Die Sprache wird in der Spezifikation „OMG Systems Modeling Language™ (SysML®) Version 2.0 Beta 4” definiert [5]. Neben der Modellierungssprache selbst werden in dieser Spezifikation mehrere Konzepte aufgeführt, die dazu dienen, Systeme, deren Komponenten und die externe Umgebung standardisiert zu modellieren und zu beschreiben. SysML hat das generelle Ziel, durch seine Modellierungsfähigkeiten das Model-Based Systems Engineering zu unterstützen und weiter voranzutreiben. Trotz der stetigen Bemühungen der Herausgeber, SysML weiter zu verbessern und zu präzisieren, ist SysML aufgrund seines breiten Portfolios und seiner umfassenden Spezifikation schwer zu erlernen [2] und erfordert oft Fachwissen. Zudem kann der Umgang mit den Modelle oft zeitaufwändig sein um die Anforderungen der jeweiligen Branche zu erfüllen [2].

2.2. Open Platform Communication Unified Architecture

Ein Lösungsansatz zur automatisierten Generierung von Systemmodelle bieten Standards, welche Interoperabilitätsprobleme von Systemkomponenten durch einheitliche Beschreibungen lösen. Ein geeigneter Standard dafür ist Open Platform Communication Unified Architecture, kurz OPC UA, welcher sich mittlerweile als der Kernstandard von Industrie 4.0 zur Verbindung von Fertigungsprodukten, -anlagen und -prozesssoftware etabliert hat [6]. Ziel von OPC UA ist es eine plattformunabhängiger Kommunikation zwischen verschiedenen Systemen und Geräten durch einheitliche Informations-, Nachrichten-, Kommunikations- und Konformitätsmodelle zu ermöglicht [7]. Nutzer haben die Möglichkeit, zwischen einer Client-Server- sowie einer Publisher-Subscriber-Architektur zu wählen. Mittels eines Servers besteht für Nutzer die Möglichkeit, Daten über einen sogenannten *AddressSpace* öffentlich zu machen. Die Daten werden durch Knoten und Kanten repräsentiert, welche im Informationsmodell des *AddressSpace* definiert sind. Neben dem durch OPC UA standardmäßig beschriebenen Informationsmodell existieren branchenspezifische Erweiterungen, die sogenannten *Companion Specifications*, welche den Austausch branchentypischer Maschinen interoperabel gestalten sollen. Darüber hinaus besteht für jeden Hersteller die Option, eigene Informationsmodelle, sogenannte *Vendor Specifications*, zu ergänzen.

In den Informationsmodellen erfolgt die Spezifikation der Knoten und Referenzen. Die für diese Arbeit wichtigsten Knoten sind Objekte-, Variablen- und Methodenknoten. Die Beziehungen zwischen diesen Knoten werden mittels Referenzen abgebildet. Die Definition der Knoten und Referenzen erfolgt durch die zugehörigen Typen wie beispielsweise Objekttyp, Variablentyp und Referenztyp. Die daraus resultierenden Modellierungseigenschaften können genutzt werden, um reale Maschinen, wie etwa Roboterarme, im *AddressSpace* des OPC UA Server zu repräsentieren. Die in den Branchen vereinbarten *Companion Specifications* definieren die Struktur und die zu implementierenden Knoten für branchenspezifische Maschinen. Die Schaffung einer standardisierten Repräsentationsschicht durch OPC UA ermöglicht so die herstellerunabhängige Maschinenanbindung. Gemäß den getroffenen Vereinbarungen implementieren Hersteller ihre proprietären Lösungen hinter den festgelegten Knoten und haben die Möglichkeit, diese gemäß den Vorgaben zu erweitern. Der Zugriff auf den *AddressSpace* erfolgt mittels eines durch OPC UA definierten Service Sets. Dies ermöglicht das standardisierte Auslesen oder Schreiben der Knoten.

3. Problemstellung und Forschungsfragen

Die manuelle Erstellung von Systemmodelle für komplexe technische Systeme erweist sich als ein aufwändiger und kostenintensiver Prozess [8]. Insbesondere erfordert die Modellierung ein tiefgreifendes Verständnis der zugrundeliegenden technischen Strukturen und stellt für Ingenieure eine erhebliche Belastung dar. Während Gaiardelli et al. [9] sowie Libro et al. [10] die Existenz eines vollständigen SysML-Modells als Voraussetzung für eine OPC UA-gesteuerte Prozessausführung betrachten, zeigen Olbort et al. [11] sowie Rekik et al. [12], wie sich Informationen aus SysML-Modellen in OPC UA-Informationsmodellen überführen lassen. Allerdings gibt es bisher keine etablierten Methoden, mit denen man umgekehrt aus bestehenden OPC UA-Informationsmodellen automatisiert ein SysML-v2-Systemmodell generieren kann. Rekik et al. [12] betonen, dass die Rückführung bislang weitgehend unerforscht ist. Eine automatisierte Generierung solcher Systemmodelle würde jedoch deutliche Vorteile versprechen. Zum einen kann der Aufwand für die Modellierung durch Ingenieure signifikant reduziert und zum anderen der Entwicklungsprozess beschleunigt werden. Schließlich können auch die damit verbundenen Kosten gesenkt werden. Darüber hinaus trägt die Automatisierung zur Fehlervermeidung bei und erlaubt flexible, dynamische Anpassungen der Modelle [13]. In der aktuelle Forschung wird jedoch die Frage, inwieweit sich

OPC UA als Grundlage zur automatischen Generierung und Rückführung von SysML-v2-Modellen eignet, nur unzureichend behandelt [12]. Daraus ergeben sich folgende Forschungsfragen, welche in dieser Arbeit näher untersucht werden:

- Inwieweit eigenen sich OPC UA Informationsmodelle zur automatisierten Generierung von eines SysML-v2-Systemmodells?
- Inwieweit lassen sich Änderungen an einem generierten SysML-v2-Systemmodell wieder in OPC UA zurückführen?

4. Methodik

Zur Beantwortung der Forschungsfragen wird zunächst ein Anwendungsfall skizziert und die betrachteten Systemgrenzen festgelegt. Darauf aufbauend erfolgt eine Darstellung des Aufbaus eines SysML-v2-Systemmodells und der durch OPC UA bereitgestellten Informationen. Im nächsten Schritt wird die automatisierte Generierung eines SysML-v2-Systemmodells anhand der identifizierten Informationen in OPC UA untersuchen. Ausgehend von den erstellten Systemmodellen wird anschließend die Rückführung von Änderungen in OPC UA erforscht. Dazu sollen neu hinzugefügte Abläufe aus einem SysML-v2-Systemmodell in den AddressSpace eines OPC-UA-Servers zurückgeführt werden.

4.1. Anwendungsfall und Systemgrenzen

Zur automatisierten Generierung eines SysML-v2-Systemmodell aus OPC UA heraus wird eine Arbeitszelle mit einem handelsüblichen Industrieroboterarm betrachtet. Ausgangslage ist die Repräsentation der Arbeitszelle mittels OPC UA. Dafür ist der Roboterarm an einen eigenen OPC UA Server angeschlossen und standardisiert nach der *Companion Specification for Robotics* abgebildet.

Die *Companion Specification for Robotics* [7] bietet, eine standardisierte Beschreibung, mit der sich verschiedene Roboter unterschiedlicher Typen herstellerunabhängig darstellen lassen. In der aktuell vorliegenden ersten Version ist der Anwendungsbereich jedoch auf Systemüberwachung und Anlagenmanagement beschränkt. In der Spezifikation wird zunächst das sogenannte MotionDeviceSystem beschrieben, welches ein System darstellt, das aus folgenden Komponenten besteht: (1) das Motion Device, das den physischen Aspekt des Roboters beschreibt, (2) die Controller-Komponente als Softwareaspekt und (3) die Sicherheitszustände des Roboters.

Das *MotionDeviceSystem* ist ein Subtyp der *Components*, welche in der allgemeinen Spezifikation für Geräte (*OPC UA Device Integration - Device*) definiert sind. Für den Anwendungsfall bedeutet dies, dass auch die Modellierungselemente aus der *Device-Integration*-Spezifikation berücksichtigt werden. Der Anwendungsfall beinhaltet somit etwa die Methode *InitLock* oder die Variable *Locked* aus der *Device-Integration*-Spezifikation, aber auch Elemente wie der *Axes*-Ordner mit seinen enthaltenen Variablen aus der *Companion Specification for Robotics*.

Im Rahmen der zweiten Forschungsfrage soll untersucht werden, wie dann Änderungen am generierten SysML-v2-Systemmodell wieder in OPC UA zurückgeführt werden können. Zu diesem Zweck sollen im Systemmodell die Abläufe von Methoden des Roboterarms definiert und in OPC UA übertragen werden, um diese anschließend ausführen zu können.

4.2. Ausgangslage OPC UA Informationsmodell

Ausgangslage für die automatisierte Generierung von SysML-v2-Systemmodellen bildet die Repräsentation eines Systems in OPC UA. Alle Informationen über das System werden, wie bereits im Stand der Technik beschrieben, über Knoten und Referenzen im *AddressSpace*

eines OPC UA Servers bereitgestellt. Die Struktur im *AddressSpace* wird demnach durch das Informationsmodell und den entsprechenden *Companion Specifications* definiert. Für die automatisierte Generierung eines SysML-v2-Systemmodells müssen somit zuerst die dafür relevanten Informationen identifiziert werden.

Die *Companion Specifications* verwenden unterschiedliche NodeClasses, für die Definition der Typen. Die wichtigsten NodeClasses sind dabei *ObjectType*, *Variable* und *Method*. Zusätzlich werden *Folder* verwendet, welche auch die NodeClass *ObjectType* besitzen, jedoch unterschiedlich transformiert werden müssen.

Durch Knoten dieser *NodeClasses* können die grundlegenden Informationen über das Robotersystem beschrieben werden. Außerdem werden auch die *Methoden* beschrieben, welche das Robotersystem bereitstellt. In der *Companion Specification for Robotics* werden aktuell keine *Methoden* definiert, welche das Robotersystem bereitstellen soll. Dafür werden in der *Companion Specification for Devices Methoden* zum Sperren eins Systems definiert. Diese *Methoden* sollen auch im SysML-v2-Systemmodell dargestellt werden können.

Die *Companion Specification for Robotics* definiert einen *MotionDeviceSystemType* Typen, welcher Informationen über ein System aus Robotern beinhaltet. Enthalten sind die drei *Folder* Controllers, MotionDevices und SafetyStates. Wie im Anwendungsfall beschrieben, wird ein einzelner Industrieroboterarm betrachtet. Der *MotionDeviceType* stellt dabei alle Informationen über den Roboter, wie zum Beispiel die Variablen *Manufacturer*, *Model* oder den Folder *Axes* zur Verfügung. Beispielhaft ist das in Bild 1 dargestellt.

4.3. Entwicklung Transformation OPC UA zu SysML-v2

Aufbauend auf den verfügbaren Informationen aus dem OPC UA Informationsmodell eines OPC UA Servers soll nun eine Transformation entwickelt werden, welche die automatisierte Generierung eines äquivalenten SysML-v2-Systemmodelles ermöglicht. Die grundlegenden Transformationsregeln sind in Tabelle 1 dargestellt und werden im Folgenden näher beschrieben.

Tabelle 1: Transformationsregeln von OPC UA Informationsmodell zu SysML-v2-Systemmodell

OPC UA Informationsmodell	SysML-v2 Systemmodell
ObjectType	Part Definition
VariableType	Attribute
MethodType	Action
Folder	Multiplizität

In dieser Arbeit wird die Transformation ausschließlich unter dem Gesichtspunkt der Generierung eines SysML-v2-Systemmodells anhand der Companion Specification for Robotics betrachtet. Individuelle Erweiterungen oder eigene Knoten werden nicht behandelt. Die fertige, vorgeschlagene Transformation kann Bild 1 entnommen werden.

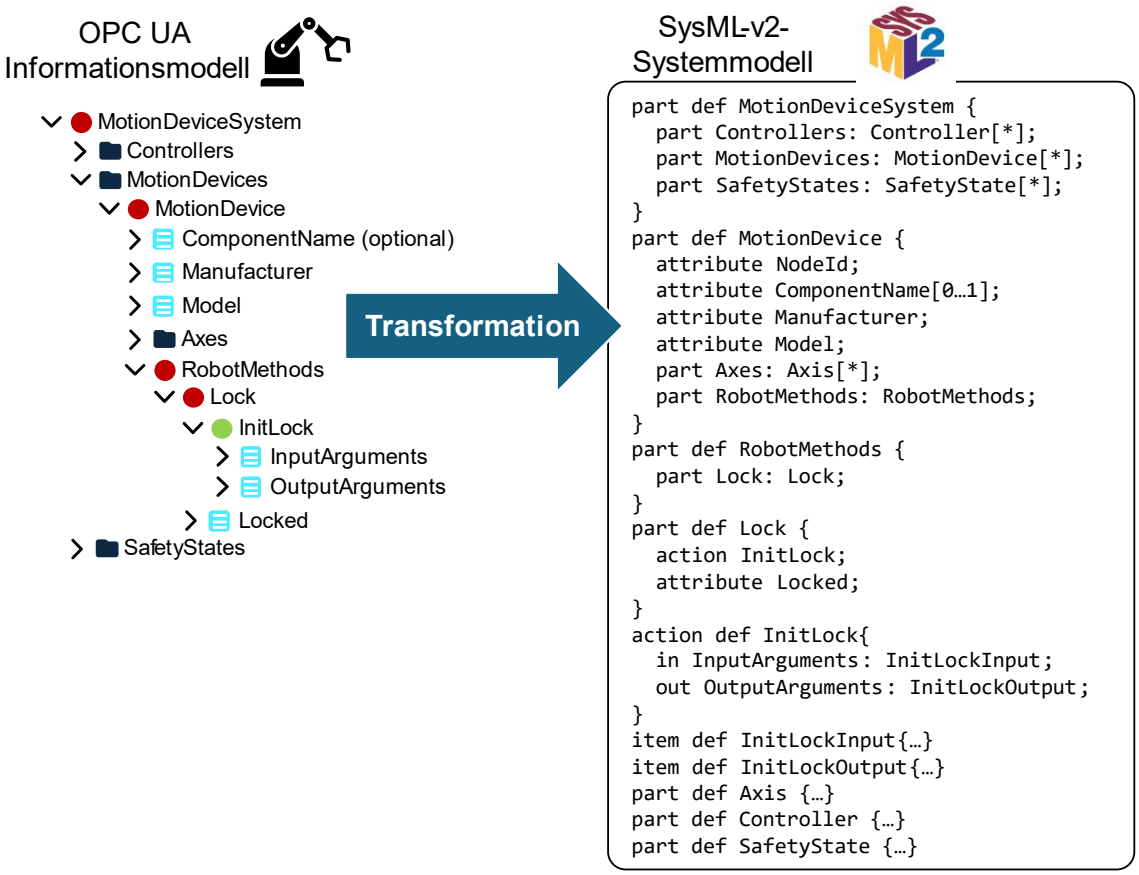

Bild 1: Transformation des Ausschnittest MotionDeviceSystem der Companion Spefication for Robotics.

4.3.1. ObjectTypes

Ein *ObjectType* definiert, wie ein *Objekt* aufgebaut ist. Da diese Objekte später im Systemmodell benutzt werden sollen, um ein konkretes Objekt dazustellen, werden die *ObjectTypes* in der Transformation zu *Part Definitionen*.

Der Type *MotionDeviceSystemType* wird entsprechend zu *part def MotionDeviceSystem* (siehe Bild 1). Diese *Part Definition* enthält alle Informationen, welche auch in dem entsprechenden *Object* definiert werden. Enthalten sein können wiederum weitere *ObjectType*s, aber auch andere Elemente, wie zum Beispiel Variablen, Methoden oder Folder.

4.3.2. VariableType

Die in einem *ObjectType* enthaltenen Variablen werden durch *VariableTypes* definiert und im SysML-v2-Systemmodell durch *Attribute* abgebildet. Diese *Attribute* können die entsprechenden Werte speichern, die auch in einer *Variable* gespeichert sein können. Ein Beispiel, wie eine Variable *Manufacturer* in der SysML-v2-Textnotation von *MotionDevice* dargestellt werden kann, wird in Bild 1 gezeigt. Variablen können verpflichtend oder optional sein. Wenn nichts weiter angegeben wird, sind die *Attribute* verpflichtend. Um auch optionale *Attribute* darzustellen, wird die Multiplizität [0…1] verwendet. Dadurch können null bis ein *Attribut* angegeben werden. Ein Beispiel dazu ist in Bild 2 dargestellt.

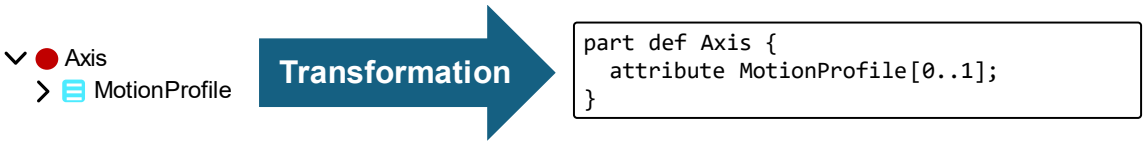

Bild 2: Transformation eines optionalen Attributes.

Knoten in OPC UA können auch Attribute besitzen, welche Informationen, über die Knoten selbst enthalten. So ist zum Beispiel die *NodeId* ein Attribut in OPC UA, welches bei der Rücktransformation wieder verwendet wird. Die OPC UA Attribute werden auch als *Attribute* in das SysML-v2-Systemmodell übertragen.

4.3.3. MethodeType

Methoden, definiert durch *MethodeType*, können aufgerufen werden und erlauben die Ausführung von Aktionen. Als solches wird eine Methode zu einer *Action* transformiert. In OPC UA können Methoden Ein- und Ausgabewerte besitzen. Solche lassen sich über *Items* im SysML-v2-Systemmodell darstellen. Exemplarisch kann die Action *InitLock* aus Bild 1 betrachtet werden.

4.3.4. Folder

In den Companion Specifications werden *Folder* definiert, welche mehrere Objekte enthalten können. Ein Folder ist auch ein *ObjectType* kann aber nicht durch eine *Part Definition* modelliert werden. Da die Folder innerhalb eines *ObjectType* liegen ist es möglich die *Folder* als *Part* im SysML-v2-Systemmodell zu definieren. Durch die Verwendung der Multiplizität [*] wird das *Part* als *Folder* ausgewiesen. Siehe Beispiel des *MotionDeviceSystemType* aus der *Companion Specification for Robotics,* welches drei *Folder* Controllers, MotionDevices und SafteyStates beinhaltet (siehe Bild 1). Die *Part Definition* für das *MotionDeviceSystem* enthält somit die *Parts Controllers*, *MotionDevices* und *SafetyStates*, welche wiederum druch die entsprechenden *Part Definitionen* spezifiziert werden. Zusätzlich sind die Multiplizität angegeben. Ein Part *MotionDeviceSystem* kann somit eine beliebige Anzahl an *MotionDevices* Parts enthalten.

4.4. Durchführen der Transformation

Basierend auf dem vorherigen Kapitel ist es möglich, dass ein SysML-v2-Systemmodell aus den OPC UA Companion Specifications zu erstellt. Aus den Types *ObjectType*, *MethodeType*, *VariableType* und *Folder* ist es möglich ein SysML-v2-Systemmodell abzuleiten. Dabei werden die einzelnen Types über die Transformationsregeln in Tabelle 1 in SysML-v2 Elemente transformiert. Um die Transformation durchzuführen, wird für jede Node im OPC UA Server die korrespondierende Beschreibung im Systemmodell verwendet. Wie das an einem Anwendungsfall von einem Knickarmroboter aussieht, wird in Bild 1 dargestellt.

4.5. Rücktransformation

Wie eine Transformation von SysML v1 zu OPC UA basierend auf QVTo funktionieren kann wurde bereits durch [12] gezeigt. In dieser Arbeit wird hingeben SysML-v2 verwendent. Für die Hintransformation wurde ein Ansatz mit starren Transformationsregeln vorgestellt. Die entsprechenden Transformationsregeln sind in Tabelle 1 dargestellt. Im betracheteten

Anwendungsfall ist das OPC UA Informationsmodell durch die verwendenten Companion Specifications festgelegt. Eine vollständige Rücktransformation ist deshalb nicht sinnvoll. Stattdessen wird in dieser Arbeit ein Ansatz vorgestellt, welcher es ermöglicht, dass Abfolgen von *Actions* im SysML-v2-Systemmodell zurück in den OPC UA Server übertragen und dort als neue Methode bereitgestellt werden können.

Die aktuellen Companion Specifications ermöglichen es bisher noch nicht, Abläufe in den OPC-UA-Server zu übertragen. Die *Actions* werden in der gewünschten Reihenfolge als Liste an den OPC UA Server übergeben. In den betracheteten Companion Specifications werden unter anderem die Methoden *InitLock*, *RenewLock* und *ExitLock* definiert. Eine beispielhafte Liste für die Rücktransformation in den OPC UA Server sieht dann wie folgt aus:

$$InitLock\ (ns = 2; i = 6166) \rightarrow RenewLock\ (ns = 2; i = 6169) \rightarrow ExitLock(ns = 2; i = 6171)$$

Um die grundlegende Funktionalität des Rücktransformationsansatzes zu demonstrieren, wird deshalb festgelegt, dass erstellte Abläufe von Aktionen als Liste gespeichert werden. In dieser Liste werden den Actions aus dem SysML-v2-Systemmodell über die NodeId dier entsprechenden OPC UA Methoden zugeordnet. Dadurch können jedoch nur einfache Abfolgen von Aktionen ohne Logik dargestellt werden. Da die NodeIds auch den *NamespaceIndex* enthalten, ist es wichtig, dass die Namespaces zwischen der Hin- und Rücktransformation unter dem gleichen *NamespaceIndex* geladen werden, da sonst eine Zuordnung der NodeIds nicht mehr richtig funktionieren kann. Eine weitere Einschränkung besteht dadrin, dass keine neuen Methoden erstellt werden können; es können nur bestehende Methoden zu einer neuen Abfolge zusammengestellt werden.

5. Ergebnisse und Diskussion

Die Transformation wurde anhand von Industrieroboterarmen (Franka FR3 und einem UR5e) beispielhaft durchgeführt. Dafür wird die entwickelte Transformation auf die Daten angewendet, welche der OPC UA Server des Roboterarms bereitstellt. Das resultierende SysML-v2-Systemmodell wird um einen Ablauf ergänzt. In der Rücktransformation wird dieser Ablauf über einen Export in den OPC UA Server übertragen.

5.1. Durchführung der Transformation von OPC UA Informationsmodell zu SysML-v2

In dieser Arbeit soll eine Transformation von OPC UA Informationsmodell zu einem SysML-v2 Systemmodell abgeleitet werden. Als Anwendungsfall wird dafür ein Knickarmroboter betrachtet. Dafür wurden ein Systemmodell für einen Knickarmroboter definiert in welches die Informationen aus dem OPC UA Informationsmodell transformiert werden können. Es werden keine Parent NodeIds benötigt, da durch die Struktur des Systemmodells jederzeit klar ist, welche NodeId das übergeordnete Objekt hat. Wichtig ist dabei allerdings, dass die OPC UA Namespaces unter derselben Namespaces ID eingebunden werden, da sonst die NodeIds nicht mehr richtig zugewiesen sind. Zusammenfassend ist die Mobilisierungsstrategie für das SysML-v2-Systemmodell stark an den OPC UA Companion Specifications angelehnt.

5.2. Bearbeiten des SysML-v2 Modells

Das resultierende SysML-v2-Systemmodell liegt nach der Transformation als Text vor. Dadurch ist eine Bearbeitung des Systemmodells möglich. Durch die aktuelle Einschränkung der *Companion Specifications* beschränkt sich die Durchführung der Bearbeitung auf den Fall, dass bestehende Methoden in einer Reihenfolge ausgeführt werden sollen. Im Anwendungsfall eines Knickarmroboters werden dafür die Methode *Lock*, dann die Methode *RenewLock* und dann *ExitLock* aufgerufen. Bearbeitungen die über einfache Abfolgen von Methoden hinausgehen, sind dennoch möglich, können aber nicht über die Rücktransformation in den OPC UA Server zurückgeführt werden.

5.3. Rücktransformation von SysML-v2 zu OPC UA

Für die Rücktransformation wird das erwähnte Beispiel betrachtet, in welchem die Aktionen *Lock*, *RenewLock* und *ExitLock* in dieser Reihenfolge aufgerufen werden sollen. Dasselbe Beispiel wurde auch für die Bearbeitung des Systemmodells verwendet. Die Methoden sind durch die Transformation in dem SysML-v2-Modell enthalten. Durch die textbasierte Notation lässt sich die Reihenfolge der Aktionen aus dem SysML-v2-Systemmodell extrahieren. Da die NodeIds erhalten bleiben, lassen sich die Aktionen wieder an den OPC UA Server, unter Einhaltung der Eindeutigkeit, übertragen. Durch einen angepassten OPC UA Server können die Aktionen als Methode über OPC UA bereitgestellt werden. Bei dem aufruf dieser Methode werden dann die Methoden, werden dann die Aktionen der reihe nach ausgeführt.

6. Zusammenfassung und Ausblick

Es wurde untersucht, wie aus einem OPC UA Informationsmodell eines Industrieroboters ein SysML-v2-Systemmodell abgeleitet werden kann. Dafür wurde die Companion Specification for Robotics und die Companion Specification for Devices als Grundlage genommen und eine Transformation in ein SysML-v2-Systemmodell entwickelt. Die entwickelte Transformation berücksichtigt dabei die unterschiedlichen Node

Zusätzlich wurde eine Rücktransformation beschrieben, durch welche es möglich wird eine Abfolge von Methoden in den OPC UA Server zurückzuführen. Dabei werden in dieser Arbeit zuerst nur starre Abfolgen von Methoden betrachtet. Die Nutzung von Schleifen und Verzweigungen, die in SysML-v2 grundsätzlich unterstützt ist, erfordert noch weiterführende Arbeiten. Aktuell wird in dieser Arbeit nur ein einzelner Knickarmroboter betrachtet. Aufbauend auf diesen Erkenntnissen kann der Ansatz auf mehrere Knickarmroboter übertragen werden. Das SysML-v2-Systemmodell stellt bereits die notwendigen Grundlagen dazu zur Verfügung, indem das *MotionDeviceSystem* aus mehreren *MotionDevices* bestehen kann. Zudem wird in der aktuellen Arbeit nicht betrachtet, wie die Namespaces im OPC UA Server unter unterschiedlichen Indizes geladen werden können und wie das im SysML-v2-Systemmodell abgebildet werden kann. Weiterhin werden die Namespaces aus dem OPC UA Server nicht in das SysML-v2-Systemmodell übertragen. Um zu bestimmen, wie die OPC UA Namespaces durch die Namespaces in SysML-v2 abgebildet werden können, sind weitere Arbeiten notwendig.

Literaturverzeichnis

[1] PROFANTER, Stefan ; BREITKREUZ, Ari ; RICKERT, Markus ; KNOLL, Alois: A hardware-agnostic OPC UA skill model for robot manipulators and tools. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, S. 1061–1068

[2] JANSEN, Nico ; PFEIFFE, Jérôme ; RUMPE, Bernhard ; SCHMALZING, David ; WORTMANN, Andreas: *The Language of SysML v2 under the Magnifying Glass*. In: *The Journal of Object Technology* 21 (2022), Nr. 3, 3:1

[3] ZIPPER, Holger ; AURIS, Felix ; STRAHILOV, Anton ; PAUL, Manuel: Keeping the digital twin up-to-date -- Process monitoring to identify changes in a plant. In: *2018 IEEE International Conference on Industrial Technology (ICIT)* : IEEE, 2018, S. 1592–1597

[4] KÖCHER, Aljosha ; HAYWARD, Alexander ; FAY, Alexander: Model-based engineering of CPPS functions and code generation for skills. In: *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2022, S. 1–8

[5] Version 2.0 Beta 4. 2025-04-05. *OMG Systems Modeling LanguageTM (SysML®)*

[6] LIN, Shi-Wan ; MURPHY, BRETT, CLAUER ; ERICH, LOEWEN, ULRICH ; NEUBERT, Ralf ; BACHMANN, Gerd ; PAI, Madhusudan ; HANKEL, Martin: *Architecture Alignment and Interoperability* (2017)

[7] OPC FOUNDATION: *OPC 10000-1 - UA Specification Part 1 - Overview and Concepts 1.05.04*

[8] SANFORD FRIEDENTHAL (Hrsg.): *Future Directions for MBSE with SysML v2* : SciTePress, 2023

[9] GAIARDELLI, Sebastiano ; SPELLINI, Stefano ; PANATO, Marco ; TADIELLO, Carlo ; LORA, Michele ; CHENG, Dong Seon ; FUMMI, Franco: *Enabling service-oriented manufacturing through architectures, models, and protocols*. In: *IEEE Access* 12 (2024), S. 85259–85274

[10] LIBRO, Mario ; GAIARDELLI, Sebastiano ; PANATO, Marco ; SPELLINI, Stefano ; LORA, Michele ; FUMMI, Franco: Exploiting SysML v2 modeling for automatic smart factories configuration. In: *2025 Design, Automation & Test in Europe Conference (DATE)* : IEEE, 2025, S. 1–7

[11] OLBORT, Johannes ; RÖHM, Benjamin ; KUTSCHER, Vladimir ; ANDERL, Reiner: *Integration of communication using OPC UA in MBSE for the development of cyber-physical systems*. In: *Procedia CIRP* 109 (2022), S. 227–232

[12] REKIK, Fadwa ; DHOUIB, Saadia ; NGUYEN, Quang-Duy: *Bridging the gap between SysML and OPC UA information models for industry 4.0*. In: *J. Object Technol.* 22 (2023), Nr. 2, 2:1

[13] ZHONG, Shaohong ; SCARINCI, Andrea ; CICIRELLO, Alice: *Natural Language Processing for systems engineering: Automatic generation of Systems Modelling Language diagrams*. In: *arXiv [cs.CL]* (2022)